1ère partie

Si le graphisme haute résolution en couleurs vous intéresse, ne vous souciez pas de l'encadré rabat-joie ci-dessous à gauche, car la carte présentée dans cet article offre à celui qui la construira, une multitude de possibilités étonnantes, rarement disponibles simultanément sur un projet de ce genre. Cette efficacité est le fruit d'une habile conjugaison du matériel et du logiciel, pendant de longs mois d'expérimentation. On trouvera dans le tableau ci-contre un synoptique des caractéristiques du système, plus éloquent que nos longs palabres.

P. Lavigne

carte graphique haute résolution en couleurs

512 × 512 ou 512 × 256 points en N&B, 8 ou 16 couleurs avec jusqu'à 256 K de RAM dynamique autonome et logiciel complet

L'électronique, comme le tourisme, a ses autoroutes (à péage, s'il vous plaît) et ses sentiers de grande randonnée. Au seuil de cette série d'articles consacrés à ce nouveau montage prestigieux qu'est la carte graphique d'Elektor, il importe de mettre les points sur les i. Il s'agit de toute évidence d'une aventure que nous abordons ici, et non d'une simple partie de campagne. Alors quittez vos chaussures de ville, et mettez vos godillots de marche!
Le manque de place nous oblige à répartir la publication de cet article sur plusieurs numéros. Que nos lecteurs impatients veuillent nous en excuser... il nous faut aussi songer à satisfaire ceux d'entre eux que ce projet-ci n'intéresse pas.

Présentation

A la question "qu'apporte une carte graphique comme celle-ci de plus que la carte VDU?", on peut répondre en trois mots: le graphisme point par point (au lieu du graphisme par blocs de la carte VDU), la haute résolution de l'image et la couleur. Et ceci sans que soit perdue la possibilité d'afficher également des textes ou des listings. C'est-à-dire que **la carte graphique avec son logiciel est aussi un terminal vidéo alphanumérique** de 80 colonnes sur 32 lignes (avec en plus la possibilité de programmer la taille et la couleur des caractères, un confort appréciable notamment lors de l'affichage de tableaux ou de menus).
Les points sur l'écran sont accessibles sur une matrice XY dont les coordonnées sont transmises au processeur graphique (GDP)*. On notera que le point d'origine (X=0 et Y=0) se trouve en haut à gauche de l'écran en mode "texte" et en bas à gauche de l'écran en mode "graphique".
Il n'est pas aisé de résumer en quelques lignes les caractéristiques d'un système de cette taille. Ce n'est pas non plus au premier abord que l'on peut les saisir toutes.
Il s'agit d'une espèce de terminal autonome, dont le matériel génère des images vidéo graphiques à partir des instructions reçues et interprétées par le logiciel associé. Un peu de la même manière qu'une imprimante reçoit des codes ASCII qu'elle convertit en signaux de commande pour les aiguilles de sa tête d'impression.
Cependant, il y a une différence essentielle entre une imprimante ou une table traçante et la carte graphique: le logiciel (interpréteur de commandes) n'est pas exécuté par une CPU autonome, mais par le microprocesseur hôte du système. Le GDP sur la carte graphique se charge pour sa part de gérer la mémoire d'image autonome et d'effectuer les tracés.
L'ensemble de la carte, avec extension couleurs, ne mobilise que 19 adresses dans l'espace mémoire du système hôte. Elle est donc facile à insérer dans tout système, notamment à 6502, le seul processeur pour lequel l'interpréteur de commande soit disponible pour l'instant. Rien n'interdit toutefois de se passer de ce logiciel et d'attaquer directement la carte graphique. On se prive alors d'un confort indéniable...Nous verrons ultérieurement comment procéder à la mise en place du logiciel.
La carte mère et l'extension couleurs sont l'une et l'autre au format européen. La première est autonome en N&B; d'un côté elle communique avec le bus du microprocesseur, de l'autre elle délivre les signaux vidéo que l'on applique directement à un moniteur. Outre le GDP et la circuiterie TTL, elle comporte 64 K de mémoire vive dynamique. Sur la carte couleurs, que nous aborderons plus tard, on trouve 3 bancs de 64 K qui constituent une extension montée en parallèle sur la carte mère. Cette seconde carte communique d'une part avec le bus de données du microprocesseur hôte, et d'autre part, via un bus interne, avec la carte mère. Elle délivre aussi les signaux RVB (rouge, vert, bleu) qui, associés au signal vidéo de la carte mère, permettent d'obtenir 2, 4, 8 ou 16 couleurs (ou teintes de gris) sur un moniteur équipé d'entrées RVB ou RVBI (où I est un bit d'intensité). Des extensions supplémentaires sont possibles: en rajoutant une seconde carte d'extension en parallèle sur la première (c'est prévu), on obtient 32, 64 ou 128 couleurs...Il apparaît cependant que l'utilisation d'un circuit

*GDP: graphic display processor.

Caractéristiques de la carte graphique

carte graphique haute résolution en couleurs
elektor septembre 1985

MATERIEL

Processeur:	GDP 9366 (9365) ou GDP 9367 tracé de la diagonale de l'écran (512 points) en moins de 700 μs
Bus de données:	8 bits tamponnés
Adresses décodées:	19 (XX50...XX5F, XX64...XX66). La mémoire d'image est autonome et rafraîchie automatiquement (16 K ou 32 K par page et par couleur)
Synchro:	composite, normale ou inversée (sortie TTL tamponnée)
Vidéo:	N&B, RVB ou RVBI (sorties TTL tamponnées)
Photostyle:	impulsion négative (entrée TTL tamponnée)
Résolution:	**4 pages de 512 × 256 pixels** (9366/9367) **2 pages de 512 × 512 pixels** (entrelacé/9367 seulement)
Couleurs:	N&B sur la carte principale RVB sur l'extension } **8 ou 16 couleurs**
	Note: le nombre de couleurs n'exerce aucune influence restrictive sur la résolution. L'adjonction de plusieurs cartes d'extension est possible.
Scrolling:	échappement vertical (le logiciel modifie les adresses de visualisation fournies par le GDP à la mémoire d'image).
RMW:	mode lecture/modification/écriture (combinaison OU-Exclusif entre la plume et le papier).
Echanges avec la mémoire d'image:	Lecture pixel par pixel (le microprocesseur donne les coordonnées du pixel au GDP, suivies d'une instruction spéciale)
Interruptions:	$\overline{IRQ}$ (3 modes programmables)
Générateurs:	■ de caractères alphanumériques (matrice 5 × 8). La taille des caractères est programmable séparément sur les axes X et Y ■ de vecteurs programmables (4 types de traits) — voir logiciel.

Hormis le GDP lui-même, la carte ne fait appel qu'à des circuits intégrés courants.

LOGICIEL

La carte graphique est fournie avec un logiciel complet (un peu moins de 4 K de code objet 6502). Celui-ci est subdivisé en deux programmes, qui gèrent l'un tout ce qui est **texte** (un terminal vidéo en quelque sorte: codes ASCII) et l'autre tout ce qui est **graphique**. Ce logiciel est parfaitement autonome et fonctionne en quelque sorte comme une super-routine de réception du caractère contenu dans l'accumulateur. On notera la concision des instructions (voir tableau 1 ci-dessous).

