numéro 21

Programmez votre CPC ou CPC+ avec

QUASAR

JS

- ACTUS
  - ~Another World s'arrête
  - ~notre futur
  - ~les prods
- TESTS
  - ~Starkos
- ASSEMBLEUR
  - ~hard : Multiface Two
  - ~coding : CRTC
- DOSSIER
  - ~les ROMs d'extension

Edition du Croco Chanel 4 - 4'S'E Mai - Printemps 2007

-sommaire ——— pages-

EDITORIAL

D'aucuns nous avaient déjà enterrés, mais l'heure n'est pas aux épitaphes et c'est bel et bien le numéro 21 de Quasar CPC que vous tenez entre vos mains fébriles ! Certes, notre retard est colossal, mais comme disait l'autre : vieux motard que jamais. En outre, vous noterez que notre atermoiement ne tient pas la comparaison face à celui de la Demo Iz Art... mais nous avons bon espoir de monter sur le podium avec l'Asic Inside un de ces jours !

Quoi qu'il en soit, c'est avec un grand plaisir que nous vous proposons ce nouvel opus à l'occasion du Croco Chanel N°4 qui sera sans doute l'événement CPCistique le plus important de ces 15 dernières années. Comme vous l'aurez compris, la morosité n'est pas à l'ordre du jour ! Certes, les temps changent et tout n'est pas rose. Je pense notamment à la fin d'Another World ; ce petit fanzine de scène facétieux va bien nous manquer. Mais, comme le dit pour une fois avec justesse notre petit blond à lunettes : the show must go on ! En ce qui nous concerne, malgré nos difficultés à paraître, nous sommes toujours là. Même s'il est vrai que nous envisageons quelques réorientations, nous n'avons pas l'intention de jeter l'éponge et de laisser au seul internet et aux PCistes le monopole du CPC !

Qu'avons-nous donc dans ce nouveau numéro ? Eh bien comme d'habitude, des articles techniques assez poussés qui contiennent une foule d'informations inédites concernant le CPC. Vous trouverez également un test plus que complet de Starkos, le fameux logiciel de musique de Targhan. Parallèlement, nous vous proposons quatre pages d'actus qui font le point sur les productions passées et à venir. Concernant la forme, nous n'avons pas dérogé à la règle puisque ce numéro est encore entièrement réalisé sur CPC. Toute la PAO est effectuée grâce à Oxford et AMX Pagemaker, les illustrations sont des scans réalisés avec le fidèle Dartscan et l'impression se fait sur une imprimante jet d'encre pilotée depuis un bon vieux CPC6128.

Je vois que le petit blond à lunettes s'agite comme une puce, je ne vais donc pas plus longtemps vous retarder et je vous invite à vous plonger dans ce nouveau -et non ultime- numéro de Quasar CPC !

Bonne lecture et à bientôt !

——— La Rédac'

ACTUS

Nous allons commencer ces actus par un petit état des lieux concernant les fanzines CPC. En effet, entre les vraies mauvaises nouvelles et les rumeurs plus ou moins exotiques, le petit blonds à lunettes ne sait plus quoi penser et j'en ai marre qu'il me tire sans cesse la manche en me demandant d'un air désorienté "c'est quand qu'on va où" ?

Le petit Another World est mort

Eh oui, cette année a été tristement marquée par la fin de cet acide fanzine de scène. Bien que récent dans la longue histoire du CPC, cette gazette avait su trouver sa place et rythmait la vie de la scène à coup d'articles caustiques et d'interviews truculentes. Leur ultime numéro, le 47, est donc sorti au mois de Mars 2007 et il ne manquera pas de vous faire sourire bien que toute fin soit triste. Cette page étant tournée, il y a toutefois fort à parier qu'ils ne vont pas en rester là, et je leur fais confiance pour continuer de bousculer tout ces petits CPCistes frileux ! Pour de plus amples informations, n'hésitez pas à les harceler :

THOMASETTE David & Franck
26, rue des maisonnettes
54300 LUNEVILLE
courriel : another-world@cpcscene.com

Les aventures de Quasar CPC

L'arrêt d'Another World fait donc de Quasar CPC le dernier fanzine (papier) et soulève de nombreuses interrogations quant à notre existence même. En effet, vous l'aurez remarqué, la parution de notre fanzine est de plus en plus chaotique ces dernières années et nous sommes de moins en moins nombreux à nous y impliquer. Eh oui, le petit blond à lunettes n'est finalement pas responsable de toutes les misères du monde et la vie est dure pour les fanzines papier ! Le profil des CPCistes a beaucoup changé et les utilisateurs de CPC se font rares. Alors que certains éminents CPCistes actifs se sont retirés, la plupart des personnes ayant de l'intérêt pour l'Amstrad travaillent désormais sur PC et s'orientent donc naturellement vers d'autres outils et d'autres médias. L'internet, en particulier, est un moyen très efficace pour permettre aux geeks s'intéressant au CPC de garder le contact. Notre fanzine n'est peut-être plus pertinent et une petite voix s'écriant "laisse béton" s'élève parfois.

C'est mon dernier bal ?

Mais à cette voix, nous allons lui tordre le cou ! En effet, je pense qu'un fanzine papier apporte quelque chose de plus et a l'avantage de laisser une trace et de marquer son temps. On est là et on y reste ! Nous n'avons pas envie d'arrêter Quasar CPC et, comme promis dans l'éditorial de notre numéro 1, nous allons continuer l'aventure avec Gérard Lamb... heu, avec le petit blond à lunettes tant qu'il y aura des gens pour nous lire... et, pour le moment, vous êtes encore nombreux puisque notre fanzine est diffusé en moyenne à 60 exemplaires avec des pointes à plus de 120 pour certains numéros.

Jusqu'à La fin du monde !

Certainement ! Mais ça ne veut pas dire qu'il ne faut pas se remettre en question en cours de route. En effet, Quasar CPC a une orientation particulière, c'est un fanzine assez technique développant divers dossiers simultanément d'un numéro sur l'autre. Ceci va éventuellement changer : il n'y aura peut-être jamais de numéro 22. Le petit blond à lunettes vient de faire un bond et se met à me hurler dans les oreilles : "c'est comme ça que vous voulez continuer ? En ne sortant pas de numéro 22 ?! Où c'est qu' j'ai mis mon flingue !?" Une bonne mandale calmera ses ardeurs pour un petit moment et me laissera certainement le temps de vous expliquer ce qu'il en est dans le calme.

Aucune décision n'a été prise pour le moment, mais nous explorons plusieurs pistes et votre avis sera le bienvenu. Nous envisageons, par exemple, de renoncer à la parution de numéros "à suites" pour ne vous proposer plus que des numéros hors-série thématiques. Ces numéros pourraient être annuels et seraient très complets sur leur sujet ; nous tenterions d'y faire participer un maximum de CPCistes compétents. Parallèlement, nous envisageons d'ouvrir un Quasar Net qui serait une espèce de wiki basé sur les publications existantes de Quasar CPC et, éventuellement, du défunt Amslive. Cela nous permettrait de rendre disponible sur le net les contenus les plus intéressants des anciens numéros de façon optimale, puisque ceux-ci pourraient évoluer et s'enrichir grâce à la participation de "tout le monde". Bref, pas question de mettre la clef sous la porte !

Quasar Disc 4 sur La Ligne de départ

Eh oui, comme c'est désormais la règle, ce numéro est accompagné d'un Quasar Disc. A l'heure où j'écris ces lignes, son contenu n'est pas encore figé. Toutefois, sachez que vous y retrouverez tous les listings publiés dans nos pages, ainsi que quelques outils pour Multiface Two et divers softs annexes. Il sera bien sûr également agrémenté d'une interview exclusive ! Pour vous le procurer, rendez-vous sur le site Futurs' ou bien commandez-le directement à la rédac' (pensez à nous envoyer une disquette vierge).

Futurs' de toi

Avant d'enchaîner sur les actus à proprement parler, faisons un petit point sur nos propres productions. Comme chacun sait (sic), nous avons divers projets en cours depuis... comment dire... hem... des années ! Il est dès lors plus que temps de liquider tout ça. Certains projets ont été officiellement abandonnés, c'est par exemple le cas d'Aventury et de Stormlord+. Nous sommes en train de nous organiser pour publier très prochainement le travail qui a été effectué. Concernant Aventury -qui pour mémoire était un (trop) ambitieux jeu de rôle- nous avons une foule de de graphismes et de musiques finalisés, mais peu de code réellement présentable ; ce travail sera donc sans doute publié sous la forme d'un slide show. Pour Stormlord+ (le clone de Stormlord pour CPC+ codé par SNN), les choses sont un peu plus intéressantes puisque, même s'il n'est pas fini, le jeu est fonctionnel. Nous allons donc le diffuser sous la forme d'une beta jouable.

En revanche, d'autres projets sont encore réellement d'actualité et, "dès que le vent soufflera", ça repartira. Les plus célèbres sont le Soundtracker DMA de Zik et l'Asic Inside. Pour le tracker, nous avons une bonne nouvelle puisqu'une version démo (bien plus complète que celle dévoilée lors du Croco Chanel Noï) va être diffusée sous peu. Ah, notre petit blond à lui a repris ses esprits et m'offre son sourire moqueur : "sous peu ? ça veut dire avant la fin du siècle ?". Que neni ! Le temps de la raillerie est révolu, wait and see. Vient ensuite notre plus belle casserole, à savoir Asic Inside. L'idée de sortir une megademo a été abandonnée car le travail est trop conséquent au regard du temps que nous sommes prêts à y consacrer. Comme il serait dommage de laisser tout ça dormir sur des disquettes, nous avons décidé de sortir une à une les différentes parts les plus avancées sous la forme de petites intros. Néanmoins, ne vous attendez à rien d'exceptionnel, il s'agit juste de publier les previews déjà vues par la plupart des CPCistes ayant écumés les meetings ces dix dernières années.

Outre ces deux productions réservées exclusivement aux CPC+, nous avons également en stock quelques babioles dédiées aux véritables CPC. Je ne vais vous donner ici plus de détails, aucune sortie imminente n'est prévue concernant ces jeux et démos, mais ça continue d'avancer, ça n'est pas à l'abandon. Par ailleurs, nous n'avons pas prévu de nous lancer dans de nouveaux développements : nous allons gérer et finaliser les productions déjà en cours et focaliser notre activité sur Quasar CPC et ses "produits dérivés".

Le dernier volet concerne nos projets hardware. Ceux-ci sont essentiellement gérés par Zik et, même s'il reste vivement intéressé, le temps lui manque cruellement et sa priorité CPC se porte plutôt sur le Soundtracker DMA. Ceci étant, qu'en est-il ? La Mélodie (ex-Soundplayer2, carte son) est pour ainsi dire au point mort. Il en va de même pour la SoundplayerNG dont les plans avaient été diffusés dans notre numéro 20 : elle n'est pas disponible. Petit rayon de soleil : il nous reste quelques kits de Soundplayer+, n'hésitez pas à nous contacter si vous êtes intéressés. Je vous rappelle qu'il s'agit d'une carte son et réseau compatible Digiblaster et Virtual Net 96 se connectant sur le port imprimante des CPC (non compatible avec les CPC+ pour la partie réseau).

C'est pas du pipeau

Le bilan ne serait pas complet si je ne disais pas un mot à propos du projet de Siou que nous avions tenté de promouvoir il y a quelques années : la CPC-ISA. Vu l'état du marché des cartes ISA de nos jours, il est peu probable que nous consacrions de nouveau du temps à ce projet. Il y a plusieurs v2.1 en circulation, ces cartes sont parfaitement fonctionnelles et des drivers ont été développés (cartes son, ide, série, etc.) et sont diffusés sous forme de ROMs. Les dernières versions sont disponibles sur le site Futurs'.

Croisière pour un cadavre

Si nous n'avons pas été productifs ces derniers temps, d'autres l'ont été ! En effet, c'est avec grand plaisir que nous avons vu Semilanceata sortir une demo pour CPC+. A vrai dire, il s'agit à mon avis de la première véritable démo CPC+ ! Même si la plupart des effets -à la thématique morbide- sont relativement simples, leur mise en scène est enfin digne du CPC+ et l'Asic est réellement exploité comme il aurait toujours dû l'être. En effet, la majorité (totalité ?) des démos CPC+ sorties jusqu'alors souffraient d'une programmation et d'une mise en scène calquées sur les techniques old, là où l'Asic permet normalement de réaliser des démos beaucoup plus proches de ce qui se fait sur la scène 16 bits. Sappy, c'est son nom, crée ici une véritable ambiance à la "Ephidrena" et met très bien en valeur des effets simples mais diversifiés dans une ambiance sonore très "mod". SML ayant dû mettre en place un bon nombre d'outils jusqu'alors inexistants sur CPC+ afin de développer cette petite démo de très bonne facture, ça serait dommage de ne pas en profiter pour nous proposer une suite avec des effets plus trapus ? On a l'eau à la bouche là, la suite messieurs !

Mais qu'est-ce qu'il a écrit !?

Groops!

De retour chez moi après un petit tour du monde, quelle ne fut pas ma surprise de découvrir une enveloppe frappée du sceau normand dans ma boîte-aux-lettres ! Après l'avoir ouverte fébrilement, j'y découvre une disquette arborant fièrement une étiquette multicolore marquée : "Groops!". Quel plaisir de recevoir ainsi à l'ancienne, par la poste, une bonne vieille disquette CPC ! Cerise sur le gâteau, son contenu est excellent ! En effet, Eliot, Slyder et Napo nous proposent là un petit casse-tête fort sympathique. Non seulement le jeu en lui-même est très intéressant (il fait penser à Puzznic au premier abord), mais en plus sa réalisation est impeccable ! Overscan intégral, plusieurs musiques vraiment bonnes, des graphismes tout à la fois clairs et détaillés, plusieurs modes de jeu, un mode d'emploi incorporé... bref, du tout bon ! Je ne vais pas vous en faire un test détaillé, ce n'est pas le propos, mais je vous conseille vivement de vous le procurer. Il est disponible sur l'internet sur le site http://www.cpcscene.com ou directement sur le site des auteurs : http://www.binarysciences.com. Vous découvrirez d'ailleurs sur ce site que quelque chose d'autre se prépare... Sudoku Master... les écrans sont prometteurs ! Miam !

Le Croco Chanel 4

Eh oui, c'est à l'occasion de ce meeting (prévu les 4, 5 et 6 Mai 2007) qui sera sans doute l'un des plus gros -sinon le plus gros- meeting CPC de tous les temps, que nous avons décidé de sortir ce numéro de Quasar CPC. Si vous y êtes présents, je devine que le meeting bat son plein au moment où vous lisez ces lignes. De nombreux anciens de la scène ont prévu de s'y rendre, je suis certain que ça donnera lieu à des discussions passionnantes. D'après toutes les annonces qui ont pu passer ces derniers temps nous devrions assister à une véritable avalanche de prods... voila un meeting fort bien engagé, ça sent bon "les paupières qui tombent et les yeux qui s'lézardent".

Amstrad Expo 2005 !?

Sur ce, je vous laisse avec une petite surprise qui parle "du bon temps qu'est mort ou qui r'viendra" : Le livre d'or de l'Amstrad Expo 2005. Bonne lecture et merci à tout ceux qui s'étaient prêtés au jeu !

Genesis8 : Un meeting de plus à bosser sur le site web et améliorer mon chtit programme pascal utilisé sur le site, bien rempli donc. L'ambiance est sympa comme d'habitude, manque de foie gras à la bouffe, mais j'ai fait sans. A la prochaine !

Poulette73 : Que de découvertes pour mon premier meeting !! Rien ne manquait : ambiance, bonne humeur, convivialité, tout ça au milieu d'un univers mélangeant les codes, graphs, zics, jeux et bidouilles hards... Les parties de LAP sur UN96 me manquent déjà ;-) Merci à OffseT pour sa patience car je l'ai pas mal dérangé par mes questions alors qu'il codait pour le concours !!

Fenyx Kell : Un vrai bonheur de rencontrer tant des gens si sympa et chaleureux et de partager cette passion pour le CPC. J'espère désormais pouvoir venir à chaque Amstrad Expo et même à d'autres meetings ! A bientôt !

Kris : Génial ! Une super ambiance, des gens très disponibles, une très bonne organisation ! Des premiers vrais cours d'asm, la route va être longue :) Merci à tous et vivement le prochain.

Giants : 1ère participation à un meeting. Bon souvenir en perspective :) Très bonne ambiance et une superbe organisation (Bravo Eliot). AE c'est sûr, on y reviendra.

Parano-Sprite : J'ai été très content de faire la rencontre de toutes les personnes de ce meeting. Etant mon 1er meeting, ce fut vraiment superbe. L'accueil était génial. Un grand merci à Eliot pour la préparation et l'organisation de ce meeting. Un grand merci à tous et longue vie à Quasar CPC ;))) Et aussi merci Supersly pour ton affiche...

