P. Lavigne

Arrivés à ce stade de la publication de l'article-fleuve consacré à la carte graphique, nous sommes confrontés à un cruel dilemne: par quoi continuer, le logiciel ou l'extension couleurs?
Et bien voilà, notre choix est fait; ce sera le logiciel. Ce qui devrait permettre à nos lecteurs enthousiastes de profiter dès maintenant de leur carte en Noir&Blanc... la couleur suivra.

le logiciel pour la carte graphique

3ème partie

4 Koctets de code objet 6502

Note: Constructeurs de la carte principale, veuillez jeter un coup d'oeil au bas de la page 11-49. Gardez votre fer au chaud! Le mois prochain on attaque la couleur...

Le système graphique haute résolution en couleurs proposé par Elektor se décompose en trois parties: la carte principale avec le processeur graphique de Thomson qui a fait l'objet des articles publiés en septembre et en octobre, la carte d'extension couleurs qui sera décrite le mois prochain, et le logiciel décrit ci-après. Ce logiciel est le même, avec ou sans couleurs; il n'y a rien à y changer lorsque l'on passe du Noir&Blanc à la couleur; de même qu'il n'y a rien à changer sur la carte principale lorsque l'on rajoute l'extension couleurs. Nous ne l'aborderons pas ici sous l'angle de la programmation en langage assembleur — ce qui nous emmènerait trop loin, compte tenu de l'ampleur du listing source — mais sous l'angle de l'utilisation pratique des instructions disponibles, avec un langage évolué comme par exemple le BASIC.

Terminal vidéo et/ou terminal graphique

La carte graphique se présente, de par son aspect matériel, comme un terminal de visualisation au même titre que la carte VDU ou l'Elekterminal. Le logiciel que nous allons décrire lui permet de fonctionner en terminal vidéo alphanumérique (l'écran est organisé en 32 lignes de 80 caractères) et/ou comme terminal de visualisation graphique, avec un écran de 512 × 256 ou 512 × 512 pixels et 16 couleurs. Il est bien écrit "et/ou", c'est-à-dire que l'on peut garder le terminal alphanumérique dont on dispose déjà sur son micro-ordinateur, et rajouter la carte graphique comme terminal graphique: mais on peut aussi supprimer le terminal vidéo dont on disposait jusqu'alors (carte VDU, Elekterminal et autres...) et utiliser la carte graphique à la fois comme terminal alphanumérique (pour les listings, etc) et comme terminal graphique. Bien entendu, au niveau du matériel, rien ne change; c'est le logiciel qui permet de passer d'un mode à l'autre.
Pour bien saisir la fonction de ce logiciel, on peut l'imaginer comme celui d'une imprimante ou d'une table traçante; il reçoit des codes, les interprète comme étant soit des caractères alphanumériques, soit des instructions de tracé graphiques, et exécute les opérations nécessaires pour la visualisation des caractères ou des vecteurs et des points sur un écran. Pour cela, il commande en fait le GDP et les quelques registres auxiliaires qui se trouvent sur la carte graphique. La grande différence entre ce logiciel et celui d'une imprimante ou d'une table traçante est qu'il est exécuté par le microprocesseur du système sur lequel la carte est utilisée, alors qu'une imprimante dispose de son propre microprocesseur.
Après initialisation, le logiciel est toujours en mode "texte" (par opposition à mode "graphique"). C'est le mode dominant, celui d'où l'on vient et celui auquel on retourne, et ce pour des raisons de sécurité, lorsque le système est utilisé aussi comme terminal vidéo alphanumérique.
En mode "texte", l'écran est géré de haut en bas (l'origine du repère est en haut à gauche de l'écran) comme il est normal pour un terminal alphanumérique. Outre les fonctions *Carriage Return* (CR) et *Line Feed* (LF) automatiques en fin de ligne, ce mode connaît aussi les manipulations classiques du curseur et l'effacement (partiel ou total) de l'écran. Ces opérations sont toujours effectuées en tenant compte de la taille des caractères (puisque celle-ci est variable): lorsqu'elle est, par exemple, double de la taille normale, le LF sera lui aussi double du LF normal. Il en va de même pour l'échappement vertical (*scrolling*) lorsque le curseur est arrivé en bas de l'écran.
En mode "graphique", l'écran est géré du bas vers le haut. L'origine du repère cartésien est en bas à gauche de l'écran (X=Y=0). Dès lors, on ne raisonne plus en termes de caractères alphanumériques, de lignes et de colonnes, mais en termes de points, de pixels et de segments.
On distingue deux types d'accès au mode "graphique": l'un est définitif, l'autre est provisoire. Par accès définitif, on entend que l'on quitte le mode "texte" pour de bon, alors que par accès provisoire, on entend que toutes les instructions, jusqu'au CR suivant, sont interprétées et exécutées en mode "graphique". Avec le CR, on revient automatiquement en mode "texte".
La domination du mode "texte" sur le mode "graphique" est essentielle lorsque l'on utilise le système décrit ici à la fois comme terminal de visualisation alphanu-

mérique et comme terminal graphique. Dans ce cas, il est indispensable de revenir automatiquement au mode "texte" en cas d'interruption (*break*) d'un programme graphique. Considérant que le langage évolué le plus fréquemment utilisé avec cette carte serait le BASIC, il a fallu choisir une des lettres du mot BREAK, qui est imprimé automatiquement par l'interpréteur BASIC lors d'une interruption (BREAK IN LINE 120 par exemple), pour déclencher la procédure de retour du mode "graphique" au mode "texte". Comme nous le verrons ci-dessous lorsque nous passerons en revue les instructions du mode graphique, c'est la lettre A du mot BREAK qui a été retenue.

Complet et complexe

Nous disposons donc d'un système à deux fonctions bien distinctes l'une de l'autre: la fonction "texte" et la fonction "graphique". Nous verrons cependant au cours de cet article que les deux fonctions peuvent interagir dans une large mesure pour afficher, par exemple, du texte au beau milieu d'un dessin, et inversement.
Du fait que l'on peut mettre en oeuvre la carte graphique et son logiciel comme terminal alphanumérique, ce logiciel comporte également les routines de saisie de caractères via le clavier, avec la gestion du curseur. De sorte que, finalement, nous sommes en présence de trois sous-ensembles:

1. la visualisation de codes ASCII sur un écran sous la forme de 80 colonnes et 32 lignes de caractères alphanumériques.
2. la saisie de codes ASCII émis par le clavier et la gestion du curseur clignotant (vitesse de clignotement programmable)
3. la gestion de l'écran graphique point par point, à partir d'instructions spécifiques.

Les deux premiers sous-ensembles appartiennent au mode "texte", le troisième au mode "graphique". On peut les utiliser indépendamment les uns des autres, sans contrainte ni restriction.
Pour l'essentiel, ce logiciel se présente donc comme une super routine d'impression appelée CHROUT (pour *character output*). Elle reçoit tous les codes ASCII destinés à la carte graphique, sans distinction de mode, et se charge elle-même d'afficher les caractères alphanumériques lorsque l'on est en mode "texte" et d'exécuter les instructions graphiques correspondantes lorsque l'on est en mode "graphique" (**figure 1**).
Accessoirement, si l'on utilise aussi la carte comme terminal de visualisation autonome, on dispose également de la routine de saisie: CHRINP (pour *character input*). CHROUT et CHRINP sont les deux

1

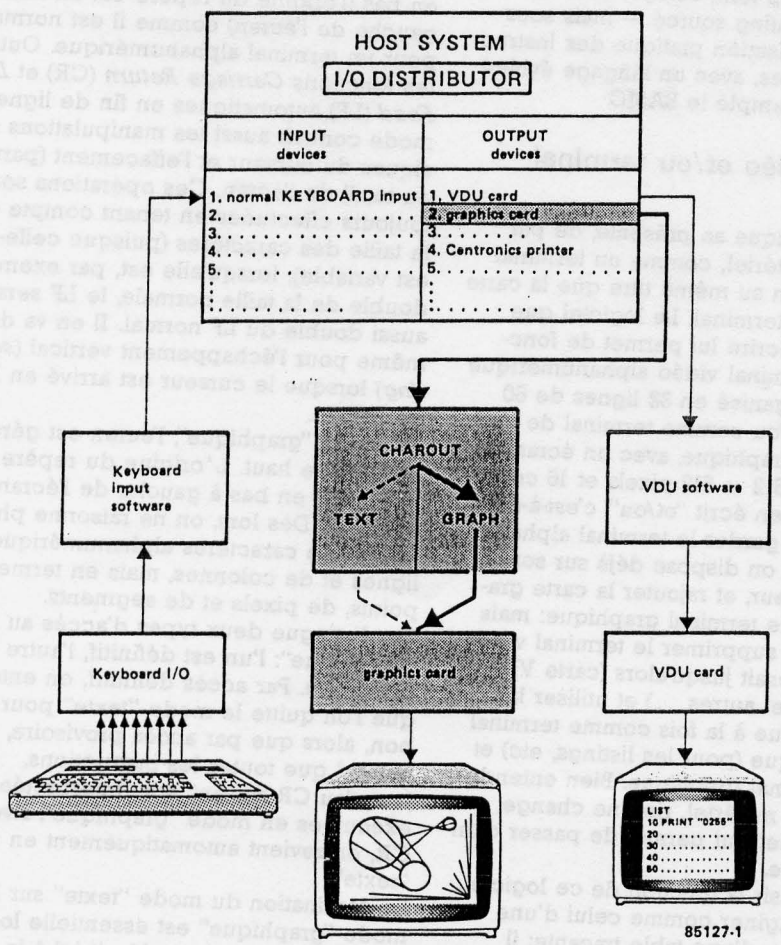

Figure 1. Il est fort aisé d'insérer la carte graphique et son logiciel pour CPU 6502 dans un système muni d'un distributeur d'entrées/sorties. Il est intéressant de garder le terminal alphanumérique existant, pour n'utiliser la carte graphique que comme terminal graphique. Pour ce faire, il suffit de modifier une seule adresse dans le système hôte; à savoir celle qui dans le distributeur de sorties donnera accès à la routine CHAROUT du logiciel graphique.