Tableau 1

CHR$	Text mode commands		Graphic mode commands
(1)	Transfer video buffer contents to screen	A	Set text mode
(2)	Open the video buffer	B n	Set background color to color "n"
(3)	Close the video buffer	B —n	Combine the background color with color "n"
(4)	Use graphic commands in text mode	C n	Set pen color to color "n"
(5)		C —n	Combine the pen color with color "n"
(6)		D x,y	Draw from current position to x,y destination
(7)		D x,y,x,y,...	D may be followed by several x,y destinations
(8)	Backspace (cursor left)		X,Y coordinates are given from absolute origin
(9)	Horizontal tabulation (cursor right)	E	
10)	Line Feed (cursor down)	F	
11)	Vertical tabulation (cursor up)	G ±n,x,y	n=1: ; n=2: ; n=—1: ; n=—2
12)	Clear screen&home cursor	H	Home pen to current origin (without drawing)
13)	Carriage Return	I	Set current location as new absolute origin
14)		J x,y	Draw from current pen location to **relative**
15)			position x,y
16)		J x,y,x,y,...	J may be followed by several x,y destinations
17)	Set **text** mode	K	
18)	**Set graphic** mode	L n	Set line type n; n=0: ——; n=1: . . . . ;
19)			
20)	Reset the character size		n=2; — —; n=3; — . — .
21)		M x,y	Move to absolute location x,y without drawing
22)		N	
23)		O s,r,t	Draw a circle or a disk with current absolute
24)			origin as center
25)			s = **sectors**; r = **radius**; t = **thickness**
26)	Erase current line		
27)	Escape		
28)	Home cursor		
29)	Clear to end of line	P characters	Print alphanumeric characters without leaving
30)			graphic mode
31)		Q d	Set the print direction; d = 0: **horizontal**;
			d = 1: **vertical**
		R x,y	Move to relative x,y position from current pen
			location (without drawing)
		S x,y	Set character size x on X axis / y on Y axis
		T t	Set character type t; t = 0: normal;
			t = 1: italics
		U p,u	Select pen/eraser up/down; p=1: pen; p = 0:
			eraser; u=1: down; u=0: up
		V x,y	Get pixel status in specified x,y location
		W m	Set Read-Modify-Write mode; m=0: **no RMW**
			mode; m=1: **RMW**
		X a,s,i	Draw a coordinate axis from current location in
			direction "a" using increments "s"(teps)
			(+ or —) and marking "i"(ntervals)
		Z p	Select page "p"

xemples en BASIC:

```
'RINT CHR$(18) "M0,127,I"
'RINT"B6,C4,D255,127,255,0"
'RINT"C2,M127,63,I"
'RINT"O255,60,2"
tc...
```

lote: Les instructions E, F, ; N, Y ainsi que certaines ıstructions CHR$ ne sont pas tilisées pour l'instant. Elles ont réservées à des extensions n préparation, comme par xemple F pour *fill*, etc.

Tableau 1. Dans le tableau ci-contre, on trouve les instructions acceptées par l'interpréteur de commandes que nous étudierons plus tard. Ces instructions ne sont pas destinées directement au GDP qui serait d'ailleurs bien incapable de les comprendre sous cette forme-là. On notera encore que la plupart de ces instructions sont les mêmes que celles l'on utilise couramment pour certaines tables traçantes.
Avant de présenter le logiciel interpréteur de commandes, il nous faut étudier le matériel de la carte mère, puis celui de la carte d'extension couleurs.

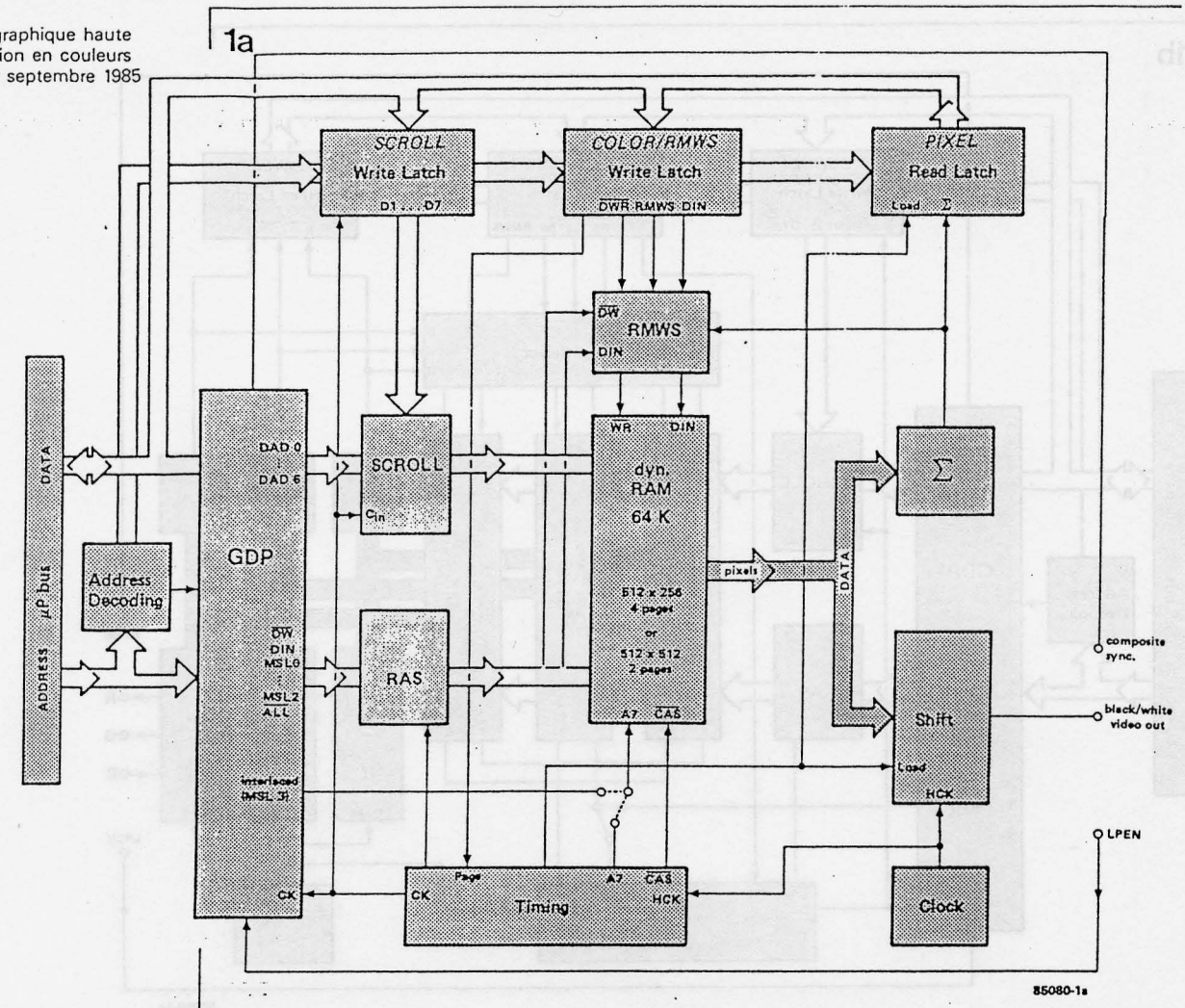


Figure 1a. Ce synoptique montre clairement que le microprocesseur du système avec lequel est utilisée la carte graphique a accès à celle-ci non seulement via les registres du GDP mais aussi via trois autres registres de lecture ou d'écriture (SCROLL, COLOR/RMWS/ PAGE et PIXEL en italiques ci-dessus). On dispose ainsi de fonctions que le GDP lui-même ne connaît pas, à savoir l'échappement vertical, la couleur, le mode lecture/ modification/écriture, la commutation de page et la lecture du contenu de la mémoire vidéo pixel par pixel.