Pulko Mandy : Super meeting, bonne ambiance, dommage que je doive déjà partir... Promis, la prochaine fois je me mets à l'assembleur ! (Enfin... si j'ai le temps...)

Roudoudou : Merci Eliot pour cette organisation. Entre les anciennes et les nouvelles têtes, je n'ai pas été déçu de mon 1er meeting depuis 3 ans ! Il ne manquait plus que des filles.

Ramlaid : Merci Eliot ! Encore une fois, la qualité normande est au rendez-vous ! Pleins de petits nouveaux sont arrivés à ce meeting et c'est ma foi fort sympathique. Promis, je m'entraîne à LAP pour la prochaine fois !

Mig : Comme d'hab, ce fut un meeting grand comme un overscan (pas frapper Iron, pas frapper). Merci aux organisateurs et à tous les participants pour l'ambiance. Promis, la prochaine fois je me mets au code.

Plissken : Malgré un budget très limité, je suis quand même venu. Ah, CPC, quand tu nous tiens !

Super Sylvestre : Winape est le meilleur émulateur, OCP le meilleur logiciel de retouche sur CPC, Quasar CPC le meilleur journal technique CPC, et Amstrad Expo le meilleur meeting normand depuis que j'ai re-découvert le CPC. Merci à tous d'être venus !

Dada : Génial ! Que de participants, que de prods de qualité !!! Et une organisation impeccable, comme d'habitude !

Sid : je crois que tout le monde l'a déjà dit mais incroyable organisation, merci Eliot pour ce superbe meeting. Je me souviendrai de ces repas ;-) Sincères remerciements aussi aux vétérans pour leur patience pour des gars comme nous. On n'a pas arrêté de les déranger avec nos questions, bon à part Iron... il mord trop fort. Tchoo et à la prochaine.

Pote Iron : On nous fait croire que Quasar va sortir... INTOX !!!! En tout cas, excellent cru. L'occasion une fois de plus de se retrouver pour échanger nos idées, nos rires et nos émotions. Deux, trois têtes de cons présentes, sans doute les nouvelles têtes ? De grands espoirs de futurs grands codeurs (Flash Solaar, Kris). Enfin, le plaisir toujours présent de faire vivre nos bonnes vieilles machines !!! En espérant vous revoir bientôt, je vous embrasse tous XXX.

Master : L'Amstrad Expo depuis l'an 2000 est le rendez-vous incontournable chaque année ! En plus, cette année il y avait un maximum de personnes, voir de nouvelles têtes fait du bien à la sienne. Il est agréable de voir qu'après toutes ces années le CPC vit toujours.

Slyder : 2005 restera dans les annales, la plus grosse Amstrad Expo jusque là : plein de participants dont beaucoup de nouveaux "d'internet". Le concours a lui aussi eu beaucoup de succès malgré un thème peu inspirant : les compos sont de bonne qualité ; un gros effort général a été fait. Comme à chaque fois, c'est la larme à l'oeil qu'on finit de remettre en ordre la salle vide. Vivement l'année prochaine !!

Flash Solaar : Très bonne ambiance, participants sympas et prêts à aider les débutants, organisation excellente (merci Eliot) : j'achèterai les droits et j'en ferai un film. Ce sera le succès de l'année ! Merci à tous et à bientôt.

Emergency : C'est ma 1ère Amstrad Expo, j'ai loupé 2003 et 2004 pour des raisons... enfin y'a pas d'excuses. Alors je suis happy, j'ai récupéré OCP et Starkos et je compte bien me mettre à l'assembleur. Au prochain meeting 2006, je viendrai avec des zics pour faire les concours. Merci pour tout !

Eliot : Super meeting ! J'ai hâte d'aller au prochain...

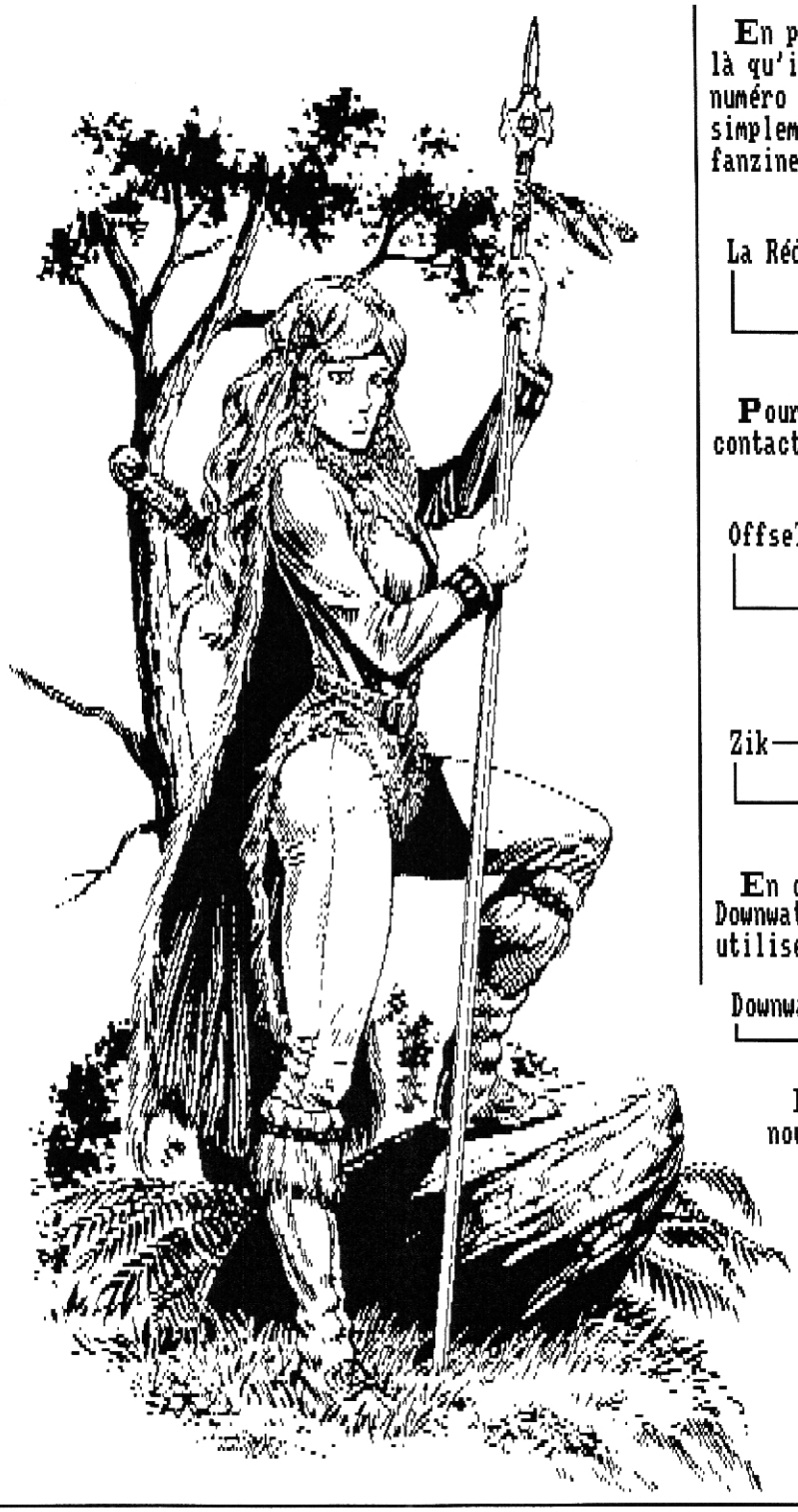
Targhan : Bon, je vais éviter les formules habituelles. Je ne viendrais pas en meeting si j'aimais pas ça ! Bref, encore un coup de maître d'Eliot, et merci pour le cadeau :)

Ram7 : Encore un meeting qui ne faillit pas à sa réputation... Vivement les prochains !

J'veux une om'lette aux œufs !
Pas vous ? Ah bon.

LA RUBRIQUE X

Et voici donc la rituelle Rubrique X dans laquelle vous allez pouvoir trouver les coordonnées de toutes les personnes ayant participé à l'élaboration de ce numéro 21 !



En premier lieu, voici l'adresse de la rédac'. C'est là qu'il faut nous écrire pour commander les anciens numéro de Quasar CPC, demander les Quasar Disc ou tout simplement pour obtenir des renseignements sur le fanzine :

La Rédac' — RIMAURO Philippe & Gilles
8, chemin des Maillos
09200 SAINT-GIRONS
quasarcpc@cpcscene.com
futurs.cpcscene.com

Pour avoir des détails sur certains articles, contactez plutôt directement les rédacteurs :

Offset — RIMAURO Philippe
5, avenue Saint Donatien
06600 ANTIBES
offset@cpcscene.com

Zik — RIMAURO Gilles
8, avenue Marcel Langer
31400 TOULOUSE
zik@cpcscene.com

En complément, voici également les coordonnées de Downwater qui a dessiné quelques-unes des illustrations utilisées dans ce numéro (pages 12, 15, 18/19 et 29) :

Downwater — AITTAHAR Mounir
downwater@cpcscene.com

Pour conclure, n'oublions pas de citer Grim qui nous a bien dépanné en urgence en remplaçant un Barjack "tocardisant" pour réaliser les nouveaux pieds de page :

Grim — GAUZENTES Matthieu
grim@cpcscene.com
www.cpcscene.com

Un autre moyen de contacter tout le monde :

IRC — Serveur : irc.neoxys.org
Canal : #CPC

Histoires Perpendiculaires

Tel un chien à la fidélité imbécile, ce matin, il est revenu, toujours aussi lumineux, toujours aussi lisse. Rares sont les rebelles qui parviennent à animer ce champ immaculé qui s'impose au fil des jours. Souvent, baissant les yeux, on se contente alors du jeu des ombres, un régal pour l'imaginaire, mais ce qui amuserait un enfant pendant des heures nous lassera rapidement.

Certes, la nuit tombée, il parvient à nous charmer. Nous invitant à une grande soirée de gala pleine de paillettes, il nous fera oublier un temps combien il est triste et insipide. Nous relèverons la tête. Mais la concurrence de la ville, tout aussi clinquante et vaillante, est sans concession. Elle l'affadira et aura tôt fait de nous distraire et de nous détourner de lui. Enfin, l'exaltation sera de courte durée lorsque, au jour venu, tout aura disparu. Mais lui, assidu, sera toujours là sans personne pour le remettre en question.

En de rares occasions, la résistance parvient à s'organiser et, l'espace de quelques heures, une armée de moutons parvient à dicter sa loi. En ces instants rares et précieux, une débauche de merveilles éthérées prend vie. Celui qui s'affiche d'ordinaire si morne passe alors au second plan et devient un cadre qui nous offre une myriade de formes exotiques. Changeant et s'entremêlant sans cesse, elles satisfont tout à la fois nos yeux et notre esprit curieux qui travaille à donner un sens à toutes ces aspérités.

Malheureusement, notre regard pétillant voit parfois ce spectacle si rare se transformer en cauchemar. Lorsque les ovins cotoneux parviennent à leur tour à prendre l'ascendant, la lumière fait place à une pénombre donc l'homogénéité n'a rien à lui envier. De plus, ils sont oppressants, souvent vindicatifs, et c'est la douche froide assurée alors que nous regrettons lâchement notre chien fidèle pourtant si apathique.

Néanmoins, à la faveur de chaudes journées d'été, nos révoltés font parfois preuve de créativité. Ils nous invitent alors à des spectacles éblouissants dignes des plus grands concerts de Jean-Michel Jarre. Mais ces féeries fugaces annoncent bien souvent le retour prochain de notre fidèle néant, toujours aussi absolu, toujours aussi décevant, et laisse un goût d'inachevé.

Voilà que le petit blond à lunettes s'écrie : "Quel temps splendide aujourd'hui !"... Mais parlons-nous vraiment de météo ?

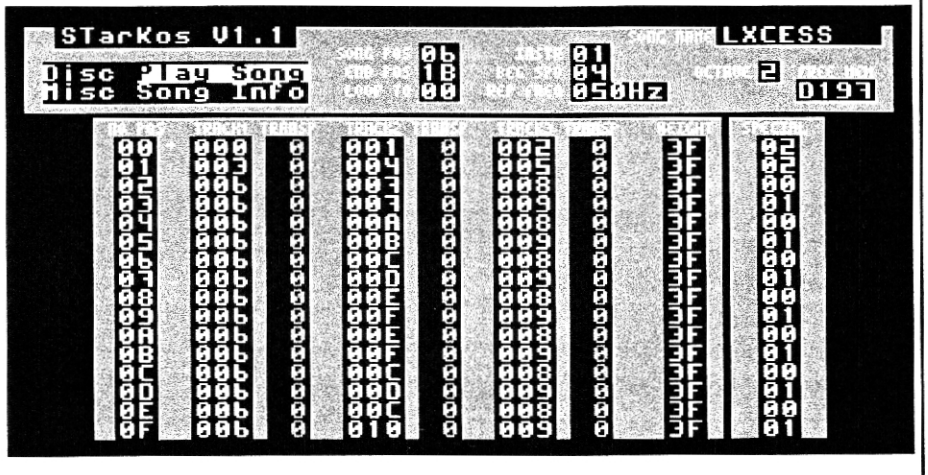
TESTS

Vous avez peut-être reconnu la capture d'écran de cette page, il s'agit de Starkos 1.1, le logiciel de musique créé par le prolifique Targhan.

A sa sortie en Janvier 2004, Starkos s'est positionné comme le nouveau tracker soundchip de référence sur CPC en remplacement du... du... Soundtrakker de BSC (oui bravo je l'ai entendu dans le fond de la salle, il y en a qui suivent). Et à priori, il a tout pour avec en prime une suite logicielle très fournie.

Starkos bénéficie également d'une notice assez bien faite, disponible en français et en anglais. On peut la trouver parmi les fichiers du Starkos Kit et aussi sur le site dédié à l'utilitaire (starkos.cpcscene.com).

Dans cet article, je vais tenter de présenter Starkos en mettant en avant les nouveautés qu'il apporte par rapport au Sountrakker de BSC, puisque ce dernier est je pense bien connu de tous ceux qui font de la musique sur CPC. C'est d'ailleurs le cas de Targhan qui, fort de son expérience double de musicien et codeur, en a profité pour combler certains manques qui existaient dans le Soundtrakker.



tous en piste !

Les habitués des trackers tels qu'on les voit généralement vont devoir changer un peu leur façon de travailler. Dans Starkos on ne place pas ses notes dans une bonne grosse pattern à trois canaux. Au lieu de ça, on dispose de pistes (tracks en anglais) qui ne sont pas associées à un canal audio à priori. Ce n'est que dans la description du séquencement de la musique (cf. la capture d'écran de cette page) que l'on va distribuer les pistes sur les canaux voulus. En mettant trois pistes côte-à-côte, on construit notre pattern. On peut donc réutiliser une piste autant de fois qu'on le souhaite ; il est aussi possible de jouer la même piste sur plusieurs canaux en même temps.

Cette façon de faire a l'avantage de consommer moins de mémoire, ce qui est un paramètre important lorsque l'on veut intégrer sa musique dans un jeu, une démo... C'est une version moins extrême que ce que propose le logiciel de musique Advanced Music Creator (testé dans Quasar CPC numéro 12). Ici, pour chaque pas (marqué n° Pos) dans la séquence de la musique, on donne les numéros des trois pistes à jouer et une longueur de pattern. Ainsi, si une des pistes est plus longue, elle sera tronquée au moment de la jouer (pour cette position de la musique) ; si elle est plus courte elle sera complétée par du vide. Donc, tout va bien, on peut écouter sa musique en regardant défiler les patterns sans être perdu.

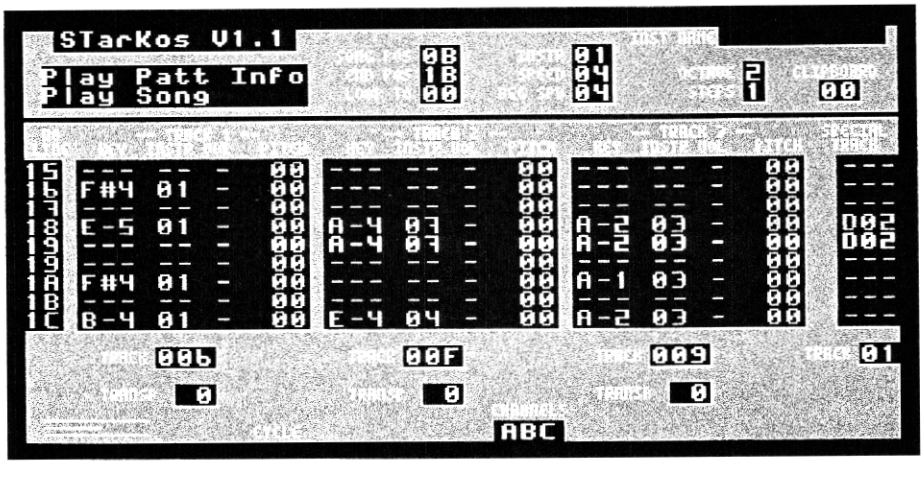
Si vous aimez les chiffres, en voilà quelques-uns pour vous rassurer sur les capacités de Starkos. Il peut gérer 512 pistes et chacune peut faire jusqu'à 127 pas. Et l'on peut définir jusqu'à 256 pas de séquence.