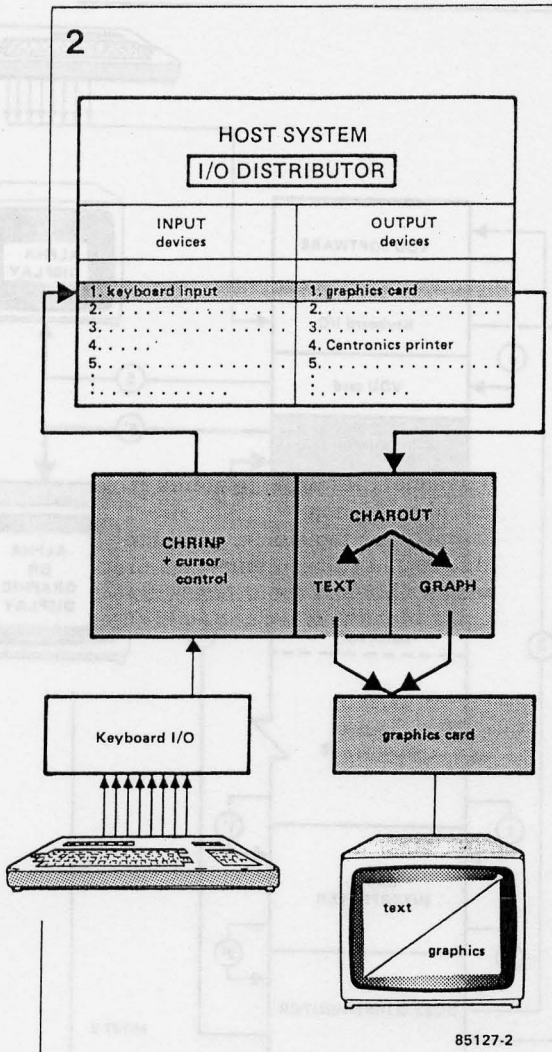

Figure 2. La carte graphique et son logiciel peuvent également être utilisés comme terminal autonome, à la fois alphanumérique et graphique. Dans ce cas, le logiciel graphique assure une triple fonction. Il suffit de modifier deux adresses sur le système hôte, l'une dans le distributeur de sorties (routine d'impression) et l'autre dans le distributeur d'entrées (réception des codes du clavier).

seules adresses à modifier sur le système hôte pour le faire communiquer avec le logiciel de la carte graphique. L'idéal étant bien sûr de disposer à cet effet d'un distributeur d'entrées/sorties dans lequel on peut placer ces deux adresses (**figure 2**). Par ailleurs, le logiciel graphique lui-même doit être adapté à la procédure d'interruption en vigueur dans le système hôte (BREAK). Ceci n'est pas obligatoire... et nous y reviendrons. Un dernier détail important dans ce survol concerne l'initialisation du logiciel, et à travers lui, du matériel. Il y a pour cela une routine autonome, appelée INITAL (pour *initialize all*).

En résumé, nous avons donc, en plus des deux modules déjà mentionnés (CHROUT et CHRINP), deux modules accessoires, l'un pour les interruptions (BRKTST) et l'autre pour l'initialisation (INITAL).
Dans son état actuel, le logiciel occupe moins de 4 Koctets de mémoire, ce qui permet de le caser dans une EPROM du type 2732. Cependant, il ne faut pas oublier qu'un logiciel de cette importance travaille avec un certain nombre de cellules de mémoire vive pour les paramètres variables, les pointeurs et autres tampons. De sorte qu'en **plus des 4 K occupés par l'EPROM**, il faudra réserver environ **une quarantaine d'adresses de mémoire vive** (heureusement pas en page zéro). Alors que lorsque le logiciel est casé lui-même en mémoire vive, il reste dans la partie supérieure du bloc de 4K qu'il occupe, assez d'adresses libres pour ces pointeurs. Ainsi par exemple, lorsque le logiciel est placé en $C000_{HEX}$ ou $D000_{HEX}$, les pointeurs occupent l'extrémité de ce bloc à partir de $CF80_{HEX}$ ou $DF80_{HEX}$ sans gêner personne. Mais lorsque ce même bloc se trouve en mémoire morte à ces adresses, les pointeurs ne peuvent plus occuper le haut du bloc, et doivent être placés en $BF80_{HEX}$ par exemple, à supposer bien entendu qu'il s'agisse là de mémoire vive (**figure 3**).
En voici assez pour les préliminaires. Passons aux instructions. Celles-ci avaient été présentées dans le premier article publié en Septembre 1985 (Elektor n°87, page 9-53, tableau 1). On les retrouve (version revue et corrigée) sur les infocartes de ce mois-ci. Nous vous conseillons de les détacher, et de les garder à portée de main au cours de la lecture de la suite du présent article, où pour des raisons d'économie de place, nous ne les avons plus reproduites.

Les instructions en mode "texte"

Les instructions propres au mode "texte" ne comportent jamais qu'un seul octet (pas de paramètres); ce sont les codes ASCII inférieurs à 20_{HEX}. Elles sont toujours données directement sous la forme du code hexadécimal. Par exemple, 08_{HEX} = curseur vers la gauche (*backspace*) ou encore $0D_{HEX}$ = retour chariot (*carriage return*). Depuis le BASIC elles peuvent être données sous la forme CHR$(8) ou CHR$(13). C'est d'ailleurs sous cette forme que nous les rencontrerons toujours avec l'instruction PRINT.
Les instructions CHR$(8)...CHR$(13) et CHR$(26)...CHR$(29) ne méritent aucun commentaire particulier, puisque ce sont des commandes vidéo classiques. Elles ont tout de même ceci de particulier qu'elles doivent tenir compte de la taille variable des caractères. Et elles le font.
Viennent ensuite quelques instructions spécifiques à notre système:
CHR$(17): passage en mode "texte".
CHR$(18): passage en mode "graphique".
Est-il nécessaire de préciser que ces deux instructions permettent de passer d'un mode à l'autre?

CHR$(20): retour à la taille minimale des caractères alphanumériques. Nous avons déjà indiqué que la taille des caractères était programmable à l'aide de deux facteurs d'agrandissement de la matrice de points originale (8 x 5), l'un pour l'axe horizontal et l'autre pour l'axe vertical (voir l'instruction S en mode "graphique"). Le code CHR$(20) ou 14_{HEX} permet un retour immédiat à la taille normale, quelle que soit la dernière taille en cours.

CHR$(4): tous les codes ASCII qui sont compris entre cette commande et un CR (*carriage return*) sont exécutés comme ins-

tructions en mode "graphique". On ne quitte pas le mode "texte" définitivement pour autant, puisque l'on y revient automatiquement avec le CR. L'instruction CHR$(4) est donc particulièrement pratique pour mélanger des dessins ou des changements de couleurs et de taille de caractères à un texte, un menu ou un listing.
Exemple: Supposons que nous sommes en mode "texte" et que nous affichons un menu en caractères deux fois plus larges que la taille normale, de couleur rouge. Imaginons qu'au bas du menu doit figurer la mention "Quel est votre choix?" en caractères de taille normale et de couleur bleue. La programmation de la taille des caractères et de leur couleur doit se faire en mode "graphique", mais l'affichage des caractères eux-mêmes en mode "texte". De toute évidence nous ne quitterons le mode "texte" que provisoirement, et utiliserons par conséquent l'instruction CHR$(4).

```
10 REM couleur rouge (=6)
20 REM double largeur
30 REM hauteur normale
40 PRINT CHR$(4)"C6,S2,1"
50 REM retour automatique
60 REM au mode texte
70 PRINT "MENU...
.
.
.
90 REM fin du menu
100 REM couleur bleue (=3)
110 REM taille normale
120 PRINT CHR$(4)"C3,S1,1"
130 REM retour automatique
140 REM au mode texte
150 PRINT "Quel est votre choix?"
```

Une autre manière d'écrire les lignes 120 et 150 aurait été:

```
120 PRINT CHR$(4)"C3"
150 PRINT CHR$(20)"Votre choix?"
```

Si l'on essaie de se passer de l'instruction CHR$(4) pour obtenir la même chose uniquement à l'aide de CHR$(18) et CHR$(17), on s'aperçoit qu'il faut sensiblement plus d'instructions, et c'est moins élégant... Avec les deux dernières instructions du mode "texte", nous en abordons la partie la plus difficile à comprendre pour le néophyte.

CHR$(1) et CHR$(2): ouverture, fermeture et répétition d'une zone de mémoire tampon qui fait office de miroir de la mémoire d'image (*video buffer*). Ceci mérite quelques explications.
Nous savons qu'en N & B, la mémoire d'image s'étend sur 16 Koctets; avec 8 couleurs (RGB), ce sont 48 K et avec 16 couleurs (RGBI) c'en sont 64...! Fort heureusement, ces 16, 48 ou 64 K ne sont pas prélevés dans la mémoire disponible sur le système hôte: la carte graphique et son extension couleurs possèdent leur mémoire autonome. Mais qu'arrive-t-il lorsque l'on veut transférer le contenu de la mémoire vidéo vers la mémoire normale, et de là sur une disquette? Si l'on fait un relevé pixel par pixel de la mémoire vidéo, ça fait beaucoup...voire trop si l'on considère qu'une image, même la plus simple, en 16 couleurs, occupera toujours et irrémédiablement 64 Koctets. Il y a là une disproportion insupportable dans le rapport contenant/contenu!
En examinant de plus près la place que prennent les instructions qui ont servi à générer une image quelconque, on s'aperçoit que c'est sensiblement moins encombrant de sauvegarder les instructions elles-mêmes plutôt que leur résultat sur l'écran et dans la mémoire d'image. Surtout grâce à la syntaxe particulièrement compacte que nous connaissons ici. Si de surcroît on prend soin d'éliminer les codes non significatifs, comme les espaces et les LF *—line feed—* on obtient un rendement optimal. Il faut déjà qu'une image soit bigrement complexe, même en 16 couleurs, pour que les instructions qui ont permis de la générer, occupent, lorsqu'elles sont mises bout à bout, plus de 16 ou 48 K; sans parler des 64 K, ni surtout d'images à résolution verticale de 512 lignes!
En tout état de cause, **les instructions mises bout à bout occuperont toujours moins de place que les images qu'elles génèrent, et l'espace mémoire requis pour leur sauvegarde est proportionnel à la complexité de l'image.** Une fois les instructions sauvegardées dans un tampon