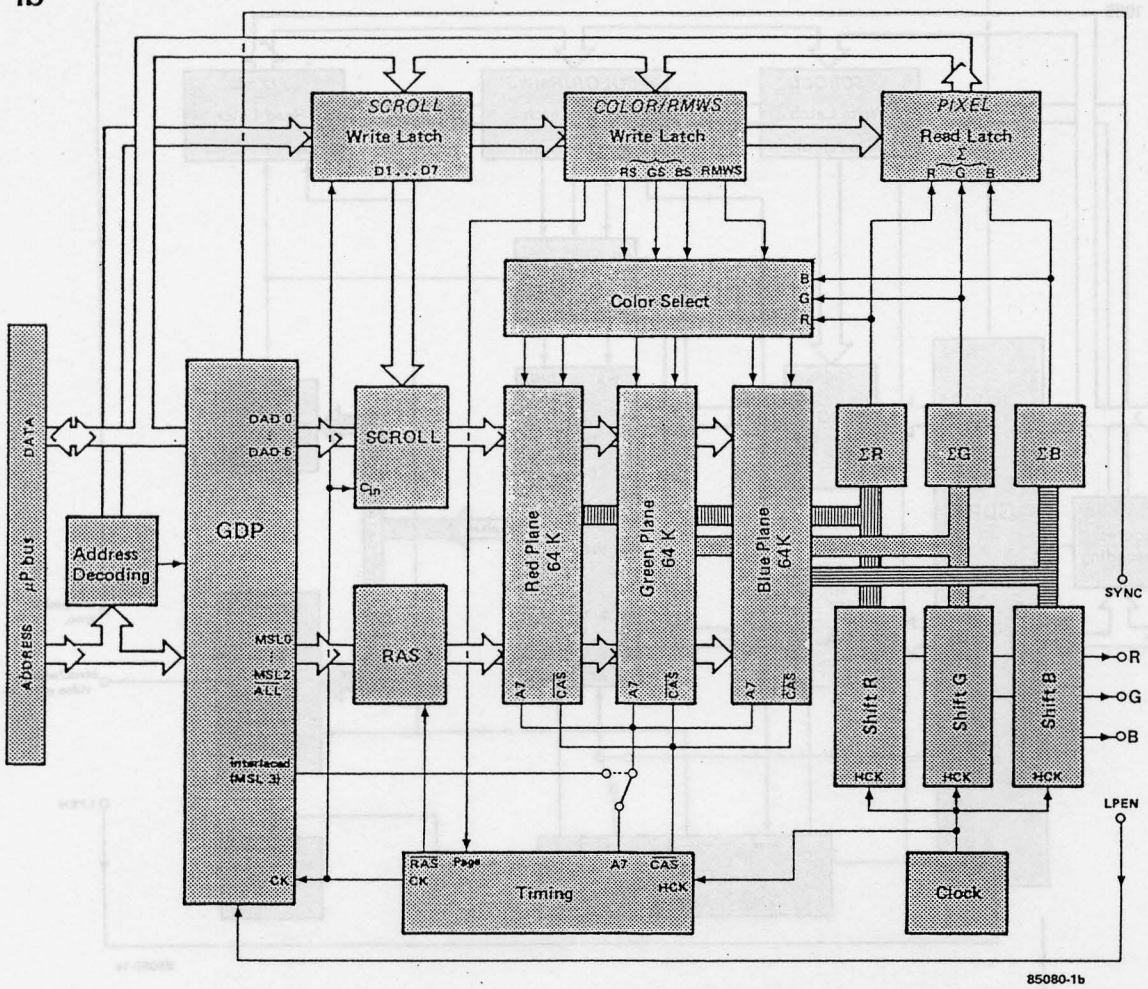
spécialisé dans la multiplication des couleurs (*color palette*) serait plus judicieuse. Qui vivra verra!

Un synoptique rassurant

Simplifié comme il l'est sur la **figure 1a**, le schéma présente une structure somme toute assez simple: à gauche, le GDP (*graphic display processor*) autour duquel tout tourne. A droite, la sortie vidéo N&B, dont le signal est fourni par un classique registre à décalage (*shift register*). Et entre les deux, la mémoire vidéo gérée par le GDP. Ce sont 64 Koctets absolument indépendants de la mémoire vive de l'ordinateur avec lequel la carte graphique est utilisée. Dans le bas de la figure, on trouve deux blocs (*timing* et *clock*) dont dépend la chronologie des signaux internes, dérivée de l'horloge-points (*dot clock*) qui est de 12 ou 14 MHz selon la version. L'accès à la carte graphique par le bus de données du microprocesseur est double: il y a d'une part une communication entre le bus de données et le GDP, et d'autre part une communication de ce bus avec des verrous de lecture ou d'écriture (*read/write latch*). Cette communication s'effectue sous les auspices d'un bloc de décodage d'adresses interne, de telle sorte que la carte graphique n'occupe qu'une vingtaine d'adresses de l'espace mémoire adressable par le micro-ordinateur avec lequel elle est utilisée. Un rapide survol des registres de lecture ou d'écriture nous révèle leur fonction. Le premier (*scroll*) donne la possibilité au microprocesseur de modifier les adresses fournies par le GDP à la mémoire vidéo, afin d'obtenir un échappement vertical du contenu de l'écran (ceci est intéressant surtout lorsque l'on fait défiler du texte — par exemple des listings). Le deuxième (*color*) permet, comme son nom l'indique déjà, de modifier la couleur des points. Mais ce n'est pas tout. Ce bloc permet également de commuter la mémoire d'une page à l'autre, que ce soit en mode 512 x 256 (non entrelacé/quatre pages) ou en mode 512 x 512 (entrelacé/deux pages). A quoi vient s'ajouter une troisième fonction extrêmement importante, à savoir la validation et l'invalidation du mode lecture-modification-écriture (*read-modify-write mode select*). Ce mode particulier permet de changer le contenu de l'écran, puis de le remettre dans son état initial **sans qu'il soit nécessaire de mémoriser cet état initial avant la modification.** Nous aurons l'occasion de revenir en détails là-dessus.

Le dernier bloc (*pixel*) associé au bloc Σ (sigma = somme), permet au micro-ordinateur de prendre connaissance de l'état d'un pixel dont il a, au préalable, donné les coordonnées au GDP. Il s'agit donc, fort logiquement, d'un registre de lecture. Entre le GDP et la mémoire vive se trou-

1b



vent encore deux blocs dont la fonction est essentielle dans le système. Celui du haut (SCROLL) modifie, comme nous l'avons déjà indiqué, les adresses fournies par le GDP à la mémoire vidéo afin d'obtenir, lorsque c'est souhaitable, un échappement vertical du contenu de l'écran. Celui du bas (RAS = *row address strobe*) ne communique pas avec le monde extérieur à la carte graphique: sa fonction est d'effectuer la distinction entre deux modes d'accès différents du GDP à la mémoire vidéo: d'une part l'accès simultané aux huit circuits intégrés de mémoire (pour le rafraîchissement ou la modification de la couleur de fond, situation dans laquelle tout l'octet adressé est concerné), et d'autre part l'accès à un seul des huit circuits (pour l'écriture ou la lecture d'un seul point, situation dans laquelle un seul bit sur les huit de l'octet adressé est concerné).

Sur la **figure 1b**, nous présentons, à titre d'avant-première, le synoptique du système avec couleurs (8). Ceci permettra au lecteur de se faire une idée d'ensemble du projet. Nous entrerons dans les détails de ce second synoptique en temps utile. Pour l'heure, il suffit de savoir que l'extension couleurs n'est pas unique; on peut, en principe, la répéter en autant d'exemplaires qu'on le souhaite. Nous verrons cependant qu'en pratique, ceci n'est pas très raisonnable. . .

Vidéo

Avant d'aborder la carte graphique proprement dite, il n'est certainement pas inutile de rappeler en quelques mots et avec quelques chiffres en quoi consiste le signal vidéo qui permet d'obtenir une image graphique sur l'écran.