Les fins observateurs que vous êtes ont tout de suite remarqué que l'on précise aussi dans la dernière colonne le numéro de "special track". Je me dois de vous préciser de quoi il retourne ! Une piste spéciale permet d'indiquer des changements de tempo de la musique mais aussi de placer des digidrums. Il s'agit en fait d'envoyer un signal à un moment choisi de la musique à une routine qui se chargera de jouer le son échantillonné spécifié (ou faire autre chose, pourquoi pas). Pour ne rien vous cacher, on peut aussi placer un sample dans les pistes normales mais ce n'est pas tout à fait équivalent, je vous laisse regarder la notice pour les détails.

La chose à noter c'est que le support des samples est intégré, on n'a pas besoin de trouver des stratagèmes comme c'était le cas avec le Soundtrakker pour ajouter de beaux digidrums (ou pas) à sa belle musique (ou pas). Le kit inclut un exemple de musique avec digidrum.

Un éditeur brillant

Maintenant que vous savez agencer des pistes comme bon vous semble, passons à l'édition de leur contenu. La capture d'écran de cette page vous montre comment ça se présente. On retrouve logiquement les trois pistes normales ainsi que la piste spéciale choisies dans l'écran précédent pour le pas courant de la musique (Song Pos). Si vous avez en tête le format des patterns du Soundtrakker de BSC, celles-ci doivent vous interpeller un brin. Car là précisément se situe une autre différence majeure de Starkos. L'approche est différente des trackers conventionnels : au lieu d'avoir pour construire sa musique quelques sons relativement simples que l'on va placer dans la pattern et travailler avec tout un tas d'options, ici on fait l'inverse. Les options disponibles dans une piste sont réduites au strict minimum, on peut modifier le volume (colonne "vol") et la hauteur du son (colonne "pitch"). Mais en contrepartie on peut créer des sons (des instruments) très complexes et en grand nombre. Il est possible de



créer 255 instruments différents.

Pour illustrer ceci, imaginez que vous voulez faire un son qui produise un accord arpégé (arpeggio) répété. Dans le Soundtrakker on utilise un son pur et on lui applique une option dans la pattern qui précise l'accord voulu. Dans Starkos l'arpège sera dans l'instrument lui-même et rien n'apparaîtra dans la piste. Si maintenant vous voulez changer d'accord, il faut changer les paramètres de l'option dans le Soundtrakker alors qu'il faut créer un nouvel instrument dans Starkos. Bien évidemment il est possible de copier des instruments très facilement, pas de panique.

Donc là aussi quelques habitudes sont à changer. Je vous parlerai plus de l'éditeur d'instruments dans le chapitre suivant.

La saisie des notes se fait de manière classique, le clavier du CPC se change pour l'occasion en une sorte de clavier de piano (si on veut). On peut éditer les pistes en parallèle et des raccourcis clavier que l'on a tout intérêt à connaître facilitent beaucoup les choses. Notamment, on peut changer le numéro de piste dans la pattern directement sans repasser par l'éditeur de séquence de la musique.

Si vous avez suivi, vous savez que toutes les pistes n'ont pas nécessairement la même longueur. Dans la représentation de la pattern, par défaut, seul le nombre de pas précisé dans l'éditeur de séquence est visible. Mais un raccourci clavier permet d'activer le mode "beyond" afin d'éditer la piste dans sa totalité. Même la partie qui ne sera pas jouée dans cette pattern devient accessible.

Monte Le son !

Et voici le clou du spectacle, l'éditeur d'instruments. Là on sent que Targhan s'est fait plaisir. Pour ceux qui cherchent la capture d'écran, il faut regarder au dos de cette page. L'instrument représenté ne tire pas parti de toutes les possibilités offertes (loin de là), mais il illustre mon exemple de tout à l'heure sur le cas de l'arpeggio.

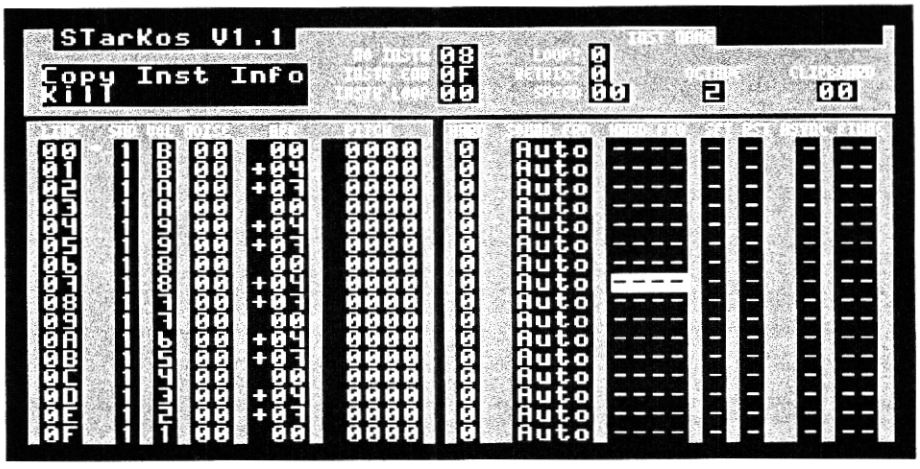
Je ne vais pas faire la description des différentes colonnes, la notice du logiciel fait ça très bien. Mais je vais relever quelques points marquants.

Le nombre ne lignes visibles n'est qu'une infime partie du maximum possible qui est de 256. L'instrument fait la longueur voulue choisie dans cette plage.

Un paramètre SPEED permet de choisir la vitesse d'exécution du son, ça me plaît.

Les possibilités offertes sur les sons hards sont sans commune mesure avec le Soundtrakker. Pour chaque pas de l'instrument, on peut indiquer si un son hard est voulu ou pas et avec quelle courbe, à quelle fréquence... Si l'on sélectionne "Auto" comme fréquence, le logiciel la calcule de façon à ce qu'elle s'approche au plus près de la note jouée à un coefficient près choisi (colonne SFT pour "shift"). C'est très pratique, souvenez-vous que dans le Soundtrakker vous étiez obligé de trouver la bonne valeur pour chaque note dans la pattern (avec des bonnes surprises parfois cependant).

Il est possible aussi dans Starkos de dompter ces sons hards qui ont une tendance certaine à n'en faire qu'à leur tête (je fais allusion aux célèbres vagues, les "mwwwaiii", dixit la notice de Starkos).



Supplément gratuit

Le kit Starkos n'a pas usurpé son appellation de "kit", une ribambelle d'outils accompagne l'éditeur de musiques.

C'est bien beau d'avoir fait une jolie musique avec des sons à faire pâlir un C64 (ça n'engage que vous, je vous trouve bien présomptueux !), mais vous n'allez pas toujours l'écouter dans l'éditeur. Vous souhaitez certainement l'utiliser dans vos productions et pour ça il faut transformer le fichier SKS avec l'outil GenSong. Ca s'appelle compiler la musique et ça vous donne au final un ou plusieurs fichiers qui contiennent les données de la musique. C'est le même principe général que celui du Soundtrakker de BSC sauf qu'ici les données de la musique et l'application qui va jouer la musique (le player de musique) sont stockées dans des fichiers distincts. Le player est générique ; si vous avez besoin d'avoir plusieurs musiques, le même player sera capable de les jouer toutes. D'où un gain de place mémoire important et une plus grande souplesse.

La colonne RST (de même que l'option RETRIG mais pour le début du son seulement), permet de faire repartir l'enveloppe hard du début à volonté. Positionner à 1 la valeur de la colonne HSYNC (pour "Hard Sync") sacrifie la justesse du son normal mélangé au son hard en alignant sa fréquence sur celle (moins précise) du son hard. De cette façon on peut obtenir des sons hards plats ou contrôler la vitesse des battements avec FTUNE (pour "Fine Tune").

Bien entendu, rien n'oblige à avoir un son normal mélangé au son hard (et réciproquement), les colonnes SND et HARD sont là pour ça.

Pour l'édition des courbes de volume et de bruit, vous pouvez si vous le souhaitez utiliser l'éditeur alternatif. Il s'agit d'une représentation graphique des courbes, plus agréable qu'une liste de chiffres. Pour avoir un aperçu de ce que ça donne, jetez un oeil en bas à droite de cette page.

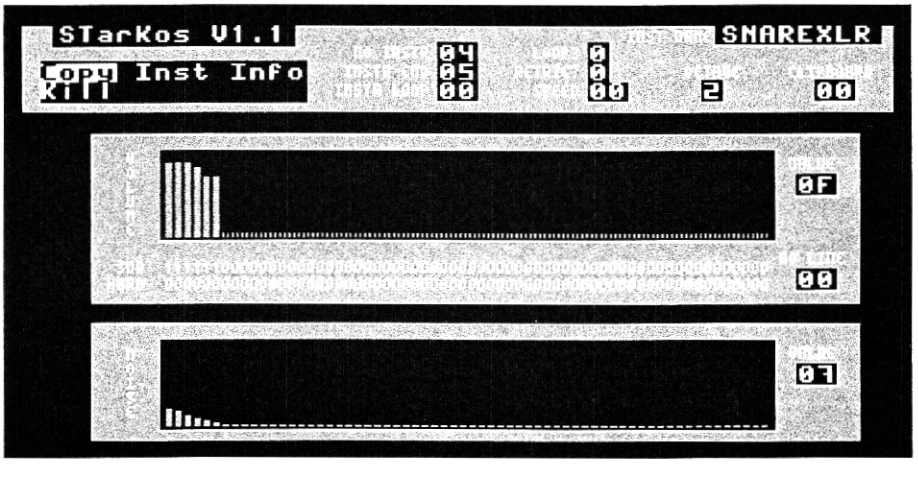
Voilà pour le tour d'horizon de l'éditeur d'instruments. Il y a clairement moyen d'expérimenter beaucoup de choses, avec des résultats plus ou moins heureux à prévoir ! Moi qui avait tendance à créer mes sons de de zéro à chaque nouvelle musique avec le Soundtrakker, je pense qu'avec Starkos on a plutôt intérêt à se constituer sa propre "bank de sons" au fur et à mesure. Le kit Starkos contient plusieurs musiques (fichiers SKS) et instruments (fichiers SKI) de démonstration, ça permet de s'en inspirer et de prendre en main le logiciel assez rapidement. Vous pourrez trouver des SKS de musiques supplémentaires sur le site internet de Starkos dont j'ai donné l'adresse en début d'article.

Si avec tout ça vous n'avez pas envie de vous y mettre et tenter de faire une musique, je ne sais pas ce qu'il vous faut. Et en plus, ce n'est pas tout, voyons la suite...

Ou coup il faut aussi générer le player avec l'outil GenPlayer. Trois types de players sont disponibles : Basic, ASM et Interruption. Je vous renvois à la notice pour la description de leurs spécificités. L'utilisation du player est simple, il faut juste ne pas oublier de lui indiquer où se situe en mémoire les données de la musique à jouer. Notez au passage que le player prend significativement moins de temps machine que celui du Soundtrakker.

Les deux outils suivants sont plus des modules que des outils et ils sont fournis sous forme compilée mais aussi sous forme de code source assembleur. Il s'agit de RELPLAY et RELSONG.

Le premier permet de reloger un player, c'est-à-dire



changer l'adresse à laquelle il était prévu qu'il soit exécuté. C'est un outil très utile dans le cas par exemple d'une demo ou un jeu intégrant plusieurs musiques et avec des contraintes qui font que votre player ne peut pas rester tout le temps à la même adresse. Vous pourrez grâce à cet outil utiliser un seul player pour toutes les musiques même dans un cas pareil.

Le relogeur de musique fait la même opération mais pour les données d'une musique cette fois.

Vous trouverez également dans le kit un convertisseur vers le format YM des musiques Starkos. Ce format, une fois passé dans les mains du kit AYC de Madram (du groupe Overlanders) vous permettra d'utiliser votre musique Starkos dans un contexte où le temps machine est très précieux.

Voilà pour le panorama des outils fournis, Targhan donne en plus dans la notice des informations techniques sur les players : des adresses de données intéressantes, des bidouilles possibles, des indications pour agrémenter vos musiques de digidrums...

Les gros défauts de Starkos

J'imaginais Targhan se ruer sur ce paragraphe à la vue de ce titre et donc je n'ai pas pu m'empêcher !

En réalité je n'ai pas de reproches à faire vis-à-vis de Starkos, juste quelques petites remarques tout au plus. Je vais passer rapidement sur le couplet de plaintes sur les habitudes à changer, les raccourcis clavier qui ne me tombent pas sous les doigts etc. Finalement on s'y fait très bien (et il faut dire aussi que la version 1.1 a apporté beaucoup de nouveautés qui ont amélioré l'ergonomie).

Il faut faire attention à l'optimisation mémoire qui surgit dès qu'on a le dos tourné et qui remanie les instruments en particulier. Gare à bien positionner la longueur avant de quitter l'édition, sinon vous perdrez ce qui dépassait.

Il vaut mieux bien connaître l'hexadécimal avant d'utiliser Starkos car toutes les valeurs numériques sont affichées dans cette base. Je ne pense pas que ça pose problème aux CPCistes qui utilisent Starkos mais c'était à mentionner.

Il est possible de jouer le pas courant de la pattern pendant son édition avec la touche COPY mais les valeurs de la colonne "pitch" ne sont pas prises en compte. Ce n'est pas vraiment gênant, il faut jouer toute la pattern pour entendre tous les effets, il suffit juste de le savoir.

Je regrette un peu l'option "vibrato" du Soundtrakker. Comme c'est un effet d'expression, il aurait sa place dans la piste aux côtés du volume et du pitch. Mais on peut s'en sortir en créant plusieurs sons.

Donc à l'arrivée, il semblerait qu'en gros, Starkos a bon sur toute la ligne ! J'apprécie certains détails comme la possiblité de donner le nom de l'auteur et un commentaire dans le fichier SKS, le vumètre qui indique les sons hards... Et je m'aperçois que j'ai oublié de vous citer une fonctionnalité importante : la fréquence à laquelle se joue la musique n'est pas figée à 50Hz mais elle peut être choisie parmi les valeurs 13, 25, 50, 100, 150 et 300Hz. Encore un peu plus de flexibilité donc.

Pour finir !

Vous l'avez compris, le groupe Arkos a encore frappé ! Targhan a réussi à faire une version ergonomique d'Advanced Music Creator ! Plus sérieusement, j'ai une bonne idée du travail que représente la création d'un tel outil et il l'a réalisé de façon remarquable en un temps record (là je m'attends à quelques railleries me concernant !). De plus, il a pris en compte les recommendations et critiques des CPCistes de sorte que, je pense, tout le monde est satisfait du résultat. Personnellement, je continuerai d'utiliser le Soundtrakker de BSC (New Age Software) en parallèle, au moins parce que mes anciennes musiques inachevées sont sous ce format (et qu'il y en a légion).

Si vous avez des idées d'améliorations à apporter à Starkos, le connaissant, Targhan est à l'écoute et il se fera un plaisir de sortir une version 1.2 si ça se justifie (héhé). Ne lui demandez pas de faire une version en ROM de Starkos, c'est déjà fait (en deux ROMs pour l'éditeur et une troisième pour les outils GenPlayer et GenSong) ! Un bel utilitaire comme ça, il n'y a plus qu'à l'utiliser, j'espère vous en avoir donné l'envie. A bientôt !

Zik

HARDWARE

Assembleur

Voici donc, après une longue attente, la suite et fin du dossier sur la Multiface Two. Dans le numéro 20, nous avions fait le tour des fonctionnalités de la Multiface Two et v... "à quelques détails près !" vient de me lancer le petit blond à lunettes avec arrogance. Effectivement, pour une fois il n'a pas tort (mais il se prend tout de même une gifle pour le ton sur lequel il m'a parlé) ; nous allons donc commencer par ce dernier point fonctionnel laissé en suspens dans le numéro précédent.

A quoi bon sauver AF après un LD A,I ?