3

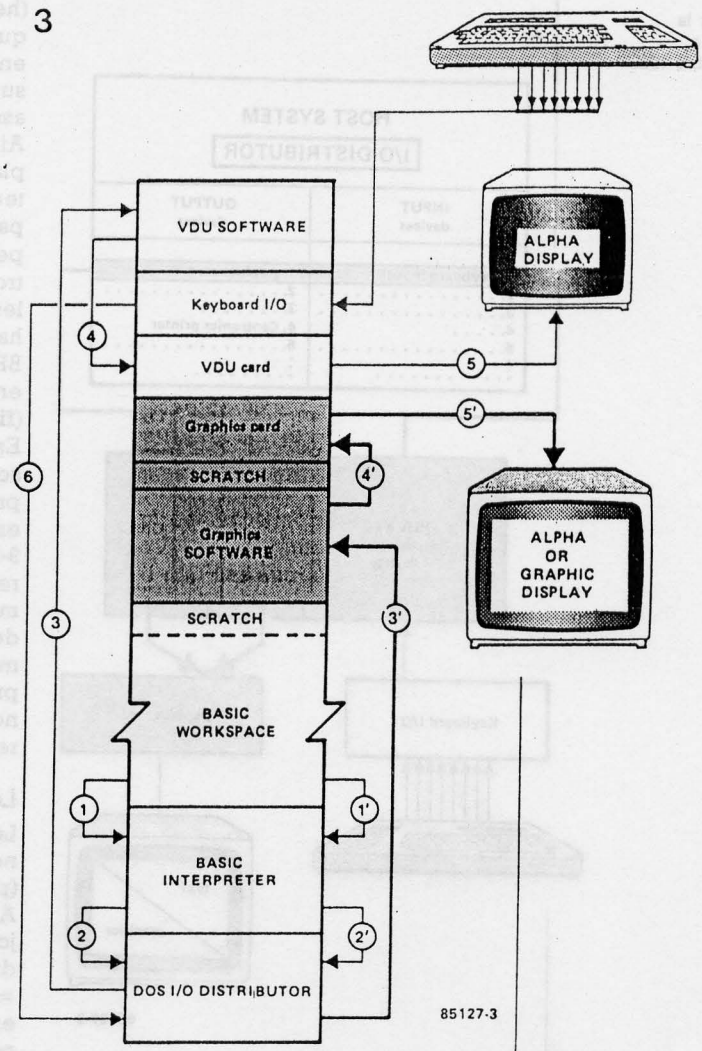

Figure 3. Voici la cartographie de la mémoire d'un système où cohabitent la carte graphique (+ son logiciel) et le terminal alphanumérique d'origine (ici une carte VDU (+ son logiciel). Les commandes graphiques imprimées (1 et 1') par un programme en BASIC (*workspace*) sont envoyées, par l'interpréteur BASIC (2 et 2') via le distributeur d'entrées/sorties, simultanément vers le logiciel VDU (3) et vers le logiciel graphique (3'). Selon la configuration du distributeur d'entrées, les codes en provenance du clavier (6) sont acheminés soit vers le logiciel VDU, soit vers le logiciel graphique (CHARINP sur la figure 2).

La carte graphique et le Junior Computer

Lorsqu'elle est utilisée sur le Junior Computer avec DOS, la carte graphique (et son logiciel) fait office de terminal graphique essentiellement, puisque la carte VDU remplit son office de terminal alphanumérique (figure 1). Rien ne vous empêche cependant de l'utiliser en mode "texte".

La routine de saisie CHRINP peut éventuellement être utilisée par curiosité, mais cela ne s'impose pas (notez bien qu'avec le DOS du J.C. on peut utiliser **simultanément** plusieurs appareils en sortie, mais jamais qu'un seul appareil en entrée (normalement le clavier). La routine de test pour les interruptions BREAK a été adaptée au Junior Computer dans le listing source du logiciel pour la carte graphique. Il ne reste donc que deux adresses à modifier sur le J.C.; on placera l'adresse de la routine CHROUT dans le distributeur de sortie en deuxième ou troisième place, entre l'adresse de la routine de visualisation sur la carte VDU et l'adresse de la routine pour l'imprimante Centronics.

Attention! Les adresses à placer dans ce distributeur sont toujours l'adresse réelle moins une. Par exemple, pour CHROUT qui est assemblé en $B003_{HEX}$, l'octet à placer en 2313_{HEX} ou 2315_{HEX} est 02, et non 03:

2313 : 02 } "IO, 02" ou "IO ,03"
2314 : B0 }

ou

2315 : 02 } "IO ,04" ou "IO ,05"
2316 : B0 }

Pour la routine de saisie CHRINP, c'est l'adresse $B005_{HEX}$ (= B006 — 1) qu'il faut placer en 2311 et 2312. L'adresse de la routine INITAL est $B000_{HEX}$. On y accède par la commande DISK!"GO B000" que l'on aura soin de donner au début de chaque programme de dessin pour remettre à zéro tous les paramètres. On aura compris, à la lecture de ces adresses, que le logiciel graphique a son code objet assemblé en mémoire vive en $B000_{HEX}$. Ce bloc de 4K est prélevé sur la mémoire vive et l'espace de travail de l'interpréteur BASIC; il faut donc modifier la piste 0 de la version V3.3 en conséquence. Effectuer la procédure de chargement de la piste 0 à l'adresse $A200 comme indiqué en septembre 1983, Elektor n°63, page 9-63, et remplacer dans le tableau 2 de cette page **l'octet BF en $A218_{HEX}$ par l'octet AF.** Remettre sur disquette la piste 0 modifiée. La modification du distributeur d'entrées/sorties doit être effectuée sur la piste 1. C'est tout!

Le tampon de sauvegarde des instructions *(video buffer)* s'étend de 7000_{HEX} à $AFFF_{HEX}$, soit 16 K; il reste 12 K pour le programme BASIC à partir de $3A79_{HEX}$. C'est largement assez.

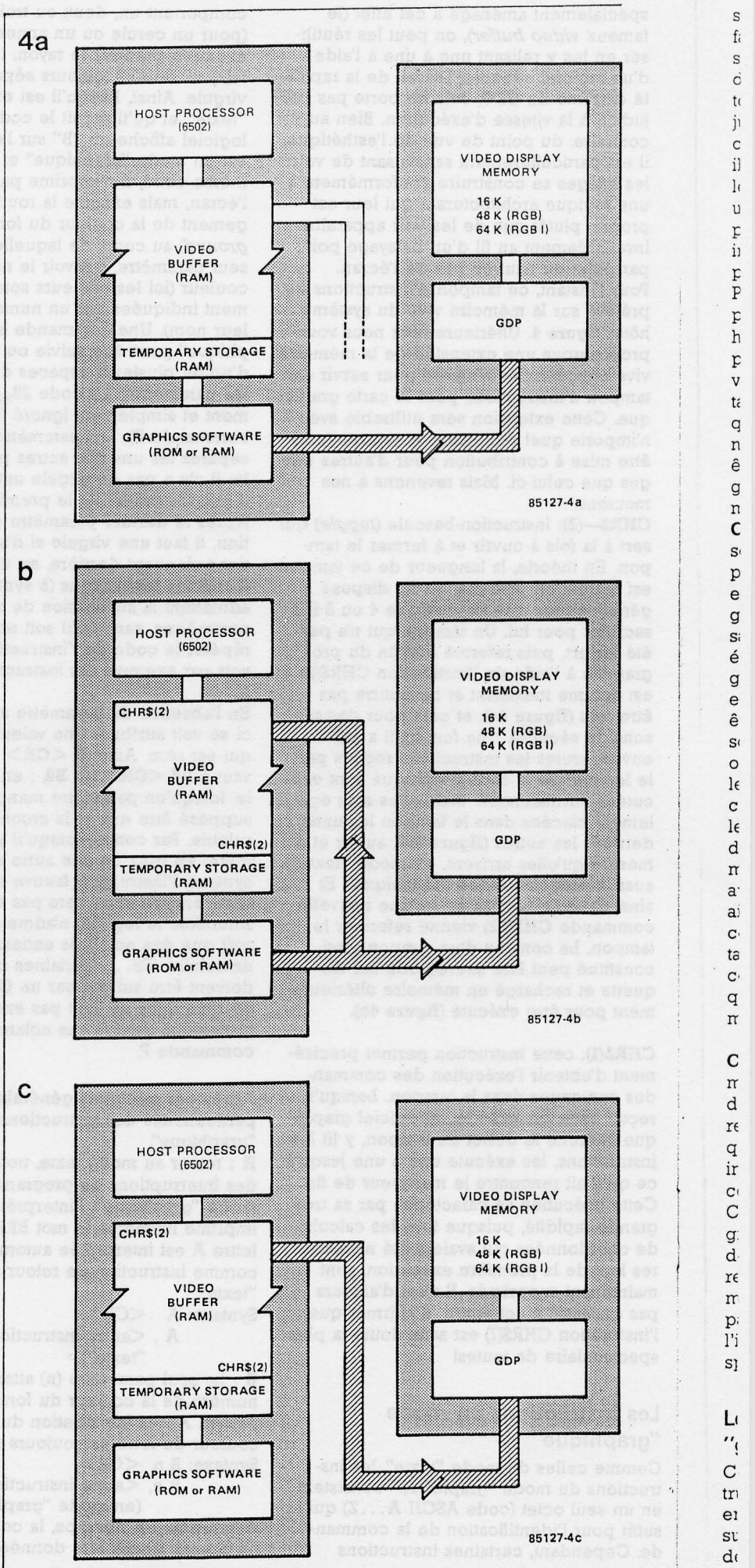

Figure 4. Sur la figure 4a, le logiciel graphique commande la carte graphique, mais ne sauvegarde pas les instructions exécutées. Sur la figure 4b, chaque instruction exécutée est également sauvegardée dans le *video buffer*. Sur la figure 4c, nous avons schématisé ce qui se produit lorsque le logiciel graphique reçoit l'instruction CHR$(1): il relit les commandes sauvegardées dans le tampon, et les exécute une à une.

spécialement aménagé à cet effet (le fameux *video buffer*), on peut les réutiliser en les y relisant une à une à l'aide d'un logiciel adéquat. Du fait de la rapidité extrême du GDP, cela ne porte pas préjudice à la vitesse d'exécution. Bien au contraire, du point de vue de l'esthétique, il est particulièrement satisfaisant de voir les images se construire conformément à une logique architecturale qui leur est propre, plutôt que de les voir apparaître invariablement au fil d'un balayage point par point du haut en bas de l'écran.
Pour l'instant, ce tampon d'instructions est prélevé sur la mémoire vive du système hôte (**figure 4**. Ultérieurement nous vous proposerons une extension de la mémoire vive conçue spécialement pour servir de tampon d'instructions pour la carte graphique. Cette extension sera utilisable avec n'importe quel ordinateur et pourra aussi être mise à contribution pour d'autres usages que celui-ci. Mais revenons à nos moutons.
CHR$—(2): instruction-bascule (*toggle*) qui sert à la fois à ouvrir et à fermer le tampon. En théorie, la longueur de ce tampon est infinie; en pratique, on ne dispose généralement que de quelque 4 ou 5 K à sacrifier pour lui. Un tampon qui n'a pas été ouvert, puis refermé à la fin du programme à l'aide de l'instruction CHR$(2) est comme inexistant et ne pourra pas être relu (**figure 4a**); et cela pour des raisons de sécurité. Une fois qu'il a été ouvert, toutes les instructions reçues par le logiciel de la carte graphique sont exécutées normalement, mais elles sont également placées dans le tampon les unes derrière les autres (**figure 4b**), au fur et à mesure qu'elles arrivent, en mode "texte" aussi bien qu'en mode "graphique". Et ainsi de suite jusqu'à ce qu'une nouvelle commande CHR$(2) vienne refermer le tampon. Le contenu d'un tampon ainsi constitué peut être sauvegardé sur disquette et rechargé en mémoire ultérieurement pour être exécuté (**figure 4c**).