Nous savons que l'écran est balayé par un faisceau d'électrons qui se déplace à raison de 64 μs par ligne. Le nombre de lignes par écran est de 625, et la fréquence de trame est de 50 Hz. Nous définissons sur l'écran une fenêtre qui en couvre environ les deux tiers, de sorte que la durée d'une ligne de balayage en-deçà de cette fenêtre est de 42 μs environ. Si une telle ligne est décomposée en 512 points, chacun de ces points durera à peu près 83 ns (une ligne de balayage complète — c'est-à-dire d'un bout à l'autre de l'écran — comporterait donc 768 points). Ce qui nous donne une fréquence de points de 12 MHz que l'on applique à un registre à décalage comme fréquence d'horloge. Si la capacité de ce registre est de huit bits (il charge un octet à la fois), la fréquence de chargement des octets sera donc de 1,5 MHz.

Au début d'une ligne de balayage, il ne se passe donc rien tant que le faisceau n'est pas arrivé dans la fenêtre de visualisation. Lorsqu'il est arrivé au bord gauche de cette fenêtre, le registre à décalage charge le premier octet et commence le déca-

Figure 1b. La version couleurs de la carte graphique est identique à la version N&B de la figure 1a, à ceci près que la mémoire a été étendue. Pour éviter que l'utilisation de la couleur ne compromette la résolution du système, il est nécessaire de rajouter autant de bancs de mémoire que l'on souhaite avoir de couleurs primitives. Ici, nous n'avons représenté que trois bancs (RGB) alors qu'en réalité leur nombre est illimité. En pratique, on se limite cependant à quatres bancs: RGB, plus un bit d'intensité (I), ce qui donne 16 couleurs.

2a

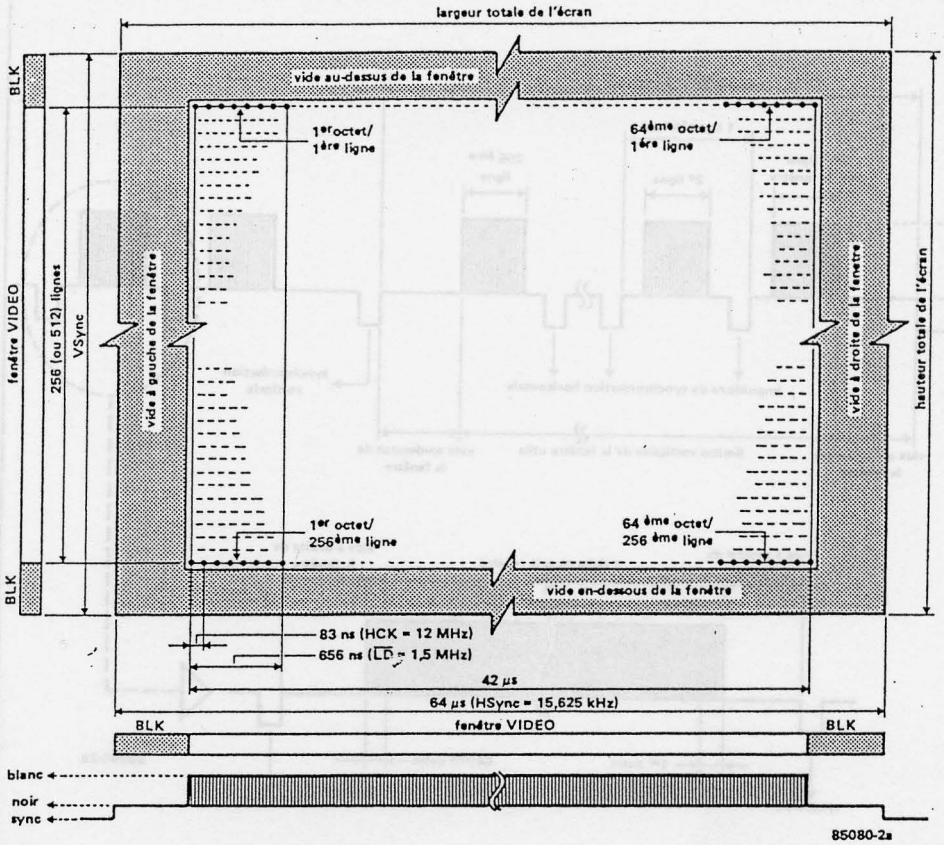


Figure 2a. L'image générée par le GDP de Thomson ne couvre pas la totalité de l'écran, mais une fenêtre seulement, ce qui nous débarrasse des distortions latérales ainsi que de celles des bords supérieur et inférieur de l'écran. Cette fenêtre est un peu plus large avec le 9367 qu'avec le 9366 ou le 9365. Ceci est dû à la fréquence d'horloge sensiblement moins élevée: 12 MHz au lieu de 14 MHz. Lorsque le faisceau du tube cathodique ne se trouve pas en-deçà des limites de la fenêtre de visualisaton, le GDP le signale en activant la ligne BLK (blanking).

lage à raison d'un point toutes les 83 ns. Au terme de 8 x 83 ns, les 8 premiers points en haut à gauche de la fenêtre de visualisation ont été allumés ou éteints. Le registre à décalage charge le deuxième octet et continue le décalage...Et ainsi de suite, jusqu'à ce qu'il ait décalé le dernier bit du 64ème octet de cette ligne. Nous sommes alors au bord droit de la fenêtre de visualisation. Il ne se passe plus rien jusqu'à l'impulsion de synchronisation de ligne qui ramène le faisceau au début de la ligne de balayage suivante, où le même processus se reproduit, et ce jusqu'à la fin de la dernière ligne, au bas de la fenêtre. Là le faisceau reste éteint, puisque le bas de l'écran, de même que le haut, se trouvent en dehors de la fenêtre et restent donc toujours vides.
Survient alors l'impulsion de synchronisation verticale ou synchronisation de trame. Le faisceau est éteint pour être ramené en haut à gauche de l'écran, où le balayage recommence. En résumé, il nous faut donc d'une part des **impulsions de synchronisation** (horizontale et verticale) dont la fréquence est immuable, quel que soit le contenu de l'image; et d'autre part **le signal vidéo** proprement dit, c'est-à-dire les impulsions correspondant aux points allumés ou éteints. Ce signal est obtenu à partir de l'horloge-points, qui cadence le registre à décalage de sortie (conversion parallèle-série). Ce registre à décalage reçoit l'information "points allumés/points éteints" de la mémoire vidéo, par paquets de huit bits chargés simultanément. L'impulsion de chargement d'un octet survient donc au terme du décalage des huit bits précédents, du moins tant que le faisceau de balayage se trouve dans la fenêtre de visualisation. En dehors de cette fenêtre, **un signal d'inhibition** (*blanking*) garantit l'extinction du faisceau. L'ensemble de cette procédure est illustrée par la **figure 2a**. Le signal vidéo typique est détaillé sur la **figure 2b**. On notera que les durées indiquées pour les signaux sont passablement approximatives. Qui chipotera pour quelques milliardièmes de seconde?