Oui, rappelez-vous, je vous avais expliqué le détail du stockage des registres par la MF2 à ceci près que celle-ci sauve également l'état de AF après avoir récupéré les valeurs des registres I et R... une petite révision sur le fonctionnement du Z80 s'impose. Commençons par le plus simple. En premier lieu, le programme de la MF2 sauve le contenu du registre I dans A, puis stocke AF dans sa RAM dans l'espace des registres Z80 ; F se retrouve en &3FF8 (nommé Mi dans le menu de la MF2) et A se retrouve en &3FF9 (nommé I, logique, dans le menu de la MF2). Il est important de conserver ainsi le contenu du registre I pour pouvoir le restaurer lorsque la MF2 rendra la main au programme car il sert en mode vectorisé (IM 2) pour connaître l'octet de poids fort de l'adresse de la table des vecteurs d'interruption (pour plus de détails voir l'article dédié aux interruptions dans notre numéro 18). Mais en même temps, notre chère MF2 sauve le contenu de F... "c'est naze, ça ne sert à rien, encore un programme pas optimisé !" s'indigne le petit blond à lunettes. Mais non ! Pas du tout ! La valeur de F après un LD A,I (ou LD A,R) est primordiale ! Elle nous permet de connaître l'état du flag IFF2.

Mais qu'est-ce donc que ce flag IFF2 et à quoi sert-il ? En fait, le Z80 dispose de deux flags pour la gestion des interruptions : IFF1 et IFF2. Ces deux flags sont positionnés simultanément à 1 lors d'EI, et à 0 lors d'un DI. Lorsque IFF1 est à 1, alors le Z80 accepte les interruptions ; lorsqu'il est à 0, il les ignore... voilà qui semble bien obscur ! En fait, pas tant que ça, car lorsqu'une NMI a lieu (ce qui est le cas lorsque la MF2 s'active), IFF1 est mis à 0 de sorte que les interruptions soient désactivées mais IFF2 est laissé inchangé ! Ce qui signifie qu'il vaut 1 si on était en EI lors de la NMI, et 0 si on était en DI... d'où l'intérêt de connaître son état pour la MF2 afin de savoir, au retour, si on devra réactiver les interruptions ou pas. Mais où diable se cache ce flag ? Eh bien, il est positionné dans F, à la place du flag P/V (bit 2) suite à une instruction LD A,I ou LD A,R ! Comme vu plus haut, la MF2 récupère son état à partir du LD A,I ; le LD A,R ne servant qu'à récupérer l'état de R dans ce cas là.

Les Limitations de La Multiface two

Venons-en maintenant aux défauts de conception de cette excellente interface. Eh oui, tout n'est pas parfait dans la MF2 ! Le plus gros soucis vient du fait que contrairement au CPC, la MF2 décode les ports I/O des divers circuits avec rigueur. Je m'explique, sur CPC, chaque circuit est accessible via un port : &7Fxx pour le Gate Array, &BCxx-&BFxx pour le CRTC, &F4xx à &F7xx pour le PPI, etc. Mais il se trouve qu'en pratique, le décodage de ces ports n'est que partiel ; dans un soucis d'économie de portes logiques, seuls quelques bits finement choisis sont utilisés pour décoder et ainsi sélectionner le bon circuit en fonction du port. Vous trouverez ci-dessous un petit tableau illustrant ce décodage. "x" y désigne un bit non décodé, "0" un bit testé à 0, "1" un bit testé à 1 et "r" un bit permettant la sélection d'un registre sur le périphérique (les ports CRTC ou PPI par exemple).

Nom des périphériques (adresse des ports logiques)	bits décodés par le CPC							
	15	14	13	12	11	10	9	8
Gate Array - PENR, INKR (&7Fxx)	0	1	x	x	x	x	x	x
Gate Array - RMR, MMR (&7Fxx)	0	x	x	x	x	x	x	x
CRTC (&BCxx-&BFxx)	x	0	x	x	x	x	r	r
Sélection ROM Haute (&DFxx)	x	x	0	x	x	x	x	x
Imprimante (&EFxx)	x	x	x	0	x	x	x	x
PPI (&F4xx-&F7xx)	x	x	x	x	0	x	r	r
Périphériques d'expansion	x	x	x	x	x	0	r	r

On voit dès lors que le CRTC est aussi bien accessible via &BCxx (10111100) que &ACxx (10101100) pour la sélection du registre (rr=00)... et c'est là que réside le problème de notre MF2 qui, forte de sa rigueur de décodage, n'interceptera que les accès sur le port &BCxx. Bien sûr, accéder au CRTC par le port &ACxx n'est vraiment pas très propre puisqu'on accède simultanément au port imprimante... mais le fait est que ça fonctionne et qui plus est, sans grande conséquence, même si une imprimante était connectée compte tenu de l'absence de gestion du "strobe" qui est indispensable pour valider les données d'impression. En revanche, si une Soundplayer était connectée, des effets secondaires indésirables seraient à prévoir sur le son ou le réseau. D'autres combinaisons d'adressage sont possible et dans certains cas, permettent des optimisations de code on ne peut plus subtiles. En outre, une conséquence directe est qu'il est possible de feinter la MF2 assez facilement. Par exemple, tapez en Basic la ligne suivante :

```
OUT &BC00,0:OUT &AC00,5:OUT &BD00,0
```

En pratique, que fait ce programme ? Eh bien il sélectionne le registre 0 du CRTC via le port légal &BCxx, il sélectionne ensuite le registre 5 (qui est le registre de retard vidéo) tout en mettant également 0 sur le port imprimante. Mais du point de vue de la MF2, les choses sont différentes. Elle va voir une sélection du registre 0, elle ne verra pas celle du registre 5, puis elle verra l'envoi d'une donnée sur le registre... 0 ! Résultat des courses : activez la MF2 puis faites "Return" pour rendre la main : paf, écran noir et CPC planté. "Pourquoi ?", demande le petit blond à lunettes d'un air interloqué. Eh bien tout simplement car au retour, la MF2 va remettre le système dans son état initial... lequel était pour elle un registre 0 à 0 (ce qui fait irrémédiablement "planter" le CRTC) ; eh voilà une MF2 dans les choux !

Bluffer le bluffeur

Enfin, pas tout à fait dans les choux... il est facile, avant de rendre la main au système, de vérifier la liste des valeurs du CRTC (en &3A80) et de les corriger pour que le retour se passe bien. Il en va de même pour la palette des couleurs avec laquelle on peut faire les mêmes blagues. Beaucoup de démos utilisent l'astuce du registre 0 fictivement à 0 (parfois volontairement, parfois simplement en conséquence d'une optimisation d'adressage) alors que des jeux (comme Batman The Movie) ont plutôt bidouillé sur la palette des couleurs (écran noir au retour mais rien de planté, juste toutes les couleurs en noir... ce qui est moins "méchant" mais en réalité, bien plus pénible à remettre en place au niveau de la MF2 avant de rendre la main).

De la même manière, contrairement à ce que j'ai souvent entendu, il est vraiment enfantin d'utiliser la MF2 pour interrompre puis refaire partir une démo pleine de ruptures après modification. Le petit blond à lunettes se fait avoir à tous les coups, il lance sa démo, coupe avec la MF2, patche ce qu'il a à patcher, et rend la main... mais là, une fois sur deux, ça plante ! Pourquoi ? Eh bien tout simplement car selon le moment où le programme a été interrompu, le CRTC contient des valeurs susceptibles de bloquer le fonctionnement du CPC. Il suffit de remettre dans les principaux registres du CRTC des valeurs viables avant de rendre la main et ça se passera bien à tous les coups ! Vous aurez juste un petit saut d'écran au retour, mais plus de plantage. Bien sûr, cette manipulation devient indispensable lorsque le demomaker s'est amusé (par goût ou par besoin d'optimisation) à adresser le CRTC ou le Gate Array par des ports illégaux.

Le bug "imparable"

Outre ce petit problème de décodage des ports qui est souvent utilisé pour gêner l'utilisation de la MF2 (sans jamais l'empêcher toutefois), il existe un bug beaucoup plus sévère sur lequel je suis tombé par hasard alors que je désassemblais sa ROM afin d'en comprendre le fonctionnement intime. Il n'est cette fois-ci pas matériel mais logiciel et concerne la routine de détection de la mémoire. En effet, dans sa ROM, la MF2 détermine si le CPC possède seulement 64ko de RAM ou plus afin de positionner l'octet à l'adresse &3A96 à 0 (64ko) ou à 2 (au moins 128k)... eh bien ce test est buggé et laisse des traces en mémoire ! En utilisant habilement ces traces, il est possible de rendre l'utilisation de la MF2 pratiquement impossible. Ce bug est à ma connaissance présent dans toutes les MF2 distribuées (y compris la MF2+ de 1990) mais j'ai constaté à ma grande surprise que la ROM de MF2 livrée avec l'émulateur CPC CaPriCe32 a été patchée et ne souffre plus de ce bug. Il serait en théorie possible d'utiliser cette version patchée (ou d'en faire une autre soi-même) et de la reprogrammer dans l'EPROM de le MF2... tâche néanmoins plus délicate qu'il n'y paraît vu que la ROM de la MF2 est matériellement cryptée (elle n'est pas câblée de façon linéaire).

Je vois le petit blond à lunettes s'agiter sur sa chaise : "c'est quoi le bug ? c'est quoi le bug ?". J'y viens. Afin de détecter si le CPC a ou non une page de RAM supplémentaire, la MF2 teste la présence de la première bank de celle-ci (bank &C4). Pour ce faire, elle poke des valeurs en &4000 et regarde ce qui se passe lors de la commutation de la bank pour en déduire si une bank se commute effectivement ou pas. Oui mais voilà, dans l'un des cas de ce test, le programme oublie purement et simplement de restaurer la valeur

initialement présente en bank &C4 ! Pour se retrouver dans ce cas de figure, rien de plus simple, il suffit que la valeur &AA soit présente en &4000 bank &C4 ; après une activation de la MF2 cette valeur sera remplacée par un &55. Il est impossible, lorsqu'on est sous la MF2 de savoir si la valeur initiale était &AA ou &55 afin de rectifier le tir avant le retour au programme comme on peut le faire pour les registres du CRTC par exemple. Le petit blond à lunettes se la raconte en disant qu'il n'y a que deux cas possibles : si ça plante au retour avec &55, c'est donc qu'il faut poker &AA avant de rendre la main... certes, mais imaginez un programme anti-multiface un peu plus élaboré qui modifierait régulièrement cette valeur, la passant tantôt à &AA, tantôt à &55 ; tout devient alors bien plus compliqué pour contourner la protection. Oui, rien n'est perdu, l'idéal étant de neutraliser le test lui-même, mais que de complication pour un petit bug logiciel !

Les multiples facettes ont été mise à jour ?

Voilà, nous avons plus ou moins fait le tour des fonctionnalités (numéro 20) et des défauts (juste au dessus) d'une des plus célèbres interfaces sur CPC. Bien sûr, nous pourrions pousser encore plus en avant sur le fonctionnement intime de la MF2 ou sur d'autres petits défauts qui permettraient de la déceler ou d'embêter ses utilisateurs, mais je pense que l'essentiel (et même plus) a été dit et qu'il est plus intéressant de s'attaquer à sa programmation avec un petit exemple simple mais qui met en œuvre les principaux mécanismes qu'il convient de maîtriser.

Programmons La MF2

Vous trouverez à la fin de cet article un petit programme tout simple qui ne fait rien de plus que d'installer dans la MF2 un outil qui permet de copier les 64ko de mémoire centrale dans les 64ko de banks d'un CPC6128 (ou d'un CPC464/664 étendu) lorsque vous appuyez sur le bouton STOP. A la lumière des informations fournies dans notre numéro 20, la procédure d'installation dans la MF2 devrait vous paraître assez simple. Il s'agit simplement, après avoir vérifié sommairement la présence d'une MF2 non masquée en tentant de lire un octet de sa ROM, de copier le programme à intégrer dans la MF2 en &2000 (la plage de RAM de la MF2 qui avait été largement décrite dans l'article précédent).

Vient ensuite le programme dédié à la MF2 lui-même. Celui-ci s'exécute dans la mémoire de la MF2 et possède un header. La directive de compilation ORG &2000,&8100 permet d'assembler le code en &8100 (son adresse de stockage) pour une exécution en &2000 (là où il est mis dans la RAM de la MF2 par le code d'installation logé en &8000). Le header contient les informations suivantes :

- l'adresse d'exécution du programme (label MF2Code),
- la configuration Gate Array pour les ROMs (voir l'article sur le Gate Array dans le numéro 19),
- la configuration Gate Array pour la RAM (voir aussi le numéro 19 de Quasar CPC),
- un octet vierge (inutile à la MF2)
- le mot-clef "RUN" pour indiquer le mode Direct Jump

Si vous regardez tout ça à la lumière de l'article du numéro précédent tout devrait vous paraître limpide (n'en déplaise au petit blond à lunettes). A la suite de ce header nous avons donc le bout de code qui sera invoqué par la MF2 lorsque vous presserez le bouton STOP. Ce programme va donc copier une à une les 4 pages de 16ko de RAM centrale sur les 4 pages de 16ko des banks après avoir fait un flash rouge pour indiquer le bon fonctionnement.

Copie dans Les banks

Le programme commence par copier la page en &4000 dans la bank &C7 (la quatrième page de 16ko des banks). Pour ce faire, il place &C1 sur le Gate Array ; ceci a pour effet de commuter la bank &C7 en &C000. Il effectue ensuite simplement une copie de &4000 vers &C000. Le programme place ensuite &C2 sur le Gate Array, dans ce mode les 64k de bank sont commutés sur la plage d'adressage du Z80 (comme notre programme s'exécute dans la RAM de la MF2 ça ne pose aucun problème). Dès lors, il peut copier ce qui se trouve en &C000 (la bank &C7) en &4000 (la bank &C5) : nous avons copié la deuxième page de 16ko de la RAM centrale sur la deuxième page de 16ko de banks, mission accomplie nous pouvons passer à la page suivante.

On envoie ensuite &C7 au Gate Array et on copie de &C000 à &4000 ; voilà la quatrième page de 16k de RAM centrale copiée là où il faut en bank. On trouve ensuite un nouveau petit bout de code qui a pour mission de gérer le flash sur le border puis on attaque la copie de la page en &8000 en bank &C6. Jusque là, tout était simple et chaque page est à sa place. Il ne nous reste plus qu'à copier la page située en &0000 dans la bank &C4 mais c'est un peu plus compliqué vu que notre programme s'exécute dans la RAM de la MF2 qui est elle-même commutée dans ces adresses ! Le programme va donc devoir jongler un petit peu. Tout d'abord on commute la bank &C4 dont nous aurons de toute façon besoin en destination de copie. Ensuite, on sauve le contenu de 100 octets de RAM situés en &8000 dans la RAM de la MF2 avant d'y placer notre petit bout de code personnel (label SwapCode) puis de l'appeler (JP &8000).

Ce petit bout de code est la seule difficulté du programme. Premièrement il déconnecte la page de la MF2 située en &0000-&3FFF ; pour ce faire il faut

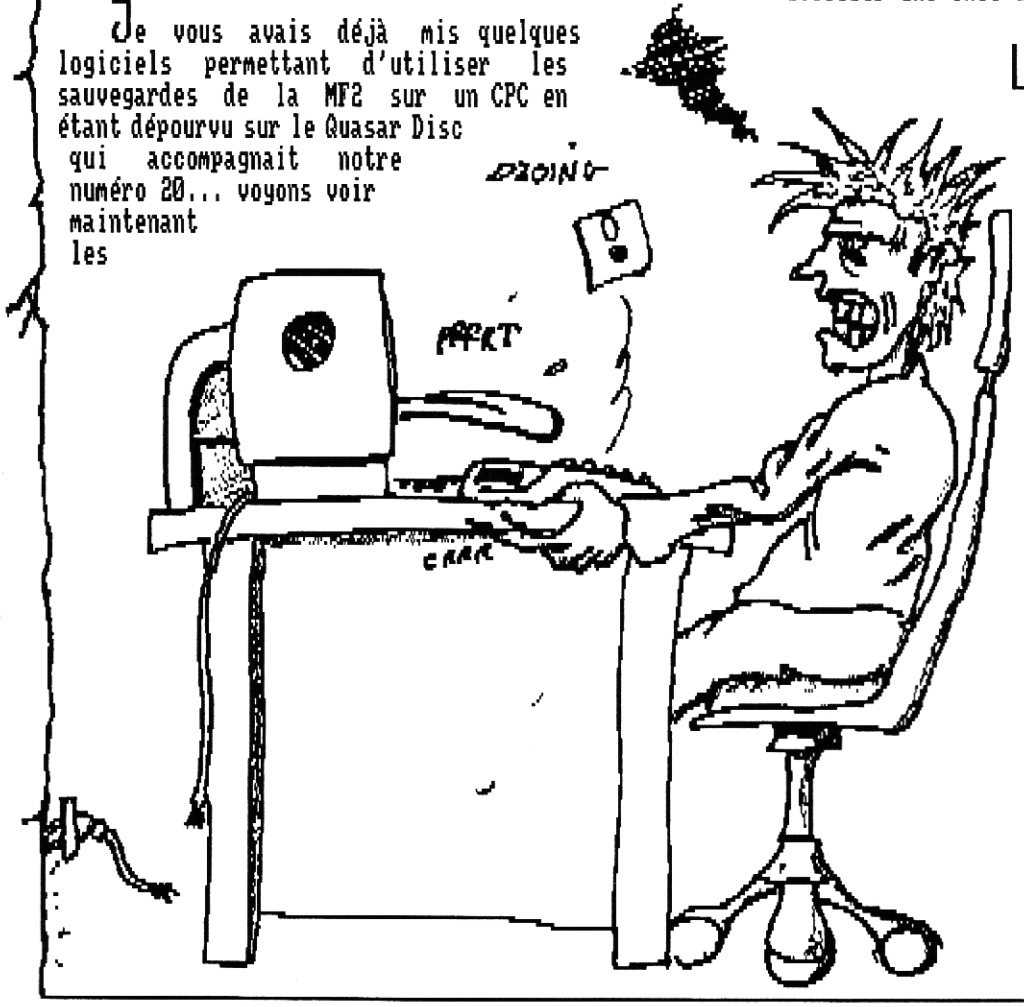
invalider la page ROM/RAM de la MF2 elle-même grâce au port &FEEA puis déconnecter toute ROM basse via le Gate Array (la MF2 est une ROM Basse). Dès lors, on a accès à la RAM centrale située entre &0000 et &3FFF que l'on peut copier simplement en &4000 où nous avions préalablement commuté la bank &C4 (la première page de 16ko des banks). Enfin, on recommute la ROM basse et on active la page ROM/RAM de la MF2 avant de retourner à notre code situé dans celle-ci. Le programme remet le contenu originel de la zone en &8000 dont nous avons eu besoin avant de quitter.