CHR$(1): cette instruction permet précisément d'obtenir l'exécution des commandes contenues dans le tampon. Lorsqu'il reçoit cette commande, le logiciel graphique cherche le début du tampon, y lit les instructions, les exécute une à une jusqu'à ce qu'il ait rencontré le marqueur de fin. Cette exécution se caractérise par sa très grande rapidité, puisque tous les calculs de coordonnées qui avaient été nécessaires lors de la première exécution, sont maintenant supprimés. Il n'est d'ailleurs pas excessif, à cet égard, d'affirmer que l'instruction CHR$(1) est sans doute la plus spectaculaire de toutes!

Les instructions en mode "graphique"

Comme celles du mode "texte", les instructions du mode "graphique" consistent en un seul octet (code ASCII A...Z) qui suffit pour l'identification de la commande. Cependant, certaines instructions comportent un, deux ou trois paramètres (pour un cercle ou un anneau, il faut par exemple préciser le rayon, l'épaisseur et le(s) secteur(s)) toujours séparés par une virgule. Ainsi, lorsqu'il est en mode "texte", et qu'il reçoit le code 42_{HEX}, le logiciel affiche un "B" sur l'écran; lorsqu'il est en mode "graphique" et qu'il reçoit le même code, il n'imprime pas de B sur l'écran, mais exécute la routine de changement de la couleur du fond (*background*) au cours de laquelle il attend un seul paramètre, à savoir le numéro de la couleur (ici les couleurs sont effectivement indiquées par un numéro et pas par leur nom). Une commande en mode "graphique" peut être suivie ou précédée d'un ou plusieurs espaces de même que les paramètres. Le code 20_{HEX} est purement et simplement ignoré (sauf avec la commande P). Les paramètres doivent être séparés les uns des autres par une virgule; il n'y a pas de virgule entre le code d'une instruction et le premier paramètre. Après le dernier paramètre d'une instruction, il faut une virgule si d'autres instructions viennent derrière, ou un CR.
Certaines instructions (à syntaxe récursive) admettent la succession de séries de paramètres, sans qu'il soit nécessaire de répéter le code de l'instruction lui-même: voir par exemple les instructions D, J ou X.
En l'absence du paramètre attendu, celui-ci se voit attribuer une valeur par défaut, qui est zéro. Ainsi **B <CR>** ou **B,** équivaut à **B0 <CR>** ou **B0,** ; en règle générale, lorsqu'un paramètre manque, il est supposé être nul, et la commande est valable. Par contre, lorsqu'il y a un paramètre en trop, ou une autre erreur de syntaxe, l'instruction fautive est ignorée toute entière et ne sera pas exécutée. Attention: le logiciel n'admet et ne reconnaît que des nombres entiers; ainsi 15,6 deviendra 156... Certaines instructions doivent être suivies par un CR, à défaut de quoi elles ne sont pas exécutées correctement; c'est le cas notamment de la commande P.

Après ces quelques généralités, voici les particularités des instructions du mode "graphique".
A : retour au mode texte, notamment lors des interruptions de programmes en mode "graphique": l'interpréteur BASIC imprime lui-même le mot BREAK dont la lettre A est interprétée automatiquement comme instruction de retour au mode "texte".
Syntaxe: A <CR>
A , <autre instruction (en mode "texte")>
B : Le seul paramètre (n) attendu est le numéro de la couleur du fond ou du papier. Après initialisation du logiciel, la couleur du fond est toujours noire.
Syntaxe: B n <CR>
B n, <autre instruction> (en mode "graphique")
Remarque: En principe, la commande B ne devrait jamais être donnée en mode

RMW lorsque le fond n'est pas vierge. Essayez quand même, l'effet vous plaira peut-être...

C : Le seul paramètre (n) attendu est celui de la couleur de la plume. La syntaxe est comme celle de la commande B. Après initialisation du logiciel, la couleur de la plume est toujours blanche. La commande C peut être utilisée sans restriction en mode RMW.
Remarque: Le paramètre n de l'instruction B et de l'instruction C peut être positif (dans ce cas la couleur antérieure est ignorée) ou négatif (dans ce cas la nouvelle couleur est combinée à l'ancienne couleur par une opération logique AND).

D : Au moins deux paramètres sont attendus; ce sont les coordonnées **absolues** (et par conséquent toujours positives) de l'extrémité du segment à tracer; l'origine de ce segment est la position actuelle de la plume. Par coordonnées absolues, on entend qu'elles sont spécifiées par rapport à l'origine du repère cartésien (X=Ø ; Y=Ø).
Syntaxe: D x,y <CR>
D x,y, <autre instruction>
Syntaxe récursive:
D x,y,x,y...x,y <CR>
D x,y,x,y...x,y,
<autre instruction>

G : La lettre G est celle du mot "géométrie". Dans l'état actuel du logiciel, cette instruction donne accès à deux figures géométriques, selon la valeur du premier paramètre. Le signe de ce paramètre indique si la figure est remplie ou non. Les paramètres x et y donnent les dimensions de la figure en nombre de points à compter respectivement sur les axes horizontal et vertical, à partir de la position actuelle de la plume. Lorsque les paramètres x et/ou y sont négatifs, cela signifie que la figure sera tracée en arrière et/ou en-dessous de la position actuelle de la plume.
Syntaxe: G ± n,x,y <CR>
G ± n,x,y, <autre instruction>

H : Retour de la plume à la dernière origine définie (voir I) indépendamment de la position actuelle de la plume. Aucun paramètre n'est attendu.
Syntaxe: H <CR>
H, <autre instruction>

I : Les coordonnées actuelles de la plume deviennent le point d'origine. Aucun paramètre n'est attendu.
Syntaxe: I <CR>
I, <autre instruction>

J : Cette instruction est l'équivalent de la commande D, mais en coordonnées **relatives**: au moins deux paramètres sont attendus qui sont les coordonnées relatives de l'extrémité du segment à tracer; l'origine de ce segment est la position actuelle de la plume. Par coordonnées relatives, on entend que les coordonnées sont spécifiées par rapport à la position actuelle de la plume; dans ce cas, il est possible que ces coordonnées soient négatives lorsque l'extrémité du segment se trouve en arrière et/ou en-dessous de la position actuelle de la plume.
Syntaxe: voir la commande D

L : Le seul paramètre attendu définit le type de ligne utilisé par le GDP pour le tracé de vecteurs. Notez que le type de ligne exerce son influence non seulement sur les contours, mais également sur les surfaces pleines, comme les cercles, les carrés ou les rectangles et les triangles, ce qui produit des effets de matière fort intéressants.
Syntaxe: voir la commande C

M : Deux paramètres sont attendus qui sont les coordonnées **absolues** (donne toujours positives) du point vers lequel doit se déplacer la plume (sans dessiner!). Il n'est tenu aucun compte de la position antérieure de la plume.
Syntaxe: M x,y <CR>
M x,y, <autre instruction>
souvent: M,,I = M Ø,Ø,I
pour le retour de la plume en bas à gauche du repère cartésien

N : Trois paramètres sont attendus qui définissent les coordonnées **absolues** d'un point à dessiner (cf PLOT x,y) et la couleur(e) de ce point.
Syntaxe: N c,x,y <CR>
N c,x,y, <autre instruction>
Syntaxe récursive: voir la commande D

O : Tracé d'un cercle, d'un anneau, d'une portion ou segment d'anneau ou de cercle. La dernière origine définie — pas la position actuelle de la plume — tient lieu de centre. Selon le type de cercle ou d'anneau, la plume ne se retrouve pas forcément au centre à la fin du tracé.
Syntaxe: O n,r,t <CR>
O n,r,t, <autre instruction>
où n est le code pour la portion ou le secteur de cercle à tracer, r le rayon et t l'épaisseur de l'anneau.
Quand r=t, nous sommes en présence d'un disque (tracé le plus rapide!); quand r > t, nous obtenons un anneau d'épaisseur t: les apparences sont trompeuses: il y a dans ce cas moins de points à tracer, et pourtant le tracé est plus lent...

P : Cette instruction a, en mode graphique, une fonction comparable à celle de CHR$(4) en mode "texte". Tous les codes qui suivent l'instruction P en mode "graphique" sont imprimés en tant que caractères alphanumériques (donc comme si l'on était en mode "texte") à l'endroit où se trouve la plume
Syntaxe: P caractères <CR>
Exemple:

```
10 E = 12
20 PRINT "M 128,15,I"
30 PRINT "P Exemple numéro "; E
```

Le texte "Exemple numéro 12" est impri-

mé à partir du point X = 128, Y = 15 sans que l'on quitte le mode "graphique".
Notez que la commande P est, en mode "graphique", la seule qui tienne compte des espaces.

Q : Cette instruction est accompagnée d'un paramètre qui permet de passer de l'impression horizontale (normale) à l'impression verticale des caractères alphanumériques de l'instruction P.
Syntaxe: voir la commande C

R : Cette instruction est l'équivalent de l'instruction M (déplacement de la plume sans tracé) mais en coordonnées **relatives** à la position actuelle de la plume. De ce fait les coordonnées peuvent être négatives.
Syntaxe: voir la commande M

S : Les deux paramètres attendus sont un facteur d'agrandissement de la matrice de points des caractères alphanumériques respectivement sur l'axe horizontal et sur l'axe vertical. Ces deux facteurs sont spécifiés indépendamment l'un de l'autre, de sorte que l'on peut obtenir des caractères très larges mais de faible hauteur, ou des caractères très hauts mais de faible largeur. Les paramètres peuvent prendre toutes les valeurs comprises entre Ø et 15, sachant que S1,1 correspond à la taille normale.
Syntaxe: voir la commande M

T : Un paramètre est attendu qui permet de passer des caractères normaux aux caractères *italiques* indépendamment du sens d'impression.
Syntaxe: voir la commande C

U : Deux paramètres sont attendus; le premier permet de choisir entre la plume (tracer) et la gomme (effacer), le deuxième indique si la plume ou la gomme, selon le choix précédent, est haute ou basse. Cette instruction permet d'effectuer des tracés à l'aide d'algorithmes récursifs, dans lesquelles la plume (ou la gomme) est tantôt active (basse) tantôt inactive (haute).
Syntaxe: U p,u <CR>
U p,u, <autre instruction>
Remarque: Lors de l'initialisation, le GDP est toujours programmé "plume basse". Il n'est donc pas nécessaire de donner, au début de chaque programme de dessin, l'instruction "U 1,1" à condition que l'on ait pensé à exécuter la routine d'initialisation.