Noir et Blanc ≠ Couleur

Nous avons vu ce qu'était le signal vidéo en N&B, et nous avons également montré comment ce signal naissait à travers un registre à décalage, à partir de bits parallèles, stockés en mémoire par paquets de huit (voir aussi figure 3). En termes de télévision, c'est un signal de luminance binaire: il indique, par son niveau logique, si un point donné est éteint ou allumé, c'est tout. Il n'y a pas d'intermédiaire. Il existe également ce que l'on appelle un signal de chrominance et qui indique, par son **niveau analogique**, quelle est le degré de luminosité de la couleur (en termes de couleurs primitives saturées), ou la teinte de gris de ce point, lorsqu'il n'est pas éteint. Or, pour un ordinateur, il n'est pas question de travailler avec des valeurs analogiques. Les valeurs des couleurs doi-

2b

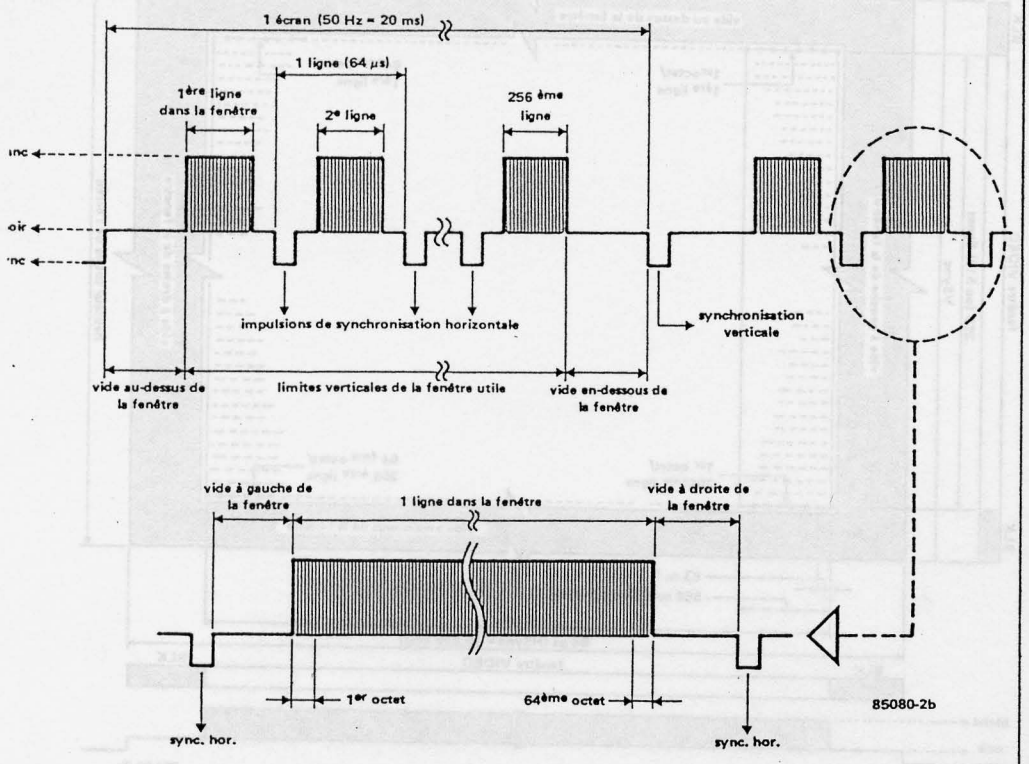


Figure 2b. Comme la fenêtre de visualisation est sensiblement plus petite que la surface totale de l'écran, la durée des pauses pendant lesquelles le signal vidéo est au niveau "noir" est nettement plus longue que normalement.
Pour simplifier les choses, nous n'avons pas représenté les impulsions de synchronisation horizontale dans le "vide au-dessus et en-dessous de la fenêtre"; cependant, celles-ci existent bel et bien!

vent être ramenées à des signaux binaires. C'est pourquoi nous faisons appel à trois signaux binaires juxtaposés, qui indiquent chacun l'état du point de couleur auquel il correspond. Les trois points ensemble forment ce que l'on appelle un pixel (de l'anglais *picture element*) ou triad, dont la couleur est donc définie par la magnitude du mot binaire que forment les niveaux logiques des trois signaux juxtaposés. Ceux-ci peuvent être appliqués directement à un moniteur couleurs, muni d'entrées RVB compatibles TTL. Lorsque l'entrée R du moniteur est au niveau haut, tandis que les deux autres sont au niveau bas, le point allumé sera rouge. Si c'est l'entrée V qui est au niveau haut et les deux autres au niveau bas, le point sera vert. Il en va de même pour le bleu.
Quant aux teintes obtenues par addition de ces trois primitives, elles sont les suivantes:
R + V : jaune
V + B : cyan (bleu clair)
R + B : magenta (violet)
Bien entendu, R + V + B = blanc et R = V = B = Ø = noir.
Le signal de chrominance n'existe donc pas en tant que tel. Nous sommes en présence de trois signaux de luminance binaires concernant chacun une des trois primitives R, V et B.
Les trois signaux sont issus de trois registres à décalage distincts, identiques chacun à celui de la **figure 3**. Ceux-ci sont alimentés à partir de trois bancs de mémoire parallèles; chacun de ces canaux est à l'image du canal unique que nous avions en N&B. Bien entendu, les signaux de synchronisation horizontale et verticale, le signal d'horloge-points et le signal de chargement des registres à décalage sont communs. A un pixel sur l'écran correspondent ainsi **trois bits qui ont la même adresse**, mais chacun dans un banc de mémoire distinct.
Supposons que l'écran est noir et que

3

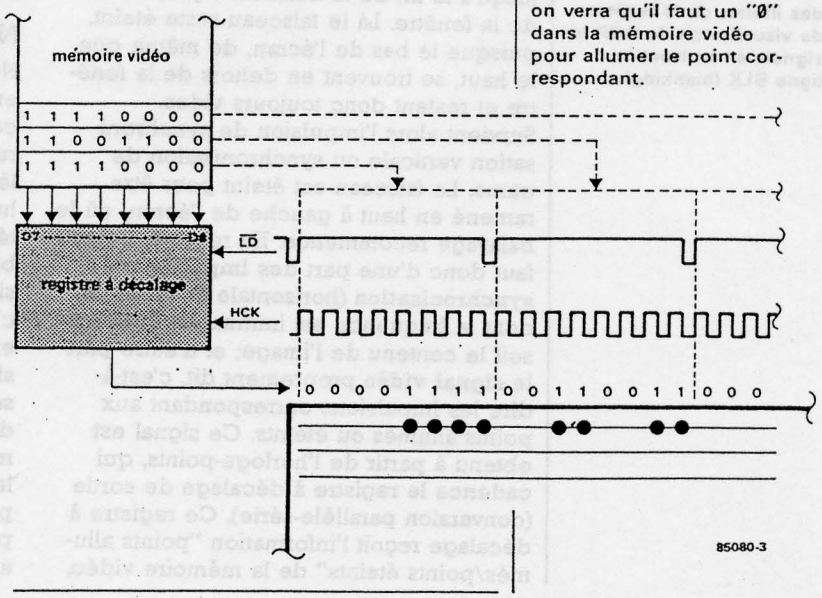


Figure 3. Chaque banc de mémoire vidéo alimente un registre à décalage comme celui-ci. Une horloge de sérialisation (HCK) cadence le décalage des huit bits dont le chargement est effectué chaque fois que passe au niveau logique bas la ligne $\overline{LD}$. Ici, un "1" logique correspond à un point allumé; en réalité, sur le schéma de la carte on verra qu'il faut un "Ø" dans la mémoire vidéo pour allumer le point correspondant.