Et voilà, ce qui était à faire a été fait. On a à la sortie de ce programme la copie parfaite de la RAM centrale dans les banks ; couplé avec le Hacker ce petit outil peut être bien pratique. Bien sûr, il ne s'agit pas là d'un programme très complexe et il est possible de faire bien mieux compte tenu des 6ko de mémoire vive disponibles dans la MF2... d'autant que le programme stocké dans cette mémoire peut faire appel à du code stocké sur disquette ou même dans une ROM d'extension. Toutefois, il offre un bon exemple puisqu'il jongle avec les pages mémoire, chose qu'il faut maîtriser à la perfection pour pouvoir faire un logiciel adapté à la MF2. D'ailleurs...

Quels softs pour la MF2 ?

Je vous avais déjà mis quelques logiciels permettant d'utiliser les sauvegardes de la MF2 sur un CPC en étant dépourvu sur le Quasar Disc qui accompagnait notre numéro 20... voyons voir maintenant les programmes qui customisent cette interface. Je vous en ai déjà parlé brièvement la dernière fois, le plus célèbre d'entre eux est The Insider ; développé par les concepteurs de la MF2 en personne. Il intègre tout un tas d'outils pratiques dont un désassembleur ! Ce programme étant relativement complexe, il a besoin d'avoir sa disquette toujours présente dans le lecteur interne de votre CPC afin d'accéder à ses divers modules vu que tout ne rentrait pas dans les 6ko de mémoire libre de la MF2. Un autre défaut de ce logiciel est son manque certain d'ergonomie et son interface lilliputienne.

Il existe un autre programme similaire à The Insider qui dispose d'une interface plus agréable et n'a pas besoin de faire des accès disque à tout bout de champ pour fonctionner : Tearaway. Ce programme de 1991 (The Insider date de 1989) développé par CPC Network offre globalement les mêmes outils que ce dernier avec un confort d'utilisation amélioré... toutefois, il est moins puissant puisqu'il ne fonctionne que sur un CPC doté d'au moins 128ko et n'est pas compatible avec les logiciels exploitant les banks puisqu'il les utilise lui-même pour fonctionner ! En fait, c'est grâce à ce stratagème qu'il peut offrir une interface digne de ce nom et éviter tout accès disque. A quand un logiciel de ce type s'appuyant sur une ROM d'extension ? Nous pourrions alors avoir des outils très complets sans utiliser une once de mémoire vive.



Le mot de la fin

J'espère que les informations que j'ai pu vous fournir dans ces deux derniers numéros de Quasar CPC vous donneront envie de développer des logiciels élaborés pour la Multiface Two. Si vous avez des questions sur des points précis, n'hésitez pas à me contacter !

OffseT

```
; Copy 64k v1.0
; Programme pour Multiface Two
; OffseT of Futurs' - 02/2003
;
; Installe un programme
; en mode Direct Jump
; qui copie les 64k de
; RAM centrale dans les
; premiers 64k de bank
;

        Org &8000       ; Implantation
        Limit &80ff     ; programme
        Nolist          ; d'installation
```

```
        ld hl,msgintro  ; Texte d'intro
        call afftxt

        di              ; Interdiction interruptions
        ld bc,&7f8a
        out (c),c       ; Commutation ROM Basse
        ld bc,&fee8
        out (c),c       ; Commutation page MF2
        ld a,(2)        ; Test MF2 présente
        cp &f
        jr z,erreur

        ld hl,&8100
        ld de,&2000
        ld bc,&200
        ldir            ; Installation de Direct Jump

        ld bc,&feea
        out (c),c       ; Déconnexion page MF2
        ei              ; Autorisation interruptions

        ld hl,msgsucces ; Texte de réussite
AffTxt  ld a,(hl)
        or a
        ret z
        call &bb5a
        inc hl
        jr afftxt
Erreur  ld hl,msgerr
        jr afftxt

MSGIntro
        db "Copy 64k 1.0",13,10
        db "OffseT of Futurs' - 02/2003",13,10,0
MSGSucces
        db "Programme installé avec succès,",13,10
        db "Multiface Two prete.",13,10,0
MSGErr  db "Multiface Two non trouvée !",7,13,10,0

;

        Org &2000,&8100 ; Implantation code
        Limit &9700     ; Direct Jump

MF2Head dw mf2code      ; Header du code Direct Jump
        db &89          ; Config ROM
        db &c0          ; Confin RAM
        db 0            ; N/A
        db "RUN"        ; config jump

MF2Code ld bc,&7f10     ; Sélection border
        out (c),c
        ld c,&4c        ; Border rouge
        out (c),c

        ld c,&c1        ; Copie de la page &4000
        out (c),c       ; vers la bank &C7
        ld hl,&4000
        ld de,&c000
        ld bc,&4000
        ldir
        ld bc,&7fc2     ; Copie de la bank &C7
        out (c),c       ; vers la bank &C5
        ld hl,&c000
        ld de,&4000
        ld bc,&4000
        ldir
        ld bc,&7fc7     ; Copie de la page &C000
        out (c),c       ; vers la bank &C7
        ld hl,&c000
        ld de,&4000
        ld bc,&4000
        ldir

        ld bc,&7f4b     ; Border blanc
        out (c),c

        ld c,&c6        ; Copie de la page &8000
        out (c),c       ; vers la bank &C6
        ld hl,&8000
        ld de,&4000
        ld bc,&4000
        ldir

        ld bc,&7fc4     ; Copie de la page &0000
        out (c),c       ; vers la bank &C4
        ld hl,&8000
        ld de,buffer
        ld bc,100
        ldir
        ld hl,swapcode
        ld de,&8000
        ld bc,100
        ldir
        jp &8000
RetourSwapCode
        ld hl,buffer
        ld de,&8000
        ld bc,100
        ldir
        ret             ; Retour MF2

SwapCode                ; Code de copie page &0000
        ld bc,&feea     ; Déconnexion page MF2
        out (c),c
        ld bc,&7f8d     ; Déconnexion ROM basse
        out (c),c
        ld hl,&0000     ; Copie de la page &0000
        ld de,&4000
        ld bc,&4000
        ldir
        ld bc,&7f89     ; Connexion ROM basse
        out (c),c
        ld bc,&fee8     ; Connexion page MF2
        out (c),c
        jp retourswapcode

Buffer  ds 100          ; Buffer swap
```

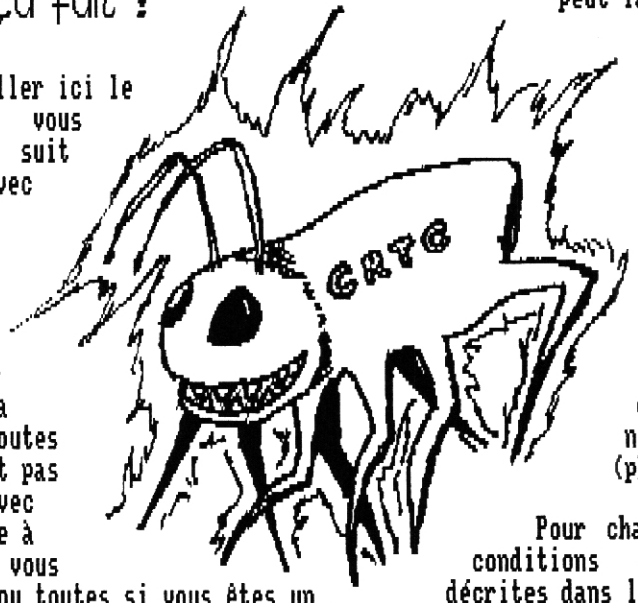
CODING

Assembleur

Il y a quelque temps de cela, Targhan, un célèbre acteur de la scène PC, m'avait demandé de lui fournir un test CRTC "neutre" pour la non moins célèbre Demo Iz Art. Ce test CRTC devait avoir pour particularité d'identifier le CPC sans modifier les registres du CRTC (pas de saut d'écran, etc.). L'exercice étant assez ludique, après avoir réalisé ce petit programme afin que notre barbare puisse l'intégrer à la DIA, j'ai continué d'enrichir ce test d'une multitude de méthodes de détection diverses et variées. Ce petit programme regroupant au final pas mal de petites choses fort sympathiques, nous avons décidé de le publier dans nos pages afin que tout le monde puisse en profiter.

Qu'est-ce que ça fait ?

Je ne vais pas vous détailler ici le fonctionnement du code car vous constaterez que le listing qui suit est largement commenté et qu'avec un minimum d'efforts vous parviendrez à le comprendre. Sachez juste que le test utilise plusieurs routines lancées consécutivement, chacune permettant d'identifier tel ou tel type de CRTC. Il y a de nombreuses redondances et toutes les routines ne sont absolument pas indispensables pour détecter avec certitude un type de CPC. Libre à vous d'utiliser celles qui vous paraîssent les plus élégantes ou toutes si vous êtes un peu paranoïaque.



Notez également qu'il s'agit d'un test destiné à être lisible et il n'est pas particulièrement optimisé ; toutefois, il fait ce qu'il faut comme il faut. Libre à vous là aussi de maquiller ou même d'intégrer carrément les routines à votre code plutôt que d'avoir un test "à part". Absolument aucun des tests publiés ici ne modifie le CRTC ou le Gate Array ; de ce fait, l'affichage n'est pas altéré durant la détection. Seul l'état de configuration du PPI est altéré par quelques tests. Le résultat cumulé des divers tests permet non seulement d'identifier avec certitude un CRTC mais aussi de disqualifier tout CPC ayant donné des résultats incohérents d'un test à l'autre (ce qui sera le cas des CPC en panne, des CRTC non connus à ce jour et accessoirement, des émulateurs).

Qu'est-ce qu'on teste ?

Nous avons plusieurs types de tests. En premier lieu, nous avons ceux qui testent les lectures sur les registres CRTC. Plusieurs registres, selon qu'ils soient accessibles en lecture ou pas, vont nous permettre de différencier divers CRTC (lecture de l'offset sur les reg12 et 13, lecture de la dernière ligne de VBL sur le reg10, lecture du bloc en cours de balayage sur le reg11). On peut également vérifier comment se comporte la lecture sur les registres qui sont censés ne rien renvoyer. Les valeurs renvoyées d'un type de CRTC à un autre étant différentes, on peut là aussi en tirer des conclusions intéressantes. Enfin, on va lire le port de statut du CRTC (&BExx) qui nous permet par exemple de détecter le border uniquement sur CRTC 1.

En complément de ces tests directement liés au CRTC, nous avons quelques tests qui scrutent le fonctionnement du PPI. En effet, sur CPC+, le PPI est émulé et a quelques différences de fonctionnement notables par rapport à son modèle (phase d'init, programmation I/O).

Pour chaque test, ce qu'il fait et ses conditions de bon fonctionnement sont décrites dans le listing.

Qu'est-ce qu'on fait là ?

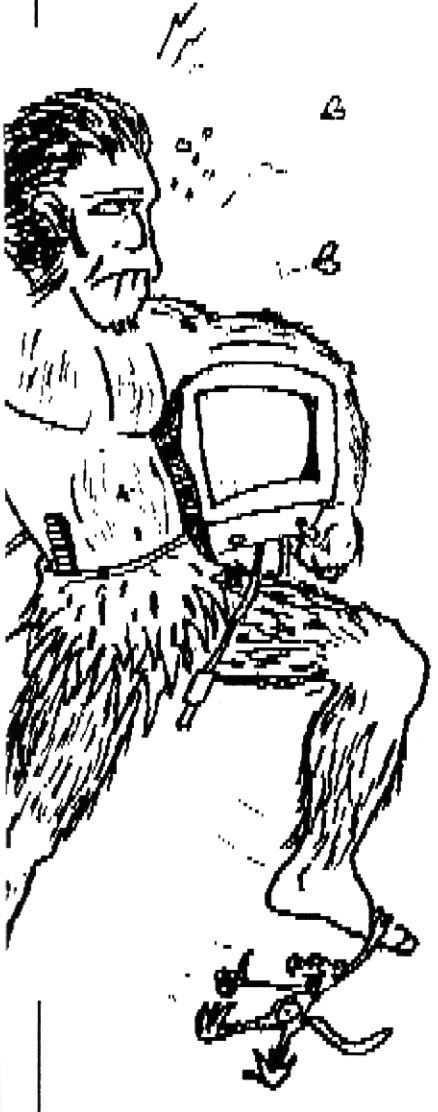
Il ne s'agit pas d'un test exhaustif, en effet, tous ces tests ont pour particularité d'être neutres ; ils ne modifient pas la programmation du CRTC et donc rien ne "saute" ou ne change de couleur l'écran. Si on décide de se dispenser de cette contrainte, il est possible d'enrichir encore ce programme d'autres routines de test comme certaines déjà publiées dans nos pages (test du mode entrelacé, test de l'overflow reg2, test du mode IM2, etc.). Toutefois, il s'agit d'une bonne base de départ qui, je le pense, vous permettra de découvrir des bits de test encore méconnus de nos chers CRTC.