V : Deux paramètres sont attendus qui sont les coordonnées **absolues** d'un pixel à lire dans la mémoire d'image.
Syntaxe: voir la commande N
Les quatres bits de couleur correspondant au pixel deviennent les bits Ø...3 d'un tampon appelé PIXBUF (pour *pixel buffer*). Nous reviendrons sur l'adresse et le moyen d'accéder à PIXBUF.

W : Un seul paramètre est attendu qui permet de passer en mode RMW. Ce mode a été expliqué en long et en large dans le premier article consacré à la carte graphique.
Syntaxe: voir l'instruction T

X : Trois paramètres sont attendus pour effectuer le tracé d'un axe gradué. Cette instruction est récursive:
Syntaxe: X a,s,i <CR>
X a,s,i, <autre instruction>
Syntaxe récursive:
X a,s,i,...a,s,i <CR>
X a,s,i,...a,s,i,
<autre instruction>

Z : Un paramètre est attendu pour changer de page. Rappelons que la mémoire de la carte graphique et de son extension couleurs est organisée en deux ou quatre pages selon la résolution verticale choisie, entre lesquelles on peut passer sans difficulté ni perturbation du contenu de l'écran grâce à l'instruction Z.
Syntaxe: Z p <CR>
Z p, <autre instruction>
La valeur de p est comprise entre Ø (page 1) et 3 (page 4).

Paramètres et variables

Le passage en revue de toutes ces instructions montre leur concision. Il faut ajouter à cela qu'elles se prêtent on ne peut mieux à être utilisées avec un interpréteur BASIC quelconque, puisqu'elles sont imprimées comme chaîne de caractères. Ce qui implique bien entendu que **tous les paramètres peuvent être traités comme des variables**. Il en va de même pour le signe de ces variables lorsque cela présente un intérêt. Et dans les cas extrêmes, les instruction elles-mêmes peuvent être traitées et remplacées par des variables. Ce procédé permet d'obtenir des algorithmes récursifs, et par conséquent des programmes compacts, comme en témoignent les exemples ci-dessous. Il s'agit de deux programmes qui ont servi à tracer l'une des figures qui illustraient l'article du mois dernier (n° 88, page 10-47) et la figure "tubulaire" sur fond bleu de la couverture de ce même numéro.

```
10 DISK!"GO B000": DISK!"IO ,04"
20 PRINTCHR$(2)CHR$(18)"M256,128,I,W1"
30 FORI=0TO511: PRINT"D"I",255,H": NEXT
40 FORI=254TO1STEP-1: PRINT"D511,"I",H": NEXT
50 PRINT"R-1,-1,I"
60 FORI=511TO0STEP-1: PRINT"D"I",,H": NEXT
70 FORI=1TO254: PRINT"D,"I",H": NEXT: PRINTCHR$(2)
80 FORI=1TO20: PRINTCHR$(1): NEXT: DISK!"IO ,01"

10 DISK!"GO B000": DISK!"IO ,04": PRINTCHR$(18)"B6,C7,W1"
20 FORI=0TO180: GOSUB30: NEXT: DISK!"IO ,01": END
30 T=6.28*I/201: X=INT(255*(1-.8*COS(T)))
40 Y=INT(120*(1-.8*SIN(T)))
50 C=C+11: PRINT"M"X","Y",I,C"C",0255,40,40": RETURN
```

carte graphique: le logiciel (suite)

Trois kilos et demi d'octets pour dessiner, tracer, colorer...

P. Lavigne

Le logiciel pour la carte graphique existe, contrairement à ce que certains pourraient croire! Nous voulons pour preuve le vidage hexadécimal ci-contre. Bien sûr, un listing source serait préférable; mais vu le nombre de pages utilisées par la carte graphique dans les numéros de ces derniers mois, il est raisonnable de laisser la place à d'autres types d'articles.

Tel qu'il est présenté ici, le logiciel est utilisable par n'importe qui, sur n'importe quel système à 6502, et — moyennant quelques modifications faciles à faire — n'importe où dans la mémoire. Voici les quelques renseignements nécessaires à la mise en place:

- l'adresse de la routine d'initialisation du système est B000; l'adresse de routine CHROUT est B003; l'adresse de la routine de réception (facultative) est B006. Lorsque l'on appelle la routine CHROUT, le caractère doit se trouver dans l'accumulateur.
- la carte graphique est adressée en E150...E15F.
- la page zéro est utilisée, mais son contenu n'est pas modifié; la pile non plus. Le tampon vidéo s'étend de 6000 à AFFF.
- la zone de mémoire réservée comme brouillon pour le logiciel s'étend de BF80 à BFFF; ceci implique qu'il s'agit forcément de mémoire vive.
- Il n'est fait appel à des routines extérieures que pour le *Break Test*, et pour la réception d'un caractère du clavier si l'on fait appel à CHRINP. Pour le reste, le logiciel est autonome.

Voyons à présent comment s'y prendre le modifier "à la main". Si l'on décide de renoncer à la fonction BREAK et que l'on renonce aussi à se servir de ce logiciel pour la réception des caractères, et qu'on limite la carte graphique exclusivement à sa fonction de terminal graphique, c'est très simple: placer un RTS (60) en B009 et c'est tout. Si l'on désire garder la fonction BREAK, il faut placer en B00A et B00B l'adresse d'une routine qui détecte l'enfoncement de la touche BREAK et revienne avec dans l'accumulateur une donnée précise: ici cette donnée doit être 03 (CTL-C). Si on en préfère une autre, il suffit de mettre l'octet correspondant en B00D. Il faut ensuite mettre en B016 et B017 l'adresse à laquelle le logiciel doit aller lorsque la touche BREAK a été actionnée (*break handler* = interruption définitive du programme en cours d'exécution).

Attention! n'oubliez pas d'inverser l'octet de poids fort et l'octet de poids faible de ces adresses lorsque vous les placerez aux endroits indiqués.

Si vous désirez aussi utiliser la routine de réception des caractères et de gestion du curseur pour la carte graphique, c'est un peu plus compliqué. Il faut placer en B029 et B02A l'adresse d'une routine de scrutation du clavier: on ne doit ressortir de cette routine que lorsqu'une touche a été actionnée; le code correspondant à la touche doit se trouver dans l'accumulateur. En B023 et B024 par contre, il faut placer l'adresse d'une routine qui n'attend pas qu'une touche soit actionnée: sa fonction est de charger le code proposé par le clavier, c'est tout. La détection (*keyboard strobe scan*) a été faite par le logiciel graphique lui-même: les instructions qui assurent cette fonction de détection de l'impulsion d'échantillonnage du clavier se trouvent de B018 à B01F (inclus).

Toutes les adresses concernées apparaissent en grisé dans le vidage. Les octets soulignés sont tous ceux qu'il convient de modifier si vous désirez utiliser le logiciel ailleurs qu'en B000...BFFF, et si vous décodez la carte graphique ailleurs qu'en E150...E16F. Supposons que vous décodiez la carte graphique en D4XX; il suffit alors de remplacer tous les octets E1 soulignés par un octet D4. Si vous désirez mettre le logiciel en EPROM, il faut déplacer la zone mémoire brouillon de BF80 vers une autre adresse où se trouve de la mémoire vive; par exemple D180...D1FF. Remplacer par conséquent tous les octets BF soulignés par un octet D1. Si vous désirez reloger l'ensemble du logiciel en page A000...AFFF (mémoire vive), remplacer tous les octets soulignés BX par un octet AX. Et enfin, si vous désirez changer les limites de la zone de mémoire réservée au tampon vidéo (6000...AFFF), modifiez en conséquence les octets 60 et AF soulignés. Remplacer par exemple 60 par 70 et AF par 9F: le tampon sera compris entre 7000 et 9FFF. Tout cela n'est pas bien compliqué en fin de compte. Et s'il vous reste des questions, n'hésitez pas à nous écrire; mais s'il vous plaît, soyez clairs, précis et concis. Nous ne pouvons pas nous engager à répondre à chacun individuellement, mais nous nous efforcerons de faire une synthèse de toutes les questions pour formuler une réponse générale.

Une histoire de nanosecondes

Nous avons constaté, maintenant qu'un grand nombre de cartes graphiques ont été construites, que toutes fonctionnaient comme prévu, à ceci près que certaines d'entre elles présentent des défauts de *timing* plus ou moins importants au niveau de IC10. Certaines PROM (il est difficile, voire impossible de déterminer à l'avance lesquelles) se révèlent trop lentes: les signaux RAS individuels n'arrivent plus au bon moment, ce qui se traduit par une instabilité ou un dédoublement de l'image, par ailleurs tout-à-fait normale. Ce défaut peut être latent sur la carte principale et n'apparaître vraiment que lors de la mise en place de l'extension; mais il peut tout aussi bien se manifester déjà sur la carte principale seule.

Un test simple et efficace consiste à appuyer un doigt (sec) sur les connexions des circuits de RAM, côté soudures. Attention aux bagues et autres chevalières! Quand le *timing* est bon, cette manoeuvre ne parvient pas à perturber le contenu de l'écran (ne pas faire ce test pendant que le GDP dessine). Si par contre l'image se détériore, c'est que la PROM produit des signaux insatisfaisants. Il y a trois remèdes possibles. Le plus simple consiste à programmer pour IC10 une autre PROM issue

d'un lot différent de celui de la première (l'année et la semaine au cours de laquelle les circuits intégrés sont fabriqués figurent sur le boîtier — par exemple 8526 indique que le circuit a été fabriqué au cours de la 26ème semaine de l'année 1985). Nous vous rappelons que dans le numéro 27 d'Elektor, septembre 1980, page 9-19, vous trouverez un programmateur de PROM parfaitement indiqué pour cela.