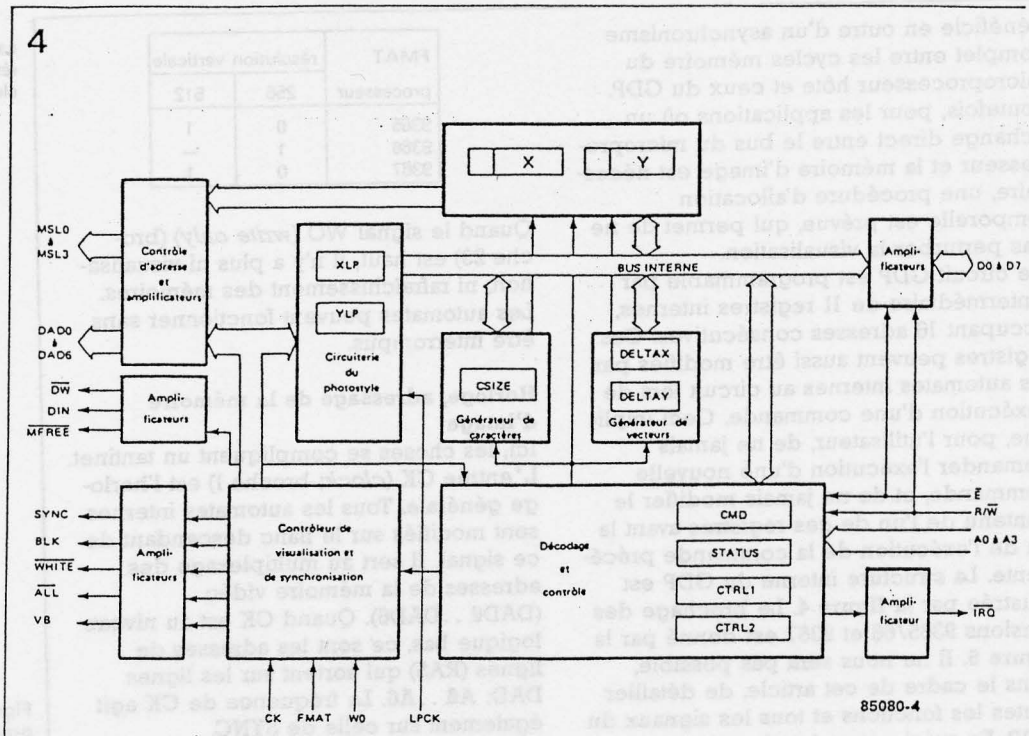


Figure 4. La structure interne d'un processeur graphique comme ceux de la famille 93XX est forcément complexe. Le protocole de communication avec l'utilisateur reste cependant très simple.

nous voulons en changer la couleur, par exemple en rouge. Il nous faut donc écrire dans la mémoire rouge, mais laisser les bits des mémoires verte et bleue comme ils sont, c'est-à-dire "éteints". Si nous voulons que l'écran devienne jaune, il nous faut "allumer" tous les bits des mémoires verte et rouge (rouge + vert = jaune, en vidéo!) et éteindre tous ceux de la mémoire bleue. Ce qui vaut ici pour l'ensemble de la mémoire et de l'écran vaut aussi pour chacun des points. A un point vert sur l'écran, correspond un bit "allumé" à l'adresse correspondante dans la mémoire verte, et un bit "éteint" dans les mémoires rouge et bleue. A un point magenta correspond un bit "allumé" à la bonne adresse dans les mémoires rouge et bleue, et un bit "éteint" dans la mémoire verte à cette adresse-là.
Sur le circuit, nous aurons donc une ligne de donnée commune à nos trois bancs de mémoire, de même qu'une ligne de validation des opérations d'écriture, mais aussi une ligne de sélection individuelle des bancs de mémoire. En pratique, cette sélection implique deux signaux: l'un doit déterminer **si l'on accède** ou non à un banc de mémoire, et l'autre doit déterminer **ce que l'on y fait**. En effet, lorsque l'on écrit en vert sur un fond rouge, il convient, aux endroits où l'on écrit, d'accéder au banc de mémoire rouge et au banc de mémoire vert. Dans ce dernier on allume les points, alors qu'aux mêmes adresses du banc de mémoire rouge on éteint les points. La même opération d'écriture a donc un effet différent dans les deux bancs de mémoire concernés: on allume des points ici, mais on les éteint là. Si l'on ne prenait pas cette mesure, il arriverait, dans notre exemple, qu'au lieu d'écrire en vert sur fond rouge, nous écrivions en jaune (rouge + vert) sur fond rouge. Prenez une feuille papier et essayez d'imaginer d'autres combinaisons possibles. Cela vous familiarisera avec la question...

Pour en finir avec ce paragraphe, il nous faut remarquer que pour des raisons d'ordre technique qu'il importe peu d'expliciter ici, un point allumé sur l'écran correspond, sur la carte graphique, à un bit de mémoire au niveau logique bas; tandis qu'un point éteint sur l'écran correspond à un bit de mémoire au niveau logique haut. C'est paradoxal, d'accord, mais c'est comme ça.

Le GDP de Thomson

Le processeur de visualisation graphique EF9365, EF9366 ou EF9367 possède toutes les fonctions nécessaires à la génération du signal vidéo et des signaux de synchronisation. Il se présente donc à l'utilisateur comme un contrôleur intelligent d'écran graphique vidéo, à balayage de trames programmable par un microprocesseur 8 bits. Outre les fonctions déjà citées, il comporte deux automates câblés d'écriture dans la mémoire d'image: un générateur de vecteurs et un générateur de caractères. Cette caractéristique permet d'atteindre une grande vitesse d'écriture (diagonale de l'écran: 512 points en moins de 700 μs), déchargeant ainsi le microprocesseur hôte de ces traitements élémentaires.
Ce circuit laisse l'espace mémoire adressable par le microprocesseur pratiquement intact, puisqu'il n'en occupe que 16 adresses. L'adjonction de registres extérieurs pour des fonctions spéciales telles que le *scroll*, la couleur, ou la commutation de pages, porte le nombre d'adresses mobilisées à une vingtaine à peine. On

bénéficie en outre d'un asynchronisme complet entre les cycles mémoire du microprocesseur hôte et ceux du GDP. Toutefois, pour les applications où un échange direct entre le bus du microprocesseur et la mémoire d'image est nécessaire, une procédure d'allocation temporelle est prévue, qui permet de ne pas perturber la visualisation.
Ce circuit GDP est programmable par l'intermédiaire de 11 registres internes, occupant 16 adresses consécutives. Ces registres peuvent aussi être modifiés par les automates internes au circuit lors de l'exécution d'une commande. Ceci implique, pour l'utilisateur, de ne jamais demander l'exécution d'une nouvelle commande, et de ne jamais modifier le contenu de l'un de ces registres avant la fin de l'exécution de la commande précédente. La structure interne du GDP est illustrée par la **figure 4**. Le brochage des versions 9365/66 et 9367 est donné par la **figure 5**. Il ne nous sera pas possible, dans le cadre de cet article, de détailler toutes les fonctions et tous les signaux du GDP. En voici cependant les plus importants.

Alimentation et niveaux logiques
La tension d'alimentation est de 5 V; toutes les entrées et sorties sont compatibles TTL. Les tensions d'entrée à l'état haut doivent être comprises entre 2,2 V et 5 V, et entre 0 V et 0,8 V pour l'état bas. Le courant d'alimentation typique est de 80 mA.

Bus du microprocesseur
L'ouverture des tampons d'entrée/sortie sur les lignes DØ...D7 (broches 33 à 26) est commandée par $\overline{E}$, le sens par R/$\overline{W}$. L'écriture correspond au niveau logique bas. $\overline{E}$ est le signal de synchronisation et le validation d'échange sur le bus.
L'adresse du registre concerné par accès en cours est appliquée aux lignes ØØ...A3 (broches 9 à 12). $\overline{IRQ}$ (broche 13) délivre une demande d'interruption, programmable par le registre CTRL1.

hotostyle
/HITE (broche 24) est prévu pour forcer niveau blanc sur le signal vidéo pour ermettre le repérage du photostyle (*light-en*). L'impulsion fournie par le photosty- est appliquée à l'entrée LPCK (*light pen ock*) (broche 21). L'adresse courante est iargée par le GDP dans les registres LP et YLP.

gnaux de synchronisation TV
synchronisation verticale et horizontale moniteur vidéo est obtenue à l'aide du gnal SYNC (broche 34). Celui-ci est aux rmes CCIR 625 lignes/50 Hz.
signal BLK (broche 25) force la suppression du signal vidéo en dehors de la nêtre de visualisation. Le signal VB (broe 16; *vertical blanking*) est au niveau rique haut pendant le retour trame.

ramétrage
ntrée FMAT (broche 8; format) doit e connectée comme indiqué ci-après:

FMAT	résolution verticale	
processeur	256	512
9365	0	1
9366	1	–
9367	0	1

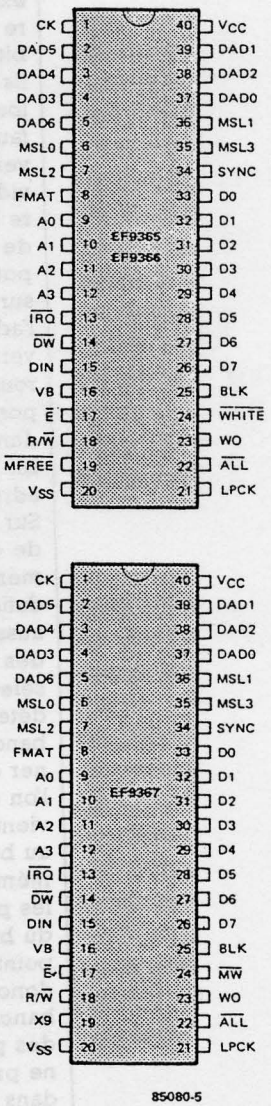
Quand le signal WO (*write only*) (broche 23) est haut, il n'y a plus ni visualisation, ni rafraîchissement des mémoires. Les automates peuvent fonctionner sans être interrompus.

Horloge, adressage de la mémoire d'image
Ici, les choses se compliquent un tantinet. L'entrée CK (*clock*; broche 1) est l'horloge générale. Tous les automates internes sont modifiés sur le flanc descendant de ce signal. Il sert au multiplexage des adresses de la mémoire vidéo (DADØ...DAD6). Quand CK est au niveau logique bas, ce sont les adresses de lignes (RAS) qui sortent sur les lignes DAD: AØ...A6. La fréquence de CK agit également sur celle de SYNC.
Les adresses de visualisation et de rafraîchissement apparaissent en deux passes sur les broches DADØ...6. L'espace mémoire maximal est de 16 K. Les lignes MSLØ...MSL3 (*memory select*) délivrent les signaux de sélection de pixel en écriture, grâce auxquels devient possible l'accès à un seul bit parmi les huit adressés via DADØ...6.
Lorsque la ligne $\overline{ALL}$ (broche 22) est au niveau logique bas, elle indique que l'accès en cours concerne tous les boîtiers de mémoire (accès collectif), par opposition à l'accès bit par bit tel qu'il est obtenu à l'aide des signaux MSL. **L'accès collectif a lieu notamment pour la visualisation (chargement des registres à décalage de sortie), le rafraîchissement ou l'effacement.**
La donnée à écrire dans la mémoire ne comporte qu'un bit issu de la broche 15 (DIN = *display in*). Au niveau logique haut, cette donnée correspond à un point éteint sur l'écran. Pour une application en N&B, DIN peut être directement la donnée d'entrée des mémoires. Pour une application en couleurs, il convient de combiner DIN avec le code de couleurs.
Le signal d'écriture dans la mémoire d'image est $\overline{DW}$ (broche 14; *display write*). Et enfin le signal $\overline{MFREE}$ (*memory free*) qui indique, lorsqu'il est au niveau logique bas que **le niveau logique du bit adressé par le microprocesseur** à l'aide d'une instruction spéciale, est présent en sortie de la mémoire concernée. Ce signal permet ainsi un échange quelconque avec la case mémoire pointée par les registres X et Y (programmés au préalable par l'utilisateur), sans perturber la visualisation. Il répond toujours à une demande externe d'accès à la mémoire d'image.

Ultérieurement, nous aurons à revenir sur ces signaux. Il nous faudra aussi, lorsqu'il sera question du logiciel, parler des registres du GDP, du générateur de vecteurs,

Figure 5. Brochage des processeurs 9365, 9366 et 9367. A deux broches près, ils sont parfaitement compatibles.

5



85080-5

6

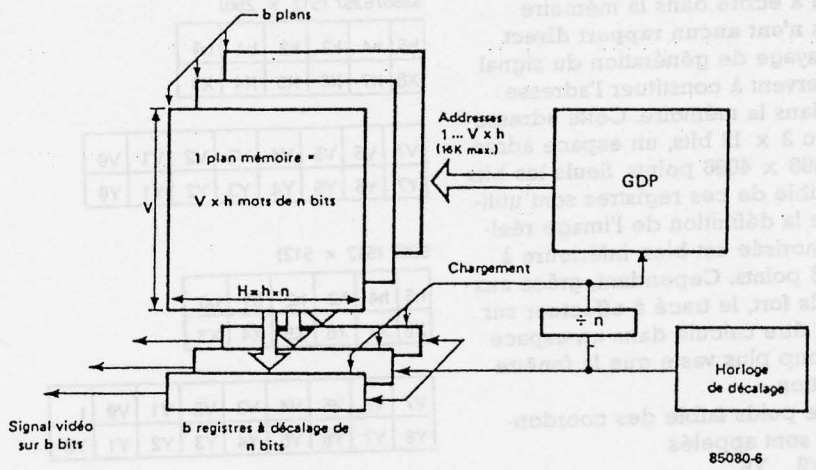


Figure 6. La structure de l'image, c'est-à-dire sa résolution horizontale et verticale, est étroitement liée à celle de la mémoire vidéo. Il faut noter que dans ce circuit, la résolution de l'image n'exerce aucune influence sur le nombre de couleurs disponibles, et vice versa. En outre, la mémoire vidéo est totalement autonome, c'est-à-dire qu'elle ne mobilise pas même un octet de la mémoire du système avec lequel la carte graphique est utilisée.

etc. En attendant, nous en resterons à l'aspect matériel pour tenter d'éclairer les principes de l'organisation de la mémoire d'image.

La mémoire d'image

Considérons une image de **V** x **H** pixels et supposons que chaque pixel puisse prendre 2^b états différents; il faut donc une mémoire d'image de **V** x **H** x **b** bits, où **V** est le nombre de lignes de balayage utile, **H** le nombre de bits par ligne horizontale et **b** le nombre de couleurs primitives (1 pixel peut donc comporter plusieurs bits — le plus souvent trois et parfois quatre). Dans les applications où H est grand, la fréquence du signal vidéo est supérieure à la fréquence maximale d'accès en lecture des mémoires. Quand par exemple H = 512 avec une fréquence de balayage télévision, la période de succession des pixels est de l'ordre de 70 ns. Il est donc indispensable de découper une ligne **H** en **h** tronçons de **n** bits adjacents, lus simultanément dans la mémoire d'image, et sérialisés par un dispositif autonome (horloge-points + registre à décalage) pour constituer le signal vidéo. Il faut donc **h** accès à la mémoire par ligne. Chaque accès charge **b** registres à décalage de **n** bits chacun. La mémoire contient **V** x **h** x **b** mots de **n** bits (voir **figure** 6). Dans le schéma qui nous occupe ici, les formats possibles sont les suivants:

b: théoriquement illimité; en pratique b = 1...4 (RGB+I)
n: 8 ((64 k x 1 bit) x 8)
H: 512
V: 256 ou 512

Rappelons que pour l'adressage individuel des bits, nous disposons, outre les signaux DAD0...DAD6 qui adressent les mots **h** de **n** bits chacun, des lignes MSL qui sélectionnent un pixel de **b** bits dans le mot **h** adressé. Comme n = 8, les lignes MSL doivent être au nombre de 3: MSL0...2. La ligne MSL3 est utilisée en mode entrelacé (V = 512) et permet de différencier la trame paire de la trame impaire. On l'utilise pour commander la ligne A7 de circuits 4164.