OffseT

N'en déplaise au petit blond à Lunettes, voici Le Listing !

```
; Test CRTC version 3.0
; (12/01/2005) par OffseT (Futurs')

      Nolist

; Réalisé entièrement sur un véritable CPC
; Plus aucun registre du CRTC n'est modifié durant les tests !

; L'historique...

; Test CRTC version 1.1
;   - Test originel (23.02.1992) par Longshot (Logon System)
;       ce test CRTC est celui crée par Longshot et utilisé dans la
;       plupart des démos du Logon System.

; Test CRTC version 1.2
;   - Amélioration de la détection Asic (02/08/1993) par OffseT
;       le test basé sur la détection de la page I/O Asic qui
;       imposait de délocker l'Asic a été remplacé par un test
;       des vecteurs d'interruption (mode IM 2). Le délockage de
;       l'Asic n'est plus nécessaire.
;       bug connu ; ce test ne fonctionne pas nécessairement sur un
;       CPC customisé (notamment avec une interface gérant les
;       interruptions en mode vectorisé) ou sur un CPC+ dont le registre
;       Asic IVR a préalablement été modifié.
;   - Correction du bug de détection CRTC 1 (18/06/1996) par OffseT
;       sous certaines conditions de lancement le CRTC 1 était détecté
;       comme étant un CRTC 0 (on peut constater ce bug dans The Demo
;       et S&KOH). La méthode de synchronisation pour le test de détection
;       VBL a été fiabilisée et ce problème ne devrait plus survenir.

; - Test CRTC version 2.0
;   - Ajout d'un test émulateur (03/01/1997) par OffseT
;       ce test est basé sur la détection d'une VBL médiane lors de
;       l'activation du mode entrelacé. Les émulateurs n'émulent pas
;       cette VBL.
;       limitation systématique ; ce test ne permet pas de distinguer
;       un véritable CRTC 2 d'un CRTC 2 émulé.

; - Test CRTC version 3.0
;   - Retrait du test émulateur (20/12/2004) par OffseT
;       ce test ne présente aucun intéret réel et a le désavantage
;       de provoquer une VBL parasite pendant une frame.
;   - Remplacement du test Asic (29/12/2004) par OffseT
;       le nouveau test est basé sur la détection du bug de validation
;       dans le PPI émulé par l'Asic plutot que sur les interruptions
;       en mode IM 2. C'est beaucoup plus fiable puisque ça ne dépend
;       plus du tout de l'état du registre IVR ni des extensions gérant
;       les interruptions connectées sur le CPC. Merci à Ram7 pour l'astuce.
;       Limitation systématique ; l'état courant de configuration des ports
;       du PPI est perdue, mais ça ne pose normalement aucun problème.
;   - Remplacement du test CRTC 1 et 2 (29/12/2004) par OffseT
;       le test originel de Longshot était basé sur l'inhibition de
;       la VBL sur type 2 lorsque Reg2+Reg3>Reg0+1. Ce test modifiait
;       les réglages CRTC et l'écran sautait pendant une frame. Il a été
;       remplacé par un test basé sur la détection du balayage du border
;       spécifique au type 1 qui n'a pas ces inconvénients.
;       bug connu (rarissime) ; ce test renvoie un résultat erroné sur
;       CRTC 1 si reg6=0 ou reg6>reg4+1... ce qui est fort improbable.
```

```
;   - Modification du test CRTC 3 et 4 (29/12/2004) par OffseT
;       le test ne modifie plus la valeur du registre 12. Toutefois
;       il en teste la cohérence et vérifie également le registre 13.
;       limitation (rare) ; ce test ne fonctionne pas si reg12=reg13=0.
;   - Réorganisation générale des tests (29/12/2004) par OffseT
;       chaque test est désormais un module qui permet, par le biais
;       d'un système de masques de tests, de différencier les CRTC au
;       fur et à mesure.
;   - Retrait des dépendances d'interruption (29/12/2004) par OffseT
;       plus aucun test ne fait usage des interruptions pour se synchroniser.
;   - Ajout d'un test de lecture du port &BFxx (29/12/2004) par OffseT
;       ce test permet de différencier les CRTC 1 et 2 des autres et vient
;       en complément du test (historique) sur le timing VBL.
;       limitation (rare) ; ce test ne fonctionne pas si reg12=re13=0.
;   - Ajout d'un test de lecture des registres 4 et 5 (30/12/2004) par OffseT
;       ce test donne théoriquement les memes résultats que le test
;       initial de détection 3 et 4 basé sur des lectures sur le port
;       &BExx ; il consiste à lire les registres 12 et 13 via leur miroir
;       sur l'adressage des registres 4 et 5 sur type 3 et 4.
;       limitation (rare) ; ce test ne fonctionne pas si reg12=reg13=0.
;   - Ajout d'un test de lectures CRTC illégales (12/01/2005) par OffseT
;       ce test vérifie qu'on obtient bien la valeur 0 en retour
;       lors d'une tentative de lecture illégale d'un registre du
;       CRTC en ecriture seule. Ceci permet de différencier les types
;       0, 1 et 2 des types 3 et 4.
;   - Ajout d'un test du port B du PPI (12/01/2005) par OffseT
;       ce test vérifie si le port B peut-etre configuré en sortie.
;       Ceci permet d'identifier le type 3.
;       Limitation systématique ; l'état courant de configuration des ports
;       du PPI est perdue, mais ça ne pose normalement aucun problème.
;   - Ajout d'un test de détection de fin de VBL (12/01/2005) par OffseT
;       ce test vérifie que le bit 5 du registre 10 du CRTC permet bien
;       de détecter la dernière ligne de VBL sur les CRTC 3 et 4. Ceci
;       permet de différencier les types 0, 1 et 2 des types 3 et 4.
;       bug systématique ; si le bit 7 du registre 3 est à zéro (double VBL)
;       le test renvoie un mauvais résultat.
;   - Ajout d'un test de lecture du registre 31 (12/01/2005) par OffseT
;       ce test vérifie sur la valeur en lecture renvoyée pour ce registre
;       est non nulle. Si c'est le cas ça veut dire qu'on a lu soit un état
;       de haute impédance (cas du type 1) soit le registre 15 qui était non
;       nul (cas des types 3 et 4). On peut alors conclure que l'on a ni un
;       type 0, ni un type 2. Si la valeur est nulle on ne peut rien conclure
;       et le test est inopérant.
;       limitation rarissime ; ce test ne fournit pas de résultat sur type 1
;       si l'état de haute impédance est altéré
;       limitation courante ; ce test ne fournit pas de résultat sur types 3
;       et 4 si le registre 15 est nul (ce qui est la valeur par défaut)
;   - Ajout d'un test de détection des blocs 0 et 1 (12/01/2005) par OffseT
;       ce test vérifie que la détection des blocs 0 et 1 est fonctionnelle
;       sur les types 3 et 4 à l'aide des flags du registre 11 du CRTC. Ceci
;       permet de différencier les types 0,1,2 des 3,4.
;       limitation systématique ; le registre 9 doit valoir 7 sinon le
;       résultat est faux.

; Note ; une limitation décrit un cas dans lequel le test ne renvoie
; aucun résultat (il ne parvient pas à distinguer les CRTC) alors qu'un
; bug connu décrit un cas dans lequel le test peut renvoyer une mauvaise
; réponse (ce qui est beaucoup plus grave !).

; Les différents types de CRTC connus...

; 0 ; 6845SP            ; sur la plupart des CPC6128 sortis entre 85 et 87
; 1 ; 6845R             ; sur la plupart des CPC6128 sortis entre 88 et 89
; 2 ; 6845S             ; sur la plupart des CPC464 et CPC664
; 3 ; Emulé (CPC+)      ; sur les 464 plus et 6128 plus
; 4 ; Emulé (CPC old)   ; sur la plupart des CPC6128 sortis en 89 et 90.
```



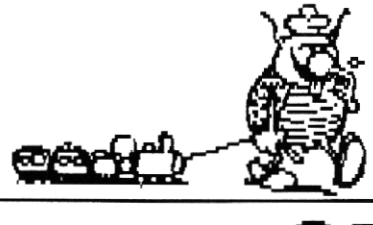
```
; Le programme qui utilise le test CRTC...

        Org &9000

        call testcrtc   ; On lance le test CRTC !
        add a,48        ; On affiche le type de CRTC
        call &bb5a
        ret             ; On rend la main

; Le test CRTC...
; Attention ! Le CRTC doit etre dans une configuration
; rationnelle pour que les tests fonctionnent (VBL et
; HBL présentes, registres 6 et 1 non nuls, bit 7 du
; registre 3 non nul, etc.)

; En sortie A contient le type de CRTC (0 à 4)
; A peut valoir &f si le CRTC n'est pas reconnu
; (mauvais émulateur CPC ou mauvaise configuration
; CRTC au lancement du test)

TestCRTC
        ld a,&ff
        ld (typecrtc),a
        di                          ; CRTC 0,1,2,3,4
        call testlongueurvbl        ;      0,1,1,0,0
        call testbfxx               ;      0,1,1,0,0,alien
        call testbexx               ;      0,0,0,1,1,alien
        call testfinvbl             ;      0,0,0,1,1,alien
        call testr4r5               ;      0,0,0,1,1,alien
        call testregswo             ;      0,0,0,1,1
        call testbloc               ;      0,0,0,1,1,alien
        call testborder             ;      0,1,0,0,0
        call testrazppi             ;      0,0,0,1,0,alien
        call testportbppi           ;      0,0,0,1,0
        call testreg31              ;      x,1,x,1,1
        ei
        ld a,(typecrtc)
        cp crtc0:jr z,type_0
        cp crtc1:jr z,type_1
        cp crtc2:jr z,type_2
        cp crtc3:jr z,type_3
        cp crtc4:jr z,type_4
        ld a,&f:ret
Type_0  ld a,0:ret
Type_1  ld a,1:ret
Type_2  ld a,2:ret
Type_3  ld a,3:ret
Type_4  ld a,4:ret
```

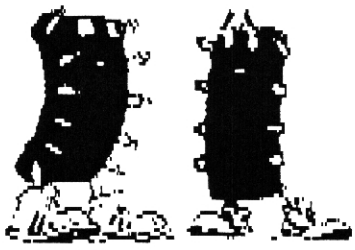
```
; Test basé sur la mesure de la longueur de VBL
; Permet de différencier les types 1,2 des 0,3,4

; Bug systématique
;   si le bit 7 du registre 3 est à zéro (double VBL)
;   le test renvoie un mauvais résultat

TestLongueurVBL
        ld b,&f5        ; Boucle d'attente de la VBL
SyncTLV1
        in a,(c)
        rra
        jr nc,synctlv1
NoSyncTLV1
        in a,(c)        ; Pre-Synchronisation
        rra             ; Attente de la fin de la VBL
        jr c,nosynctlv1
SyncTLV2
        in a,(c)        ; Deuxième boucle d'attente
        rra             ; de la VBL
        jr nc,synctlv2

        ld hl,140       ; Boucle d'attente de
WaitTLV dec hl          ; 983 micro-secondes
        ld a,h
        or l
        jr nz,waittlv
        in a,(c)        ; Test de la VBL
        rra             ; Si elle est encore en cours
        jp c,type12     ; on a un type 1,2...
        jp type034      ; Sinon on a un type 0,3,4


; Test basé sur la lecture des registres 12 et 13
; sur le port &BFxx
; Permet de différencier les types 0,3,4 et 1,2

; Limitation rare
;   si reg12=reg13=0 le test est sans effet

TestBFxx
        ld bc,&bc0c     ; On sélectionne le reg12
        out (c),c
        ld b,&bf        ; On lit sa valeur
        in a,(c)
        ld c,a          ; si les bits 6 ou 7 sont
        and &3f         ; non nuls alors on a un
        cp c            ; problème
        jp nz,typealien
        ld a,c
        or a            ; si la valeur est non nulle
        jp nz,type034   ; alors on a un type 0,3,4
        ld bc,&bc0d
        out (c),c       ; On sélectionne le reg13
        ld b,&bf
        in a,(c)        ; On lit sa valeur
        or a            ; Si la valeur est non nulle
        jp nz,type034   ; alors on a un type 0,3,4
        ret


; Test basé sur la lecture des registres 12 et 13
; à la fois sur les ports &BExx et &BFxx
; Permet de différencier les types 0,1,2 des 3,4

; Limitation rare
;   si reg12=reg13=0 le test est sans effet

TestBExx
        ld bc,&bc0c     ; On sélectionne le registre 12
        out (c),c       ; On compare les valeurs sur
        call cpbebf     ; les ports &BExx et &BFxx
        push af         ; (on sauve les flags)
        ld b,a          ; Si le bit 6 ou 7 de la valeur
        and &3f         ; lue pour &BFxx est non nul
        cp b            ; alors on a un problème
        call nz,typealien
        pop af          ; (on récupère les flags)
        jp nz,type012   ; Si elles sont différentes
        xor a           ; on a un type 0,1,2
        cp c            ; Si elles sont égales et
        jp nz,type34    ; non nulles on a un type 3,4
        ld bc,&bc0d     ; On sélectionne le registre 13
        out (c),c       ; On compare les valeurs sur
```

```
        call cpbebf     ; les ports &BExx et &BFxx
        jp nz,type012   ; Si elles sont différentes
        xor a           ; on a un type 0,1,2
        cp c            ; Si elles sont égales et
        jp nz,type34    ; non nulles on a un type 3,4
        ret

CPBEBF  ld b,&be        ; On lit la valeur sur &BExx
        in a,(c)
        ld c,a          ; On la stocke dans C
        inc b
        in a,(c)        ; On lit la valeur sur &BFxx
        cp c            ; On la compare à C
        ret


; Test basé sur la RAZ du PPI
; Permet de différencier les types 0,1,2,4 du 3

; Limitation systématique
;   l'état courant de configuration des ports
;   du PPI est perdu

TestRAZPPI
        ld bc,&f782     ; On configure le port C
        out (c),c       ; en sortie
        dec b
        ld c,&f         ; On place une valeur sur
        out (c),c       ; le port C du PPI
        in a,(c)        ; On vérifie si la valeur est
        cp c            ; toujours là en retour
        jp nz,typealien ; sinon on a un problème
        inc b
        ld a,&82        ; On configure de nouveau
        out (c),a       ; le mode de fonctionnement
        dec b           ; des ports PPI
        in a,(c)        ; On teste si la valeur placée
        cp c            ; sur le port C est toujours là
        jp z,type3      ; Si oui on a un type 3
        or a            ; Si elle a été remise à zéro
        jp z,type0124   ; on a un type 0,1,2,4
        jp typealien    ; Sinon on a un problème


; Test basé sur la détection du balayage des lignes
; hors border
; Permet d'identifier le type 1

; Bug connu rarissime
;   si reg6=0 ou reg6>reg4+1 alors le test est faussé !

TestBorder
        ld b,&f5
NoSyncTDB1
        in a,(c)        ; On attend un peu pour etre
        rra             ; sur d'etre sortis de la VBL
        jr c,nosynctdb1 ; en cours du test précédent
SyncTDB1
        in a,(c)        ; On attend le début d'une
        rra             ; nouvelle VBL
        jr nc,synctdb1
NoSyncTDB2
        in a,(c)        ; On attend la fin de la VBL
        rra
        jr c,nosynctdb2

        ld ix,0         ; On met à zéro les compteurs
        ld hl,0         ; de changement de valeur (IX),
        ld d,1          ; de ligne hors VBL (HL) et
        ld e,d          ; de ligne hors border (DE)
        ld b,&be
        in a,(c)
        and 32
        ld c,a
SyncTDB2
        inc de          ; On attend la VBL suivante
        ld b,&be        ; en mettant à jour les divers
        in a,(c)        ; compteurs
        and 32
        jr nz,border
        inc hl          ; Ligne de paper !
        jr noborder
Border  ds 4
NoBorder
        cp c
        jr z,nochange
        inc ix          ; Transition paper/border !
        jr change
NoChange
        ds 5
Change  ld c,a

        ds 27

        ld b,&f5
        in a,(c)
        rra
        jr nc,synctdb2  ; On boucle en attendant la VBL
        db &dd:ld a,l   ; Si on n'a pas eu juste deux
        cp 2            ; transitions alors ce n'est
        jp nz,type0234  ; pas un type 1
        jp type1        ; Pour plus de fiabilité au
                        ; regard de l'état de haute
                        ; impédance sur les CRTC autres
                        ; que le type 1 on pourrait
                        ; vérifier ici que HL vaut
                        ; reg6*(reg9+1) mais ça impose
                        ; de connaitre au préalable la
                        ; valeur de ces deux registres


; Test basé sur la lecture des registres 4 et 5
; Permet de différencier les types 0,1,2 des 3,4

; Limitation rare
;   si reg12=reg13=0 le test est sans effet

TestR4R5
        ld bc,&bc0c     ; On sélectionne le registre 12
        out (c),c       ; On compare les valeurs en
        call cprhrl     ; retour sur le port &BFxx
        push af         ; On sauve les flags
        ld b,a          ; Si le bit 6 ou 7 de la valeur
        and &3f         ; lue pour &BFxx est non nul
        cp b            ; alors on a un problème
        call nz,typealien
        pop af          ; On récupère les flags
        jp nz,type012   ; Si elles sont différentes
        xor a           ; on a un type 0,1,2
        cp c            ; Si elles sont égales et
        jp nz,type34    ; non nulles on a un type 3,4
        ld bc,&bc0d     ; On sélectionne le registre 13
        out (c),c       ; On compare les valeurs en
        call cprhrl     ; retour sur le port &BFxx
```

```
        jp nz,type012   ; Si elles sont différentes
        xor a           ; on a un type 0,1,2
        cp c            ; Si elles sont égales et
        jp nz,type34    ; non nulles on a un type 3,4
        ret

CPRHRL  ld b,&bf        ; On lit la valeur du registre
        in a,(c)        ; High sur &BFxx
        ld b,&bc        ; Sélection du registre Low
        res 3,c         ; On passe sur le registre Low
        out (c),c
        ld c,a          ; On la stocke dans C
        ld b,&bf
        in a,(c)        ; On lit la valeur sur &BFxx
        cp c            ; On la compare à C
        ret


; Test basé sur la valeur de retour sur les registres
; CRTC en écriture seule
; Permet de différencier les types 0,1,2 des types 3,4

; Aucune limitation/bug connus

TestRegsWO
        ld de,0         ; On lance le parcours des
        ld c,12         ; registres 0 à 11 avec
        call cumulereg  ; cumul de la valeur retour
        xor a           ; Si le résultat cumulé de
        cp d            ; la lecture est non nul
        jp nz,type34    ; alors on a un type 3,4
        jp type012      ; Sinon, c'est un type 0,1,2
CumuleReg
LoopTRI ld b,&bc        ; On sélectionne le
        out (c),e       ; registre "E"
        ld b,&bf        ; On lit la valeur
        in a,(c)        ; renvoyée en retour
        or d            ; On la cumule dans D avec
        ld d,a          ; les lectures précédentes
        inc e           ; On boucle jusqu'au
        ld a,e          ; registre "C"
        cp c
        jr nz,looptri
        ret


; Test basé sur la possibilité de programmer le port B
; en sortie
; Permet d'identifier le type 3

; Limitation systématique
;   l'état courant de configuration des ports
;   du PPI est perdu

TestPortBPPI
        ld b,&f5
SyncTPBP1
        in a,(c)
        rra
        jr nc,synctpbp1
NoSyncTPBP1
        in a,(c)        ; Pre-Synchronisation
        rra             ; Attente de la fin de la VBL
        jr c,nosynctpbp1
        ld bc,&f782     ; On configure le port B
        out (c),c       ; du PPI en entrée
        ld b,&f5        ; On lit la valeur présente
        in a,(c)        ; sur le port B puis on
        xor 254         ; la modifie judicieusement
        ld e,a          ; et on la stocke dans E
        ld d,&f5
        ld bc,&f780     ; On configure le port B
        out (c),c       ; du PPI en sortie
        ld b,d          ; On y envoie la valeur
        out (c),e       ; stockée dans E
        in a,(c)        ; On relit le port B
        ld bc,&f782     ; On reconfigure le port B
        out (c),c       ; du PPI en entrée
        cp e            ; Si la valeur E a été lue
        jp z,type0124   ; alors on a type 0,1,2,4
        jp type3        ; Sinon on a un type 3


; Test basé sur la détection de la dernière ligne VBL
; Permet de différencier les types 0,1,2 des 3,4

; Bug systématique
;   si le bit 7 du registre 3 est à zéro (double VBL)
;   le test renvoie un mauvais résultat)

TestFinVBL
        ld bc,&bc0a     ; On sélectionne le
        out (c),c       ; registre 10 du CRTC
        ld b,&f5
NoSyncTFV1
        in a,(c)        ; Pre-Synchronisation
        rra             ; Attente de la fin de la VBL
        jr c,nosynctfv1

        ld b,&bf        ; On lit l'état du registre 10
        in a,(c)
        and 32          ; Si le bit5 est nul alors
        jp z,type012    ; on a un type 0, 1 ou 2

        ld b,&f5
SyncTFV2
        in a,(c)        ; Boucle d'attente de la VBL
        rra
        jr nc,synctfv2
        ld hl,55        ; Boucle d'attente de
WaitTfV dec hl          ; 388 micro-secondes
        ld a,h
        or l
        jr nz,waittfv

        ld b,&bf        ; On lit l'état du registre 10
        in a,(c)
        and 32          ; Si le bit5 est nul
        jp z,typealien  ; on a un problème

        ld b,13         ; Boucle d'attente de
        djnz $          ; 54 micro-secondes
        ld b,&bf        ; On lit l'état du registre 10
        in a,(c)
        and 32          ; Si le bit5 est non nul
        jp nz,typealien ; on a un problème

        ld b,13         ; Boucle d'attente de
        djnz $          ; 54 micro-secondes
        ld b,&bf        ; On lit l'état du registre 10
        in a,(c)
        and 32          ; Si le bit5 est non nul
        jp nz,type34    ; on a un type 3 ou 4
        jp typealien    ; Sinon on a un problème
```

```
; Test basé sur le statut particulier du registre 31
; Permet d'identifier les types 1, 3 et 4

; Limitation rarissime
;   ce test ne fournit pas de résultat sur type 1
;   si l'état de haute impédance est altéré
; Limitation courante
;   ce test ne fournit pas de résultat sur types 3 et 4
;   si le registre 15 est nul (ce qui est la valeur par
;   défaut)

TestReg31
        ld bc,&bc1f     ; On sélectionne le registre 31
        out (c),c       ; et on fait une tentative de
        ld b,&bf        ; lecture sur le port &bfxx
        in a,(c)        ; Si on a une valeur non nulle
        jp nz,type134   ; alors c'est un type 1,3,4
        ret             ; sinon on ne peut rien
                        ; conclure


; Test basé sur la détection des blocs 0 et 1
; Permet de différencier les types 0,1,2 des 3,4

; Limitation systématique
;   le registre 9 doit valoir 7 sinon le résultat
;   est faux

TestBloc
        ld bc,&bc0b     ; On sélectionne le
        out (c),c       ; registre 10 du CRTC
        ld b,&f5
NoSyncTB1
        in a,(c)        ; Pre-Synchronisation
        rra             ; Attente de la fin de la VBL
        jr c,nosynctb1
SyncTB2 in a,(c)        ; Boucle d'attente de la VBL
        rra
        jr nc,synctb2
NoSyncTB2
        in a,(c)
        rra             ; Attente de la fin de la VBL
        jr c,nosynctb2

        ld b,&bf        ; On lit l'état du registre 11
        in a,(c)        ; (on est sur le bloc 1)
        ld d,a

        ld b,14         ; On attend 58 micro-secondes
        djnz $

        ld b,&bf
        in a,(c)        ; On lit l'état du registre 11
        ld c,a          ; (on est sur le bloc 2)

        ld b,96         ; On attend 386 micro-secondes
        djnz $

        ld b,&bf
        in a,(c)        ; On lit l'état du registre 11
        ld e,a          ; (on est sur le bloc 0)
        or d            ; Si on n'a pas lu une valeur
        or e            ; nulle à chaque fois alors
        jr nz,tbactif   ; on peut continuer
        jp type012      ; Sinon on a un type 0, 1 ou 2
TBActif ld a,&a0        ; Si pour le bloc 0 on n'a pas
        and e           ; bit7=0 et bit5=0
        jp nz,typealien ; alors on a un problème
        ld a,&a0        ; Si pour le bloc 1 on n'a pas
        and d           ; bit7=1 et bit5=1
        cp &a0          ; alors on a un problème
        jp nz,typealien
        ld a,&a0        ; Si pour le bloc 2 on n'a pas
        and c           ; bit7=0 et bit5=1
        cp &20          ; alors on a un problème
        jp nz,typealien
        jp type34       ; Sinon on a un type 3 ou 4

; Routines de typage

CRTC0   Equ 1
CRTC1   Equ 2
CRTC2   Equ 4
CRTC3   Equ 8
CRTC4   Equ 16

Type012 ld a,(typecrtc)
        and crtc0+crtc1+crtc2
        ld (typecrtc),a
        ret
Type0124
        ld a,(typecrtc)
        and crtc0+crtc1+crtc2+crtc4
        ld (typecrtc),a
        ret
Type0234
        ld a,(typecrtc)
        and crtc0+crtc2+crtc3+crtc4
        ld (typecrtc),a
        ret
Type034 ld a,(typecrtc)
        and crtc0+crtc3+crtc4
        ld (typecrtc),a
        ret
Type1   ld a,(typecrtc)
        and crtc1
        ld (typecrtc),a
        ret
Type12  ld a,(typecrtc)
        and crtc1+crtc2
        ld (typecrtc),a
        ret
Type134 ld a,(typecrtc)
        and crtc1+crtc3+crtc4
        ld (typecrtc),a
        ret
Type3   ld a,(typecrtc)
        and crtc3
        ld (typecrtc),a
        ret
Type34  ld a,(typecrtc)
        and crtc3+crtc4
        ld (typecrtc),a
        ret
TypeAlien
        xor a
        ld (typecrtc),a
        ret

; Variables

        List
TypeCRTC
        db &ff
```