Les deux autres remèdes demandent une intervention sur la carte. Dans certains cas, il suffira de mettre une résistance de polarisation au niveau logique haut de 470 ohms sur la ligne MUX (par exemple entre les broches 7 et 16 d'IC8). Dans le même ordre d'idées, on peut faire passer le signal MUX par la porte N12 restée libre dans IC27, de façon à retarder ce signal avant de l'appliquer à IC8 et IC9. Ceci implique évidemment un peu de charcutage... c'est le terme qu'emploie un lecteur, Monsieur S. Lichtenberger (Hautot/Mer) qui a adopté cette solution; nous en profitons pour le remercier ici de sa contribution. Pour notre part, nous préférons cependant la solution de remplacement de la PROM, plus élégante et moins charcutière.

Pour vous faciliter le travail, nous avons souligné tous les octets à modifier dans le code objet ci-dessous. Les apparences sont trompeuses: ce n'est pas difficile.

	Ø	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
BØØØ:	4C	2E	BØ	4C	5C	B4	4C	9C	B3	2Ø	8F	F7	C9	Ø3	DØ	ØF
BØ1Ø:	A9	11	2Ø	73	B1	4C	19	Ø8	AD	ØD	E1	29	Ø2	DØ	Ø1	6Ø
BØ2Ø:	68	68	2Ø	24	F7	4C	11	B4	2Ø	1D	F7	4C	11	B4	A2	Ø3
BØ3Ø:	8E	66	E1	8E	93	BF	A9	Ø7	8D	5Ø	E1	2Ø	8D	BØ	CA	1Ø
BØ4Ø:	EF	A9	ØØ	8D	64	E1	8D	91	BF	8D	94	BF	8D	95	BF	8D
BØ5Ø:	98	BF	A2	Ø3	9D	85	BF	CA	1Ø	FA	8E	96	BF	8E	ØØ	6Ø
BØ6Ø:	A9	6Ø	8D	99	BF	A9	ØØ	8D	65	E1	8D	67	E1	8D	92	BF
BØ7Ø:	8D	9Ø	BF	A9	48	8D	8E	BF	A9	ØF	8D	97	BF	A9	Ø7	8D
BØ8Ø:	5Ø	E1	2Ø	8D	BØ	2Ø	Ø7	B1	A9	ØB	8D	51	E1	A9	Ø4	2D
BØ9Ø:	5Ø	E1	FØ	F9	6Ø	AD	53	E1	29	FØ	18	DØ	Ø1	38	6A	4A
BØAØ:	8D	83	BF	4A	18	6D	83	BF	8D	83	BF	6Ø	AD	53	E1	29
BØBØ:	ØF	DØ	Ø2	A9	1Ø	ØA	ØA	ØA	8D	83	BF	6Ø	A9	ØØ	8D	59
BØCØ:	E1	8D	58	E1	8D	9Ø	BF	2Ø	AC	BØ	AD	92	BF	18	6D	5B
BØDØ:	E1	CD	83	BF	9Ø	ØC	AD	5B	E1	38	ED	83	BF	8D	5B	E1
BØEØ:	38	6Ø	2Ø	D6	BØ	AD	92	BF	FØ	Ø5	AD	83	BF	DØ	Ø6	A9
BØFØ:	ØØ	38	ED	5B	E1	18	6D	92	BF	8D	92	BF	38	A9	ØØ	ED
B1ØØ:	92	BF	8D	65	E1	18	6Ø	2Ø	AC	BØ	A9	ØØ	8D	92	BF	8D
B11Ø:	65	E1	8D	59	E1	8D	58	E1	8D	9Ø	BF	38	ED	83	BF	8D
B12Ø:	5B	E1	6Ø	2Ø	95	BØ	18	6D	59	E1	AD	58	E1	69	ØØ	C9
B13Ø:	Ø2	6Ø	48	AD	59	E1	48	AD	58	E1	48	A9	Ø1	8D	5Ø	E1
B14Ø:	AD	93	BF	29	7F	8D	66	E1	2Ø	8D	BØ	A9	ØA	8D	5Ø	E1
B15Ø:	2Ø	8D	BØ	68	8D	58	E1	68	8D	59	E1	A9	ØØ	8D	5Ø	E1
B16Ø:	2Ø	8D	BØ	68	8D	5Ø	E1	AD	93	BF	8D	66	E1	EE	9Ø	BF
B17Ø:	4C	8D	BØ	AC	94	BF	FØ	Ø7	C9	Ø4	9Ø	Ø3	4C	ØA	B5	C9
B18Ø:	2Ø	9Ø	14	C9	8Ø	9Ø	Ø3	6Ø	FF	FF	48	2Ø	23	B1	9Ø	Ø3
B19Ø:	2Ø	BC	BØ	68	4C	32	B1	ØA	AA	BD	5C	B3	C9	FF	Ø8	8D
B1AØ:	9C	BF	BD	5D	B3	8D	9D	BF	C9	FF	DØ	Ø4	28	DØ	Ø2	6Ø
B1BØ:	28	6C	9C	BF	2Ø	95	BØ	CE	9Ø	BF	AD	59	E1	38	ED	83
B1CØ:	BF	8D	59	E1	AD	58	E1	E9	ØØ	8D	58	E1	9Ø	Ø1	6Ø	A9
B1DØ:	ØØ	8D	59	E1	8D	58	E1	8D	9Ø	BF	48	68	AD	59	E1	48
B1EØ:	18	6D	83	BF	8D	59	E1	AD	58	E1	69	ØØ	8D	58	E1	EE
B1FØ:	9Ø	BF	2Ø	23	B1	9Ø	E4	68	8D	59	E1	CE	9Ø	BF	2Ø	AC
B2ØØ:	BØ	18	6D	5B	E1	8D	5B	E1	6Ø	FF	FF	2Ø	23	B1	9Ø	Ø3
B21Ø:	2Ø	BC	BØ	2Ø	95	BØ	18	6D	59	E1	8D	59	E1	AD	58	E1
B22Ø:	69	ØØ	8D	58	E1	EE	9Ø	BF	6Ø	2Ø	C7	BØ	9Ø	ØA	AD	58
B23Ø:	E1	DØ	F5	AD	59	E1	DØ	FØ	4C	C1	B2	AD	53	E1	48	2Ø
B24Ø:	65	BØ	68	8D	53	E1	4C	Ø7	B1	AD	58	E1	DØ	Ø8	AD	59
B25Ø:	E1	DØ	Ø3	6Ø	FF	FF	A9	Ø1	8D	51	E1	2Ø	AC	BØ	18	6D
B26Ø:	5B	E1	8D	5B	E1	AD	58	E1	48	AD	59	E1	48	A9	FF	38
B27Ø:	ED	59	E1	8D	55	E1	A9	Ø1	ED	58	E1	4A	6E	55	E1	8A
B28Ø:	48	CE	5B	E1	2Ø	A1	B2	CE	83	BF	DØ	F5	A9	ØØ	8D	59
B29Ø:	E1	8D	58	E1	8D	9Ø	BF	A9	ØB	8D	51	E1	68	68	68	AA
B2AØ:	6Ø	BA	BD	Ø5	Ø1	8D	58	E1	BD	Ø4	Ø1	8D	59	E1	2Ø	B9
B2BØ:	B2	EE	59	E1	DØ	Ø3	EE	58	E1	A9	1Ø	8D	5Ø	E1	4C	8D
B2CØ:	BØ	AD	59	E1	48	AD	58	E1	48	AD	9Ø	BF	48	A9	ØØ	8D
B2DØ:	58	E1	8D	59	E1	2Ø	56	B2	68	8D	9Ø	BF	68	8D	58	E1
B2EØ:	68	8D	59	E1	6Ø	2Ø	AC	BØ	A9	Ø5	8D	5Ø	E1	2Ø	8D	BØ
B2FØ:	A9	ØØ	8D	9Ø	BF	A9	Ø1	EA	EA	EA	38	ED	83	BF	ED	92
B3ØØ:	BF	8D	5B	E1	6Ø	AD	59	E1	48	AD	58	E1	48	AD	9Ø	BF
B31Ø:	48	2Ø	56	B2	68	8D	9Ø	BF	68	8D	58	E1	68	8D	59	E1
B32Ø:	6Ø	A2	ØØ	8E	65	E1	8E	92	BF	E8	8E	94	BF	A2	Ø3	8E
B33Ø:	51	E1	6Ø	A9	11	8D	53	E1	6Ø	AD	96	BF	FØ	11	A9	6Ø
B34Ø:	8D	99	BF	A9	ØØ	8D	98	BF	8D	ØØ	6Ø	8D	96	BF	6Ø	A9
B35Ø:	FF	DØ	F8	A9	Ø1	8D	95	BF	8D	94	BF	6Ø	FF	FF	AA	B4
B36Ø:	39	B3	4F	B3	53	B3	FF	FF	FF	FF	FF	FF	B4	B1	ØB	B2
B37Ø:	29	B2	FE	B1	3B	B2	49	B2	FF	FF	FF	FF	ØB	B2	FF	FF
B38Ø:	21	B3	FF	FF	33	B3	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
B39Ø:	C1	B2	C7	BØ	E5	B2	Ø5	B3	FF	FF	FF	FF	2Ø	DF	B4	2Ø
B3AØ:	A5	B3	4C	E9	B4	AD	94	BF	FØ	Ø3	4C	28	BØ	2Ø	3D	B4
B3BØ:	AD	92	BF	8D	8D	BF	A9	ØØ	8D	8F	BF	AD	8E	BF	DØ	Ø3
B3CØ:	4C	28	BØ	2Ø	18	BØ	AD	93	BF	Ø9	8Ø	8D	66	E1	2Ø	18
B3DØ:	BØ	2Ø	23	B1	9Ø	Ø8	2Ø	BC	BØ	BØ	Ø3	2Ø	56	B2	2Ø	18
B3EØ:	BØ	A9	ØA	8D	5Ø	E1	EE	8F	BF	2Ø	8D	BØ	2Ø	4C	B4	2Ø
B3FØ:	18	BØ	AD	8E	BF	C9	FF	DØ	Ø3	4C	28	BØ	8D	8B	BF	8D
B4ØØ:	8C	BF	2Ø	18	BØ	CE	8B	BF	DØ	F8	CE	8C	BF	DØ	F3	FØ
B41Ø:	BD	48	AD	94	BF	DØ	24	6E	8F	BF	9Ø	19	2Ø	23	B1	9Ø
B42Ø:	Ø3	2Ø	BC	BØ	A9	ØA	8D	5Ø	E1	2Ø	8D	BØ	2Ø	4C	B4	AD
B43Ø:	8D	BF	2Ø	F9	BØ	AD	93	BF	8D	66	E1	68	6Ø	8A	48	A2
B44Ø:	Ø3	BD	58	E1	9D	85	BF	CA	1Ø	F7	3Ø	ØD	8A	48	A2	Ø3
B45Ø:	BD	85	BF	9D	58	E1	CA	1Ø	F7	68	AA	6Ø	2Ø	DC	B4	2Ø
B46Ø:	65	B4	4C	E6	B4	AC	96	BF	DØ	37	C9	Ø3	9Ø	33	AC	94
B47Ø:	BF	FØ	ØC	CØ	DØ	FØ	Ø8	C9	2Ø	FØ	2C	C9	ØA	FØ	28	AØ
B48Ø:	ØØ	2Ø	FØ	B4	91	18	98	C8	91	18	2Ø	FØ	B4	EE	98	BF
B49Ø:	DØ	ØF	AD	99	BF	18	69	Ø1	C9	AF	DØ	Ø2	A9	6Ø	8D	99
B4AØ:	BF	2Ø	E6	B4	2Ø	73	B1	4C	Ø9	BØ	AD	96	BF	FØ	F8	AØ
B4BØ:	ØØ	8C	9A	BF	A9	6Ø	8D	9B	BF	2Ø	FØ	B4	AØ	ØØ	B1	1A
B4CØ:	2Ø	FØ	B4	FØ	E2	2Ø	A4	B4	EE	9A	BF	DØ	EC	AD	9B	BF
B4DØ:	18	69	Ø1	8D	9B	BF	C9	AF	FØ	CD	DØ	DD	8D	8Ø	BF	6C
B4EØ:	82	BF	8E	81	BF	6Ø	AD	8Ø	BF	AC	82	BF	AE	81	BF	6Ø
B4FØ:	48	98	48	AØ	Ø3	B9	18	ØØ	48	B9	98	BF	99	18	ØØ	68
B5ØØ:	99	98	BF	88	1Ø	EF	68	A8	68	6Ø	2Ø	1A	B5	AD	95	BF
B51Ø:	FØ	Ø7	AD	89	BF	C9	ØD	FØ	1F	6Ø	8D	89	BF	CØ	5Ø	DØ
B52Ø:	ØF	2Ø	43	B6	9Ø	15	C9	ØD	DØ	Ø5	A9	Ø1	8D	94	BF	6Ø
B53Ø:	C9	11	FØ	Ø4	C9	41	DØ	Ø9	4C	6E	B6	8D	5Ø	E1	4C	8D
B54Ø:	BØ	C9	2Ø	FØ	EA	2Ø	3C	B6	BØ	59	2Ø	4A	B6	AD	89	BF
B55Ø:	29	7F	8D	94	BF	38	E9	42	ØA	AA	BD	71	B5	8D	9C	BF
B56Ø:	BC	72	B5	8C	9D	BF	CØ	FF	DØ	Ø4	C9	FF	FØ	C1	6C	9C
B57Ø:	BF	AF	B6	F5	B6	91	B7	FF	FF	FF	FF	DB	B9	94	B6	A1
B58Ø:	B6	CE	B7	FF	FF	1A	B7	73	B8	B9	BA	BØ	B9	FF	FF	2B
B59Ø:	B7	86	B8	CF	B8	43	B7	F1	B8	18	B9	6Ø	B7	44	B9	FF
B5AØ:	FF	73	B7	CØ	Ø1	FØ	2Ø	A8	AD	94	BF	1Ø	1B	29	7F	8D
B5BØ:	94	BF	CØ	2B	FØ	11	CØ	2D	DØ	ØE	A9	Ø2	38	ED	8D	BF
B5CØ:	ØA	AA	A9	8Ø	9D	9E	BF	6Ø	CØ	2C	DØ	ØB	09	8Ø	8D	94
B5DØ:	BF	CE	8D	BF	3Ø	98	6Ø	CØ	ØD	DØ	Ø7	Ø9	8Ø	8D	94	BF
B5EØ:	DØ	8C	98	2Ø	33	B6	BØ	4A	29	ØF	48	A9	Ø2	38	ED	8D
B5FØ:	BF	ØA	AA	BD	9E	BF	48	BD	9F	BF	48	1E	9F	BF	3E	9E
B6ØØ:	BF	1E	9F	BF	3E	9E	BF	18	68	7D	9F	BF	9D	9F	BF	68
B61Ø:	7D	9E	BF	9D	9E	BF	1E	9F	BF	3E	9E	BF	68	Ø8	18	7D
B62Ø:	9F	BF	9D	9F	BF	A9	ØØ	7D	9E	BF	28	9Ø	Ø2	Ø9	8Ø	9D
B63Ø:	9E	BF	6Ø	C9	3Ø	9Ø	Ø3	C9	3A	6Ø	38	6Ø	C9	41	9Ø	FA
B64Ø:	C9	5B	6Ø	C9	2Ø	9Ø	F3	C9	8Ø	6Ø	A9	ØØ	AØ	Ø5	99	9E
B65Ø:	BF	88	1Ø	FA	6Ø	AD	94	BF	29	7F	8D	94	BF	2Ø	4A	B6
B66Ø:	AD	89	BF	C9	2C	DØ	Ø3	6C	9C	BF	A9	Ø1	DØ	22	A9	ØØ
B67Ø:	8D	95	BF	8D	94	BF	A9	ØB	8D	51	E1	A9	Ø7	2D	5B	E1
B68Ø:	FØ	Ø5	EE	5B	E1	DØ	F4	6Ø	8D	8D	BF	AD	94	BF	Ø9	8Ø
B69Ø:	8D	94	BF	6Ø	A2	Ø3	BD	85	BF	9D	58	E1	CA	1Ø	F7	3Ø
B6AØ:	ØB	A2	Ø3	BD	58	E1	9D	85	BF	CA	1Ø	F7	4C	6A	B6	AD
B6BØ:	94	BF	3Ø	Ø5	A9	ØØ	4C	88	B6	AD	A3	BF	29	ØF	48	2C
B6CØ:	A2	BF	1Ø	ØA	2D	97	BF	ØA	ØA	ØA	ØA	ØD	A3	BF	8D	91
B6DØ:	BF	8D	64	E1	A9	Ø2	2C	5Ø	E1	DØ	FB	2C	5Ø	E1	FØ	FB
B6EØ:	A9	ØC	8D	5Ø	E1	2Ø	8D	BØ	68	8D	97	BF	AD	91	BF	8D
B6FØ:	64	E1	4C	6A	B6	AD	94	BF	1Ø	36	AD	A3	BF	29	ØF	8D
B7ØØ:	A3	BF	2C	A2	BF	1Ø	ØA	2D	97	BF	ØA	ØA	ØA	ØA	ØD	A3
B71Ø:	BF	8D	91	BF	8D	64	E1	4C	6A	B6	AD	94	BF	1Ø	11	AD
B72Ø:	A3	BF	29	Ø3	8D	A3	BF	A9	ØC	DØ	29	AD	94	BF	3Ø	Ø5
B73Ø:	A9	ØØ	4C	88	B6	AD	A3	BF	29	Ø2	ØA	ØA	8D	A3	BF	A9
B74Ø:	Ø7	DØ	11	AD	94	BF	1Ø	E8	AD	A3	BF	29	Ø1	ØA	ØA	8D
B75Ø:	A3	BF	A9	ØB	2D	52	E1	ØD	A3	BF	8D	52	E1	4C	6A	B6
B76Ø:	AD	94	BF	1Ø	CB	AD	A3	BF	6A	6A	29	8Ø	8D	A3	BF	A9
B77Ø:	7F	DØ	ØF	AD	94	BF	1Ø	B8	AD	A3	BF	29	Ø3	8D	A3	BF
B78Ø:	A9	8Ø	2D	93	BF	ØD	A3	BF	8D	66	E1	8D	93	BF	4C	6A
B79Ø:	B6	AD	94	BF	1Ø	3D	A2	ØØ	2Ø	A3	B7	A2	Ø2	2Ø	A3	B7
B7AØ:	4C	D8	B7	BD	A1	BF	38	FD	59	E1	9D	A1	BF	BD	AØ	BF
B7BØ:	29	7F	FD	58	E1	9D	AØ	BF	BØ	13	A9	ØØ	38	FD	A1	BF
B7CØ:	9D	A1	BF	A9	ØØ	FD	AØ	BF	Ø9	8Ø	9D	AØ	BF	6Ø	AD	94
B7DØ:	BF	3Ø	Ø5	A9	Ø1	4C	88	B6	A2	Ø3	BD	AØ	BF	48	CA	1Ø
B7EØ:	F9	2Ø	E7	B7	4C	55	B6	BA	A9	4Ø	1E	Ø5	Ø1	2A	1E	Ø3
B7FØ:	Ø1	2A	2A	48	5E	Ø3	Ø1	5E	Ø5	Ø1	BD	Ø4	Ø1	8D	55	E1
B8ØØ:	BD	Ø6	Ø1	8D	57	E1	A9	Ø1	9D	Ø4	Ø1	9D	Ø6	Ø1	BC	Ø3
B81Ø:	Ø1	DØ	Ø5	BC	Ø5	Ø1	FØ	15	5E	Ø3	Ø1	6E	55	E1	7E	Ø4
B82Ø:	Ø1	5E	Ø5	Ø1	6E	57	E1	7E	Ø6	Ø1	ØA	DØ	E1	A8	BD	Ø3
B83Ø:	Ø1	18	7D	Ø4	Ø1	9D	Ø3	Ø1	Ø8	BD	Ø5	Ø1	18	7D	Ø6	Ø1
B84Ø:	9D	Ø5	Ø1	BD	ØØ	Ø1	9Ø	Ø2	Ø9	88	28	9Ø	Ø2	Ø9	AØ	48
B85Ø:	68	1Ø	Ø6	8D	5Ø	E1	2Ø	8D	BØ	BD	ØØ	Ø1	Ø9	1Ø	8D	5Ø
B86Ø:	E1	2Ø	8D	BØ	88	DØ	C7	68	68	9D	Ø5	Ø1	68	9D	Ø6	Ø1
B87Ø:	68	68	6Ø	AD	94	BF	1Ø	13	A2	Ø3	BD	AØ	BF	9D	58	E1
B88Ø:	CA	1Ø	F7	4C	6A	B6	AD	94	BF	3Ø	Ø5	A9	Ø1	4C	88	B6
B89Ø:	A2	ØØ	2Ø	9D	B8	A2	Ø2	2Ø	9D	B8	4C	6A	B6	BD	AØ	BF
B8AØ:	3Ø	14	BD	59	E1	18	7D	A1	BF	9D	59	E1	BD	58	E1	7D
B8BØ:	AØ	BF	9D	58	E1	6Ø	29	7F	9D	AØ	BF	BD	59	E1	38	FD
B8CØ:	A1	BF	9D	59	E1	BD	58	E1	FD	AØ	BF	9D	58	E1	6Ø	AD
B8DØ:	94	BF	3Ø	Ø5	A9	Ø1	4C	88	B6	AD	A3	BF	29	ØF	8D	A3
B8EØ:	BF	AD	A1	BF	ØA	ØA	ØA	ØA	ØD	A3	BF	8D	53	E1	4C	6A
B8FØ:	B6	AD	94	BF	1Ø	DE	AD	A3	BF	29	Ø1	8D	A3	BF	AD	A1

```
B900: BF 29 01 0A 0D A3 BF 8D A3 BF AD 51 E1 29 FC 0D
B910: A3 BF 8D 51 E1 4C 6A B6 AD 94 BF 10 B7 A2 03 BD
B920: 58 E1 48 CA 10 F9 A2 03 BD A0 BF 9D 58 E1 CA 10
B930: F7 A9 0F 8D 50 E1 20 8D B0 AD 64 E1 29 0F 8D 8A
B940: BF 4C 2D BA AD 94 BF 30 09 A9 02 4C 88 B6 18 1E
B950: 1A 1C AD A1 BF 8D 55 E1 8D 57 E1 AD 9F BF 29 01
B960: 8D 9F BF 0E A0 BF 2A AA BC 4E B9 AE A3 BF 8C 50
B970: E1 20 8D B0 AD 9F BF F0 06 20 88 B9 18 90 03 20
B980: 9C B9 CA D0 E9 4C 55 B6 20 93 B9 A9 D6 8D 50 E1
B990: 20 8D B0 EE 59 E1 D0 03 EE 58 E1 60 20 A7 B9 A9
B9A0: D4 8D 50 E1 20 8D B0 EE 5B E1 D0 03 EE 5A E1 60
B9B0: AD 94 BF 10 94 AD 9F BF D0 15 A9 FF 8D 9F BF AD
B9C0: A3 BF F0 02 A9 01 8D A0 BF 20 FD BA 4C 6A B6 AD
B9D0: A3 BF CD A1 BF D0 ED A9 02 D0 EB AD 94 BF 10 D3
B9E0: A2 03 BD 58 E1 48 CA 10 F9 2C 9E BF 08 AD 9F BF
B9F0: 29 01 F0 47 20 82 BA 20 A6 BA 20 82 BA 20 A6 BA
BA00: 28 10 2A A2 00 20 61 BA F0 23 2C A0 BF 30 0B EE
BA10: 59 E1 D0 13 EE 58 E1 4C 27 BA CE 59 E1 AD 59 E1
BA20: C9 FF D0 03 CE 58 E1 20 A6 BA 4C 03 BA A2 00 68
BA30: 9D 58 E1 E8 E0 04 D0 F7 4C 55 B6 20 82 BA 20 88
BA40: BA 20 A6 BA 28 10 E6 20 82 BA A2 02 20 61 BA F0
BA50: DC 20 88 BA A2 00 20 61 BA F0 D2 20 88 BA 4C 4A
BA60: BA 20 77 BA F0 1B BD A1 BF 38 E9 01 9D A1 BF BD
BA70: A0 BF E9 00 9D A0 BF BD A0 BF 29 7F D0 03 BD A1
BA80: BF 60 A9 00 48 48 F0 0D AD A3 BF 48 AD A2 BF 48
BA90: 49 80 8D A2 BF AD A1 BF 48 AD A0 BF 48 49 80 8D
BAA0: A0 BF 20 E7 B7 60 AD A3 BF 48 AD A2 BF 48 49 80
BAB0: 8D A2 BF A9 00 48 48 F0 E9 AD 94 BF 30 05 A9 02
BAC0: 4C 88 B6 A2 03 BD 58 E1 48 BD A0 BF 9D 58 E1 CA
BAD0: 10 F3 AD 9F BF 29 0F 8D 9F BF 2C 9E BF 10 0A 2D
BAE0: 97 BF 0A 0A 0A 0A 0D 9F BF 8D 64 E1 A9 80 8D 50
BAF0: E1 20 8D B0 AD 91 BF 8D 64 E1 4C 2D BA 4C 49 BC
BB00: 8E 84 BF BA BD 04 01 B0 0D 7D 06 01 A8 BD 03 01
BB10: 7D 05 01 E8 D0 0B FD 06 01 A8 BD 03 01 FD 05 01
BB20: E8 48 BD 00 01 9D 04 01 BD 01 01 9D 05 01 68 E8
BB30: E8 E8 9A AE 84 BF 60 18 24 38 A2 00 F0 05 18 24
BB40: 38 A2 02 48 A9 00 48 BD 86 BF 48 BD 85 BF 48 20
BB50: 00 BB 9D 58 E1 98 9D 59 E1 60 EE A6 BF D0 03 EE
BB60: A7 BF 60 8C 50 E1 4C 8D B0 A9 00 AC A0 BF C0 02
BB70: D0 07 AD A4 BF 38 ED A5 BF 8D 55 E1 4A 8D 57 E1
BB80: A9 01 2C 9F BF F0 12 AD A4 BF 20 37 BB AD A5 BF
BB90: 4A 20 3E BB A0 16 20 63 BB A9 02 2C 9F BF F0 12
BBA0: AD A5 BF 20 37 BB AD A4 BF 4A 20 3E BB A0 14 20
BBB0: 63 BB A9 04 2C 9F BF F0 12 AD A5 BF 20 39 BB AD
BBC0: A4 BF 4A 20 3E BB A0 14 20 63 BB A9 08 2C 9F BF
BBD0: F0 12 AD A4 BF 20 39 BB AD A5 BF 4A 20 3E BB A0
BBE0: 10 20 63 BB A9 10 2C 9F BF F0 12 AD A4 BF 20 39
BBF0: BB AD A5 BF 4A 20 40 BB A0 10 20 63 BB A9 20 2C
BC00: 9F BF F0 12 AD A5 BF 20 39 BB AD A4 BF 4A 20 40
BC10: BB A0 12 20 63 BB A9 40 2C 9F BF F0 12 AD A5 BF
BC20: 20 37 BB AD A4 BF 4A 20 40 BB A0 12 20 63 BB A9
BC30: 80 2C 9F BF D0 01 60 AD A4 BF 20 37 BB AD A5 BF
BC40: 4A 20 40 BB A0 16 4C 63 BB AD A1 BF 8D A4 BF 8D
BC50: A6 BF A9 00 8D A5 BF 8D A7 BF 8D 65 E1 8D 92 BF
BC60: AC A0 BF F0 14 C0 01 F0 0D 38 ED A6 BF 8D A6 BF
BC70: CE A7 BF 4C 52 BD 20 69 BB AD A4 BF 0A 48 A9 00
BC80: 2A 48 AD A6 BF 48 AD A7 BF 48 38 20 00 BB 8D A7
BC90: BF 8C A6 BF AC A5 BF A9 02 CD A0 BF D0 05 18 6D
BCA0: A5 BF A8 CC A4 BF 90 03 4C 58 BD AD A0 BF D0 03
BCB0: 20 69 BB AD A6 BF 48 AD A7 BF 48 AD A5 BF 0A 48
BCC0: A9 00 2A 48 18 20 00 BB 8D A7 BF 8C A6 BF 20 5A
BCD0: BB EE A5 BF AD A7 BF 30 2C AD A4 BF 0A 48 A9 00
BCE0: 2A 48 AD A6 BF 48 AD A7 BF 48 38 20 00 BB 8D A7
BCF0: BF 8C A6 BF 20 5A BB 20 5A BB AD A4 BF 38 E9 01
BD00: 8D A4 BF 90 53 AD A0 BF F0 4B C9 01 D0 44 AD A4
BD10: BF 0A 48 A9 00 2A 48 AD A6 BF 48 AD A7 BF 48 38
BD20: 20 00 BB AA 98 48 8A 48 AD A1 BF 0A 48 A9 00 2A
BD30: 48 18 20 00 BB C8 D0 03 18 69 01 29 80 C9 80 D0
BD40: 11 38 AD A4 BF E9 01 8D A4 BF B0 03 20 69 BB EE
BD50: A4 BF 20 69 BB 4C 94 BC AD A0 BF C9 01 D0 05 AD
BD60: A3 BF D0 01 60 CE A3 BF CE A1 BF 4C 49 BC
```