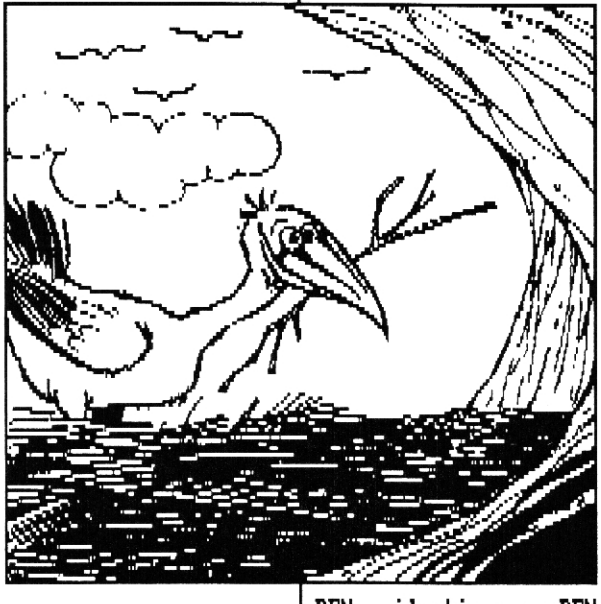


DOSSIER

En digne et juste complément du dossier sur les ROM du numéro 20, voici comme promis une description des principales ROM d'outils disponibles sur CPC pour le développement.

L'incontournable Utopia

La ROM d'outil qui s'avère rapidement indispensable à tout CPCiste souhaitant bidouiller un petit peu sa machine est sans conteste Utopia. Il s'agit d'une ROM de second plan qui ajoute au système toute une série de RSX aussi bien pour manipuler votre mémoire et vos disquettes que pour vous assister dans l'élaboration de vos créations en Basic. De plus, la plupart des RSX proposées par Utopia offrent une assistance à la saisie des paramètres ! Plus besoin de connaître par cœur la syntaxe exacte ! Tapez votre RSX sans aucun paramètre et Utopia vous les demandera gentiment un par un. Ceci vaut également pour les commandes par défaut de l'Amsdos qu'Utopia prend en charge comme REN ou ERA.



Je vois que le petit blond à lunettes est de plus en plus excité à l'idée d'utiliser tous ces outils... aussi, nous allons commencer par faire un rapide tour d'horizon des diverses RSX additionnelles d'Utopia.

Les outils disque

Une partie importante des RSX est consacrée à la gestion des fichiers et plus largement des disquettes. En effet, force est de constater que ce que nous propose la ROM Amsdos par défaut est assez minimaliste et que l'on doit souvent avoir recours à des outils externes comme Discologie pour des opérations aussi simple que protéger un fichier en écriture ou formater une disquette. Oubliez tout ça ! Avec Utopia la plupart des opérations disque deviennent possibles directement depuis le Basic ! Voici un petit mémo des RSX liées à la gestion disque qui vous donnera une idée de toutes les opérations directement accessibles :

ACCESS : modifie les droits d'accès sur les fichiers (protégé/système).

CAT : identique au CAT du Basic sauf qu'il a besoin de moins de mémoire (plus de MEMORY FULL).

COPY : permet tout simplement la copie de fichiers.

DEDIT : permet d'éditer physiquement les secteurs d'une disquette ! C'est pratiquement l'équivalent de l'éditeur de secteurs de Discologie... sauf qu'il est disponible sous Basic ! Très pratique pour faire du "catalogue art" par exemple.

DELETE/ERA : identique à la commande ERA de l'Amsdos.

DISCCOPY : copie rapide de disquettes.

DISCTEST : vérification rapide d'une disquette.

DUMP : liste un fichier en hexa directement depuis la disquette.

FORMAT : formate une disquette.

INFO : liste les type, adresse de chargement, longueur et adresse d'exécution des fichiers (c'est un CAT détaillé).

LOAD : charge un fichier de type quelconque (même ASCII !) à une adresse spécifiée.

REN : identique au REN de l'Amsdos.

RUN : identique au RUN"fichier" du Basic sauf que le système n'est pas réinitialisé.

SAVE : sauve un fichier binaire à partir de données en mémoire.

SAVEA : sauve un fichier ASCII (!) à partir de données en mémoire.

SPOOL/SPOOLOFF : active/désactive la capture de l'affichage vers un fichiers ASCII.

TYPE/LIST : liste un fichier en ASCII sans/avec les numéros de ligne directement depuis la disquette.

VERIFY/VTEXT : compare une zone mémoire avec le contenu d'un fichier et affiche les/la différence(s).

Les outils Basic/Dév

En complément de tout ces outils disque, Utopia offre également divers outils particulièrement dédiés aux programmeurs en Basic. Toutefois, comme le fait justement remarquer le petit blond à lunettes, le Basic servant très souvent de "shell" pour le développement en assembleur, la plupart de ces outils sont également très utiles aux codeurs !

ARRAYS : liste les tableaux de variables existants et leur taille.

C : effectue des calculs en 16 bits non signé.

CALL : lance une routine assembleur avec le choix de la valeur des registres Z80 en entrée et l'affichage de leur contenu en sortie.

FIND : cherche une chaîne de caractères dans le programme Basic.

FN : affiche la liste des fonctions Basic définies par l'utilisateur (commande DEF FN).

LINK : réinitialise tous les pointeurs Basic (permet entre autre de déprotéger un programme Basic protégé).

MDUMP/MEDIT : liste/édite la mémoire en HEXA.

MOVE : déplace des lignes dans un programme Basic.

NOKEYS : efface toutes les définitions de touches (KEY DEF)

NORSX : désinstalle tous les RSX RAM (aucun effet sur les RSX ROM). Très pratique lorsqu'on est en train de développer des RSX !

REPLACE : cherche et remplace une chaîne de caractères par une autre dans un programme Basic.

STATUS : affiche l'état des allocations mémoire des diverses structures du système (Symbol, Himem, etc.).

TOKENS : affiche la configuration courante des définitions de touches.

VARS : affiche la liste des variables, leur type et leur contenu.

Les outils système

Utopia contient enfin une série d'outils liés à la gestion du système dans sa globalité.

CDUMP/GDUMP : impression du contenu de l'écran en mode caractère/graphique (une bête copie d'écran).

HELP : liste les ROMs installées et leur allocation mémoire. Avec un numéro de ROM en paramètre, donne la liste des RSX de la ROM.

HELPR : liste les RSX installées en RAM.

PRINTON/PRINTOFF : active/désactive la redirection de l'affichage vers l'imprimante.

ROMOFF/ROMON : reboote le CPC sans/avec les ROM spécifiées.

U : lance une RSX d'Utopia (éventuellement avec ses paramètres s'ils sont spécifiés) désignée par son nom. Ceci permet d'accéder aux RSX d'Utopia qui ont été surchargées par celles d'une autre ROM ou par des RSX installées en RAM.

X/XROM : lance la RSX désignée par son nom dans la ROM de son choix.

Cerise sur le gâteau

Enfin, Utopia configure par défaut des raccourcis clavier sur les touches de fonctions couplées à CONTROL. Ces raccourcis assez bien pensés permettent un accès direct aux commandes qui sont effectivement souvent utilisées comme CAT, MODE 2, LIST, ERA,"*.BAK", STATUS, et pour les utilisateurs du kit de développement Arnor, PROTEXT et M,2 !

En conclusion on peut dire qu'il s'agit vraiment d'une ROM incontournable pour tout CPCiste qui utilise vraiment son CPC pour bidouiller ou programmer.

La ROM Maxam 1.5

Indépendamment de son assembleur dont nous parlerons plus loin, cette ROM qui fait partie du fameux kit de développement Arnor se révèle très intéressante pour son moniteur et debugger Z80. En effet, il permet de définir simplement les conditions d'exécution du code à tester et d'y placer divers types de breakpoints afin d'en observer judicieusement le fonctionnement. De plus, comme il s'agit d'une ROM parfaitement intégrée, elle n'a besoin que de quelques octets de RAM pour fonctionner ce qui lui donne un avantage certain par rapport aux autres moniteurs qui ont besoin de se charger en RAM. Son utilisation depuis le Hacker rend également de fiers services.

Le moniteur/debugger

Avant de vous donner une rapide description des commandes (RSX) du moniteur/débugger de Maxam, je vais vous en expliquer le principe. Il faut vous imaginer que le programme que vous allez tester est exécuté dans

un espace de travail parallèle. Grâce aux commandes du debugger vous pouvez définir la configuration de cet espace de travail (valeur des registres, breakpoints, etc.) et vous avez la possibilité de commuter entre l'espace d'exécution du programme et le debugger (et inversement) très simplement en modifiant éventuellement les conditions d'exécution entre temps. Ceci étant bien à l'esprit, vous allez trouver ci-dessous la liste des commandes disponibles.

AA/AF/BC/DE/HL/IX/IY/PC : charge la valeur passée en paramètre dans le registre correspondant. Si le paramètre est omis alors ces RSX affichent simplement la valeur en cours de tous les registres et la prochaine instruction qui sera exécutée.

REGS : affiche la configuration détaillée de l'espace de travail : valeur de tous les registres, mémoire pointée par chacun des registres, état des flags et des commutations mémoire, prochaines instructions, etc.

BR : placer un breakpoint à l'adresse spécifiée en paramètre (8 breakpoints au maximum). Lorsque le programme arrivera à cette adresse il sera interrompu et le debugger affichera son état courant de la même manière que la commande REGS.

TB : placer un breakpoint temporaire à l'adresse spécifiée en paramètre (un seul breakpoint de ce type est possible). Ce type de point d'arrêt se comporte exactement de la même manière que ceux positionnés via la commande BR hormis le fait qu'il ne s'activent qu'au premier passage du programme.

CB : annuler le breakpoint de l'adresse spécifiée en paramètre.

CAB : annuler tous les breakpoints.

DB : lister les breakpoints.

QB : activer le mode rapide des breakpoints (c'est le mode par défaut). Dans ce mode l'état courant du programme interrompu est affiché aussitôt un breakpoint rencontré.