ELEKTUUR

Concevoir un émetteur expérimental

Pierre Loglisci

Quel radio-amateur, réellement amateur, ne rêve pas d'être capable de construire un émetteur non pas à l'aide d'un plan copié dans un quelconque ouvrage ou revue, mais basé sur un schéma entièrement conçu et calculé par lui-même?
Devenir, dans le domaine de l'émission, son propre ingénieur-concepteur est une aspiration comparable au désir qu'expriment les passionnés de microprocesseurs de concevoir et de réaliser leur propre ordinateur personnel sans avoir copié ni le matériel ni le logiciel. Eux cependant courrent le risque d'une incompatibilité majeure avec les autres amateurs de micro-informatique, ce qui n'est heureusement pas le cas en HF.
L'ouvrage présenté ici ouvre de nouveaux horizons à tous ceux qui se sentent l'âme d'un expérimentateur. Plus de 70 schémas, photographies et dessins pavent un sentier à la portée des moins alpinistes d'entre nous. Associé à la platine d'expérimentation "spéciale HF" décrite en octobre 85, cet ouvrage constitue une excellente entrée en matière pour tout débutant.

Format 14 x 21 cm
Editions SORACOM
16, av Gros Malhon
35000 Rennes

Le tout MICRO

Près de 450 pages pleines à craquer de détails concernant le monde la micro-informatique: boutiques, librairies, ouvrages, logiciels, matériels. On ne peut pas, bien évidemment, être complet. Il s'agit en quelque sorte du QUID de la micro-informatique.
Cet ouvrage est indispensable à tout débutant qui ne sait pas encore quel matériel acheter. Lorsque son choix est fait, il se tournera sans doute vers une revue (ou des ouvrages) spécialisée centrée sur le type de matériel qu'il aura choisi. Chaque jour voit arriver sur le marché de nouveaux logiciels et comme il faut un certain temps pour réaliser un ouvrage de cette envergure, il ne faudra pas s'étonner de ne pas y trouver un logiciel très récent.
Il est prévu une actualisation annuelle de cet ouvrage, sous quelle forme? nous ne le savons pas encore.
Donner des prix dans un ouvrage de cette sorte comporte toujours de gros risques, car vu le train auquel se développe la micro-informatique, ce qui était vrai hier ne l'est déjà plus demain (si ce n'est aujourd'hui).

Hachette Informatique
22, rue la Boétie
75008 Paris