WB : activer le mode d'attente des breakpoints. Dans ce mode le debugger attend que l'utilisateur presse une touche avant d'afficher l'état courant du programme interrompu.

BRKOFF : désactiver les breakpoints.

BRKON : activer les breakpoints (état par défaut).

J : exécution d'un programme en mode debug ; c'est à dire avec la gestion des breakpoints et l'utilisation de l'espace de travail (valeurs des registres et pile utilisateur activés). Si une adresse est spécifiée en paramètre elle sera chargée dans PC avant de passer en exécution. Si aucune adresse n'est spécifiée alors la valeur courante de PC dans l'espace de travail sera utilisée. Par défaut PC est placé à &33 (il y a un RET à cette adresse) ; il est en outre automatiquement positionné au dernier ORG rencontré par l'assembleur de Maxam (commandes ASM, ASSEM ou ASSEMBLE).

O : reprend l'exécution d'un programme interrompu par un breakpoint. Typiquement cette commande est très semblable à J sauf qu'elle n'effectue par les initialisations de celles-ci relatives à la gestion de la pile. J ne doit pas être utilisé pour reprendre l'exécution d'un programme interrompu ; en revanche, si vous utilisez O plutôt que J pour lancer l'exécution initiale d'un programme en mode debug cela fonctionnera car le debugger détecte qu'il s'agit d'une première exécution et O est alors parfaitement équivalent à J. Vous pouvez donc utiliser systématiquement O plutôt que J pour lancer vos programmes préalablement assemblés.

DI/DIF : désassemble à l'écran ou dans un fichier.

LI/LIF : liste la mémoire à l'écran ou dans un fichier.

M : lance le moniteur de Maxam. Ce moniteur est assez complet et intègre des outils pour éditer la mémoire, déplacer des blocs de mémoire, manipuler les commutation de ROM, désassembler et même reloger des programmes assembleur.

MSH/MSL : place l'écran système en mémoire haute (&C000) ou basse (&4000). Le choix de la configuration vidéo du système s'avère souvent être un outil très pratique pour étudier des programmes prévus pour s'exécuter en &C000-&FFFF (ce qui est le cas des ROM hautes).

Voilà, nous avons fait le tour des principales commandes du moniteur/debugger de Maxam 1$^{1/2}$. Il va de soi que n'importe quelle commande est utilisable à n'importe quel moment, même après un breakpoint avant de relancer l'exécution du programme (commande O).

L'assembleur de Maxam 1.5

La ROM de Maxam contient également un assembleur très souple et parfaitement respectueux du système. Celui-ci peut-être invoqué grâce à quatre commandes et gère une multitude de directives d'assemblage. Voyons en premier lieu les commandes d'assemblage.

ASM : assemble de code stocké dans l'éditeur de Protext. Protext fait en effet également parti du kit de développement Arnor et Maxam sait en tirer profit. La valeur de PC pour l'espace programme est placé automatiquement au dernier ORG rencontré lors de l'assemblage.

TASM : identique à ASM sauf que l'assembleur ne fait pas de résolution des symboles. Très pratique pour vérifier la syntaxe d'un module (include) dépendant d'autres fichiers d'assemblage sans avoir à tout compiler systématiquement.

ASSEMBLE : assemble le code stocké sous forme de remarques (REM ou ') dans le programme Basic. Des pointeurs de variables Basic peuvent être passés en paramètre ce qui permettra de les récupérer ou de les affecter grâce aux directives d'assemblage GET et PUT.

ASSEM : identique à ASSEMBLE hormis le fait qu'aucun affichage n'est généré pendant l'assemblage.

Tout ceci nous amène tout naturellement à nous demander quelles sont les directives d'assemblage dont nous disposons sous Maxam. Comme nous avons pour habitude de vous gâter, j'ai pris mon courage à deux mains et je vous en livre tout de suite la liste avec la petite description qui va bien à chaque fois.

Les directives de compilation

ORG : c'est la directive de base, commune à tous les assembleurs et qui permet de définir l'adresse où assembler votre code source. Si un seul paramètre est spécifié alors le code est généré pour cette adresse et y est également stocké. Si deux paramètres sont fournis alors le code est généré pour la première mais stocké à la seconde.

LIMIT : permet de spécifier une adresse limite au-delà de laquelle on ne veut pas assembler. Par défaut limit est fixé au HIMEM et toute tentative d'assemblage au dessus du limit renverra un "CODE LIMIT EXCEEDED".

NOCODE/CODE : désactive/réactive le pokage en RAM du code assemblé. Ces directives sont très utiles lorsque vous avez des programmes relativement complexes dans lesquels vous devez évaluer des pseudo-structures sans pour autant qu'elles représentent du code à assembler effectivement.

END/STOP : cesse l'assemblage de tous le programme ou de l'include en cours.

DB/DEFB/DEFM/BYTE/TEXT : poker des octets. Ces octets peuvent être fournis sous forme de nombres ou de chaînes de caractères.

DW/DEFW/WORD : poker des words.

DS/DEFS/RMEM : réserver des octets mémoire. Le premier paramètre représente le nombre d'octets et le second la valeur à mettre dans la zone réservée. Si le second paramètre est omis alors des zéros sont pokés.

STR : poker une chaîne de caractère avec le bit d'arrêt (bit 7) automatiquement positionné sur le dernier caractère (les noms de RSX sont stockés selon ce formalisme par exemple).

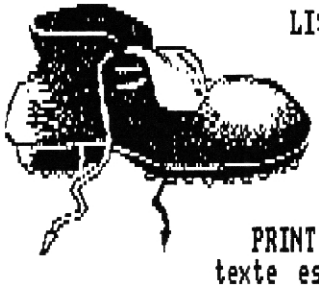
IF/IFNOT/ELSE/ENDIF : permet de faire de l'assemblage conditionnel. Les IF peuvent être imbriqués jusqu'à 10 fois ce qui est assez énorme et ne représente pas réellement une limite.

IF1/IF2 : renvoient TRUE lors de la passe d'assemblage 1 ou 2. Ceci permet de faire des assemblages conditionnels différents selon que l'assembleur est en train d'évaluer les expressions (passe 1) ou de faire la résolution des symboles et de poker le code en mémoire (passe 2).

READ : permet de faire une inclusion d'un source depuis un autre. Très pratique pour rendre son code modulaire et assembler des programmes qui ne tiennent pas en mémoire. Il peut y avoir autant que READ que vous voulez dans un source ; en revanche ils ne peuvent pas être imbriqués.

WRITE/CLOSE : permet de sauver le code assemblé situé entre les balises WRITE/CLOSE dans un binaire sur disque.

NOLIST/LIST : désactive/réactive la sortie écran du résultat de l'assemblage. Dans tous les cas les messages d'erreur sont affichés.



LIST F : active l'écriture d'un journal du résultat de l'assemblage dans un fichier ASCII (dont le nom est spécifié en paramètre).

PRINT : affiche un texte. Ce texte est une chaîne de caractères dans laquelle deux caractères d'échappement sont disponibles : $ et &. Immédiatement suivis du nom d'une variable ceux-ci permettent son affichage en décimal ou en hexadécimal au milieu de la chaîne.

PAUSE : l'assembleur fera une pause à cet endroit là et attendra que vous pressiez une touche pour continuer.

DUMP : affiche la table des symboles. Extrêmement pratique d'autant plus que grâce à la directive LIST F cette table peut-être directement stockée dans un fichier ASCII.

EQU/LET : définit une constante ou une variable de compilation. Les constantes sont fixées une fois pour toute pour toute la phase d'assemblage alors qu'une variable peut-être réattribuée (très pratique en conjonction avec NOCODE pour définir des structures de données dynamiquement allouées).

GET/PUT : permet de manipuler les variables passées en paramètre via les commandes ASSEMBLE ou ASSEM. GET sert à récupérer les valeurs de variables Basic en paramètre et PUT sert à leur attribuer une nouvelle valeur. Ceci permet de faire des "makefiles" en Basic.

BRK : insère un breakpoint en dur dans le programme. Ce breakpoint se comporte de la même manière que ceux définis de façon dynamique à l'aide des RSX du débugger mais sont directement inclus dans le code assemblé.

LIST P/PLEN/PAGE/TITLE/WIDTH : ces directives sont dédiées au formatage de la sortie sur imprimante du résultat de l'assemblage. D'un commun accord avec le petit blond à lunettes nous avons décidé de ne pas approfondir.

Labels et types

Concernant les noms de label, il n'y a ni limite de nombre, ni limite de longueur. Les labels ont le même statut que les constantes (définies par EQU) et peuvent être utilisés de la même manière : valeurs pour les instructions assembleur, attributions pour des variables de compilation (LET) ou des variables Basic (PUT). Comme dans la plupart des autres assembleurs le symbole "$" désigne l'adresse d'assemblage en cours et peut-être utilisé comme un label.

Pour les valeurs numériques, l'assembleur prend par défaut des valeurs décimales. Pour le binaire la valeur devra être précédée d'un "%" ; un "&" ou un "#" annonce quant à lui un nombre en hexadécimal. Les opérations arithmétiques et logiques sont possibles dans les paramètres des instructions et des directives mais aucune gestion des priorités d'opération n'est effectuée, donc attention.

Les directives ORG et LIMIT peuvent être utilisées plusieurs fois dans un même code source et acceptent comme toutes les autres directives aussi bien des valeurs que des constantes ou des variables en paramètre.

Le mot de la fin sur Maxam

J'espère que cette description assez complète mais qui n'a pas non plus pour objectif d'être un didactitiel vous aura donné une bonne idée des possibilités de Maxam qui sont sans équivalent sur CPC. Ceci étant... un bon moniteur/assembleur ne fait pas tout ; il faut un éditeur ergonomique pour le rendre réellement agréable à utiliser. Certes Maxam est utilisable depuis le Basic, mais l'écriture des codes sources dans les programmes Basic, même si elle est facilitée par diverses RSX, n'en demeure pas moins fastidieuse ! Heureusement, Protext est là !

En pratique le Basic sert pour écrire de petits codes liés à la gestion de la compilation et de la mémoire (des pseudo-makefiles) alors que Protext sert réellement à l'édition des codes sources.

Protext

Contrairement à Maxam 1.5 et Utopia, Protext n'est pas uniquement disponible en ROM. Il existe en effet diverses versions disque (sous Amsdos, sous CP/M ou même sous AmigaOS ou MSDOS). Toutefois, comme nous sommes ici dans un dossier consacré aux ROM, vous l'avez deviné, nous ne parlerons que de cette version bien que la plupart des fonctionnalités soient rigoureusement les mêmes pour toutes les versions.

Que permet donc de faire Protext ? Eh bien il s'agit d'un éditeur de texte hautement configurable disposant de plus de fonctions de traitement de texte. En pratique, Protext a deux modes d'éditions : le mode "program" et le mode "document". Dans le premier mode nous pourrez éditer confortablement n'importe quel document ascii (un code source assembleur ou C par exemple). Dans le deuxième mode, vous pourrez agencer votre texte en vue de l'imprimer comme n'importe quel traitement de texte. A titre de comparaison, Protext est nettement supérieur à Semword pour faire du traitement de texte ; en revanche, Textomat lui reste très largement supérieur (multi-colonnage, détourage, impression proportionnelle, impression circulaire, etc.).

Par ailleurs, contrairement à Semword ou Textomat, Protext est un éditeur à l'ancienne avec un mode édition et un mode commande (on passe de l'un à l'autre à l'aide de la touche ESC). Si cela peut paraître rébarbatif au premier abord, c'est pourtant ce qui fait de Protext le meilleur éditeur de texte sur CPC ! En mode commande, vous avez non seulement accès à toutes les fonctions internes de Protext, mais également à toutes les RSX comme si vous étiez sous un shell (comme sous le CLI de Rodos). Dès lors, vous avez accès depuis Protext à toutes les commandes de toutes les ROM.

Un puissant éditeur de texte

Il s'agit d'un éditeur de texte plus que complet. Vous pourrez faire avec Protext tout ce que vous faites avec un éditeur de texte moderne : rechercher et remplacer des chaînes, déplacer/copier/coller des bouts de texte, importer/exporter sur disque tout ou partie de textes, placer des repères, etc. En résumé, on peut dire que vous avez absolument tout à part l'indentation automatique ou la colorisation des codes sources (ce qui est rigoureusement inutile pour de l'assembleur). Protext charge et sauve des fichiers ascii tabulés standard ; il est donc compatible avec pratiquement tout le monde (y compris avec l'éditeur de Maxam 1.14). Lors de l'utilisation, le texte en cours est stocké dynamiquement en mémoire centrale entre le programme Basic et le HIMEM : 100% OS-friendly.

Un bon traitement de texte

Protext est un bon traitement de texte... mais certainement pas le meilleur. Il permet de mettre simplement en oeuvre des mises en pages simples et est idéal pour rédiger des lettres, des documentations, ou de petits livres. Il gère la pagination, le vis-à-vis des pages, les entêtes et pieds de page, les polices de caractères, la justification et les marges, tous les types de tabulations, etc. Nous l'utilisons par exemple pour tout ce qui concerne notre courrier. Il est extrêmement configurable et il est facile de se faire soi-même le driver adapté à son imprimante.

En ce qui concerne l'utilisation des polices de caractères et des effets graphiques (gras, italique, souligné, exposant, indice), Protext fonctionne comme tous les autres traitements de texte : on positionne ses balises à la main. Ce n'est certes pas très ergonomique par rapport à ce qui se fait maintenant, mais il est possible de masquer/montrer les divers codes de contrôle ce qui permet d'avoir une prévisualisation très proche de ce que donnera l'impression. La plupart des autres traitements de texte (comme Textomat) imposent de faire un "aperçu avant impression" dans ce cas. En revanche, comme c'est d'ailleurs le cas pour tous les autres traitements de textes CPC, Protext n'est absolument pas "What You See Is What You Get"...

La modularité de Protext

Comme le fait si justement remarquer le petit blond à lunettes, on peut conclure en disant que Protext est un excellent éditeur de texte (sans doute le meilleur sur CPC) et un bon traitement de texte (mais toutefois moins puissant que Textomat). Mais Protext a une botte secrète : ses plugins ! En effet, Protext est modulaire et il existe un bon nombre d'add-ons qui lui ajoutent diverses fonctionnalités. Certains sont également en ROM alors que d'autres sont simplement des drivers disque.

Promerge : La base de départ

Il s'agit là d'une ROM d'Arnor qui ajoute diverses fonctions à Protext. En premier lieu elle améliore le mode commande en y ajoutant la copie en ligne (comme sous Basic avec shift/flèches/copy) ainsi que diverses commandes comme SPLIT (pour découper un fichier sur disque en plusieurs morceaux), CALC (pour effectuer des calculs en ligne), etc. Ensuite, elle exploite les 64ko supplémentaires des 6128 (ou 464/664 étendus) pour y loger un deuxième document ; il est alors possible d'éditer deux sources/textes en même temps sous réserve de ne pas utiliser les 64ko de bank additionnels. Enfin, elle ajoute au traitement de textes Protext les fonctionnalités d'impression circulaire telles qu'elles sont disponibles sous Textomat.

Prospell : L'orthographe

Cette ROM (qui tout comme Protext et Promerge existe aussi en version disquette) n'a d'intérêt que pour la fonction traitement de texte de Protext. Comme son nom l'indique, elle ajoute les fonctions de correction orthographique via un dictionnaire configurable qui devra être stocké sur disquette. En pratique l'usage de ce dictionnaire est assez similaire à celui de Semword.

Protype et Pro-Ext

Il ne s'agit plus ici d'extensions de Protext disponibles sous forme de ROM, mais exclusivement sur format disque. Ces modules sont d'ailleurs plus spécifiquement destinés à la version disque de Protext quoiqu'il soit possible de les utiliser avec la version ROM. Protype ajoute la gestion des polices graphiques au mode traitement de texte de Protext ; ce qui permet des impressions beaucoup plus élaborées. Quant à elle, Pro-ext est une solution encore plus aboutie puisqu'en plus de permettre l'utilisation de polices graphiques, elle permet l'insertion d'images dans les documents Protext ! Ceci est assez unique sur CPC et fait de Protext un outil extrêmement puissant. De plus, ce modules intègre un éditeur de polices graphiques et d'images.

Destext : Protext sous DES

La place va ne manquer pour vous parler de cette extension qui permet ni plus ni moins que d'utiliser Protext dans l'environnement graphique DES ! DES (qui tient sur deux ROMs) est un environnement graphique pour CPC. En pratique son intérêt est assez limité (rien ne vaut un bon shell pour un maximum d'efficacité) mais c'est un outil ludique assez impressionnant que je vous conseille de tester. Voilà, nous en avons fini avec cette rapide présentation des principales ROM pour CPC.

OffseT

Futurs' freeware diffusion

Garanti sans ajout de Tocard. 100% pur troll.