Rafraîchissement et visualisation

Nous avons vu que le GDP effectuait sur la mémoire trois types d'opérations fondamentalement différentes. La visualisation (V) qui débouche sur la sérialisation du contenu de la mémoire et son affichage sur l'écran, l'écriture (E) et le rafraîchissement (R). En dehors de la fenêtre, où la mémoire est utilisée exclusivement en visualisation (et rafraîchissement) il peut y avoir écriture, sauf pendant trois périodes de rafraîchissement (R), comme indiqué sur la **figure 7**. Les trois types d'opérations sont indiqués à l'extérieur du circuit par les signaux BLK et $\overline{ALL}$:

	BLK	$\overline{ALL}$
V	0	0
E	1	1
R	1	0

Nous verrons dans l'étude des registres du GDP qu'il existe également des situations d'exception.

7

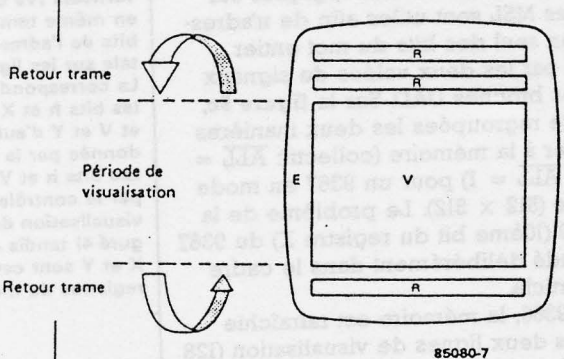


Figure 7. Il faut s'imaginer le grand rectangle aux bords arrondis (E) comme l'espace-temps dans lequel agit le GDP. Outre les périodes de rafraîchissement (R) qui ont lieu pendant le retour trame, la visualisation occupe l'essentiel de cet espace-temps. Ce qui reste comme temps de part et d'autre de la fenêtre et entre les périodes R, est réservé aux opérations d'écriture dans la mémoire vidéo.

Correspondance entre les sorties DAD/MSL et les coordonnées X/Y

Les registres X et Y du GDP sont des

registres de 12 bits (lecture et écriture) qui contiennent les coordonnées du **prochain** point à écrire dans la mémoire d'image. Ils **n'ont aucun rapport direct** avec le balayage de génération du signal vidéo; ils servent à constituer l'adresse d'écriture dans la mémoire. Cette adresse couvre, avec 2 x 12 bits, un espace adressable de 4096 x 4096 points. Seuls les bits de poids faible de ces registres sont utilisés puisque la définition de l'image réellement mémorisée est bien inférieure à 4096 x 4096 points. Cependant, grâce aux bits de poids fort, le tracé à effectuer sur l'écran peut être calculé dans un espace fictif beaucoup plus vaste que la fenêtre de visualisation.

Les 9 bits de poids faible des coordonnées X et Y sont appelés
X0...X8 et Y0...Y8.

Les compteurs internes qui génèrent les adresses de la mémoire d'image sont organisés comme suit:

- il y a 6 bits d'adresse horizontale (h = 64 mots de n bits) h0, h1, h2, h3, h4, h5
- et 9 bits d'adresse verticale (V = 256 ou V = 512)

t, V0, V1, V2, V3, V4, V5, V6, V7

où t est le bit de poids le plus faible; il représente la parité des trames lorsque V = 512. Lorsque V = 256, t n'existe pas.

La correspondance entre adresse de visualisation (bits h, V et t) et adresse d'écriture (bits X et Y) est donnée par la **figure 8**. Nous excluons ici délibérément le processeur 9365 qui permet certes le fonctionnement en mode entrelacé aussi bien qu'en mode non entrelacé, mais impose toujours une résolution verticale **égale** à la résolution horizontale (V = H). Le 9366 ne connaît pas le mode entrelacé (V = 256). Le circuit le plus intéressant est le plus récent. Il s'agit du 9367, avec lequel on peut pratiquer le 512 x 256 aussi bien que le 512 x 512. On constate qu'en mode non entrelacé, ces deux processeurs sont interchangeables.

Jusqu'ici tout est clair. Par contre, lorsqu'il s'agit d'attribuer ces bits d'adresse de visualisation et d'écriture aux broches de sorties du processeur (DAD et MSL), ça se corse. Pour tenter de simplifier les choses, nous avons différencié, sur la **figure 9**, les accès collectifs des accès pixel par pixel. Commençons par le mode 512 x 256 (non entrelacé), commun aux deux processeurs. Sur la **figure 9a**, c'est un accès collectif ($\overline{ALL}$ = 0). Dans la première moitié du cycle d'accès (CK = 0), en mode de visualisation, les lignes DAD0...DAD6 sortent les 6 bits de l'adresse horizontale h0...h5), mais aussi le bit de poids faible de l'adresse verticale (V0 sur DAD6)! Ce petit détail est important pour le fonctionnement du circuit *scroll*.

Au cours de la seconde moitié du cycle l'accès, toujours en mode visualisation, les lignes DAD fournissent le reste des bits de l'adresse verticale. Pendant ce temps, **les niveaux logiques présents sur les lignes MSL n'ont aucune signification puisque nous sommes en présence d'un accès collectif.**

8

9366/9367 (512 x 256)

h5	h4	h3	h2	h1	h0
X8	X7	X6	X5	X4	X3

V7	V6	V5	V4	V3	V2	V1	V0
Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0

9367 (512 x 512)

h5	h4	h3	h2	h1	h0
X8	X7	X6	X5	X4	X3

V7	V6	V5	V4	V3	V2	V1	V0	t
Y8	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0

Figure 8. Une coordonnée sur l'écran ne fait pas une adresse dans la mémoire. A chaque bit des registres X et Y correspond un bit d'adressage horizontal h ou vertical V, sauf aux trois bits de poids faible du registre X. Ceux-là sont utilisés pour distinguer entre eux les huits bits de l'octet adressé à l'aide des bits h et V.

9

9366/9367 (512 x 256)

Figure 9a

		MSL				DAD						
$\overline{ALL}$	CK	0	1	2	3	0	1	2	3	4	5	6
0	0	X	X	X	1	h5	h4	h3	h2	h1	h0	V0
0	1	X	X	X	1	V7	V6	V5	V4	V3	V2	V1

X = don't care

Figure 9b

		MSL				DAD						
ALL	CK	0	1	2	3	0	1	2	3	4	5	6
1	0	X0	X1	X2	1	X8	X7	X6	X5	X4	X3	Y0
1	1	X0	X1	X2	1	Y7	Y6	Y5	Y4	Y3	Y2	Y1

9367 (512 x 512)

Figure 9c

		MSL				DAD						
$\overline{ALL}$	CK	0	1	2	3	0	1	2	3	4	5	6
0	0	X0	X1	X2	V1	h5	h4	h3	h2	h1	h0	V0
0	1					V7	V6	V5	V4	V3	V2	t
1	0	X0	X1	X2	Y2	X8	X7	X6	X5	X4	X3	Y1
1	1					Y8	Y7	Y6	Y5	Y4	Y3	Y0

Sur la **figure 9b**, nous sommes en présence d'un accès pixel par pixel. Les trois bits de poids faible du registre X apparaissent donc sur les lignes MSL pour opérer la sélection de l'un des huit bits adressés par les lignes DAD. Là encore, le bit de poids faible de l'adresse verticale apparaît dans la première moitié du cycle d'accès, avec les bits de l'adresse horizontale. Cette fois les niveaux logiques sur les lignes MSL sont utiles afin de n'adresser qu'un seul des bits du mot entier adressé par les deux volées de signaux issus des broches DAD. Sur la **figure 9c**, on trouve regroupées les deux manières d'accéder à la mémoire (collectif: $\overline{ALL}$ = 0; pixel: $\overline{ALL}$ = 1) pour un 9367 en mode entrelacé (512 x 512). Le problème de la sortie X9 (10ème bit du registre X) du 9367 a été éludé délibérément dans le cadre de cet article.

Avec le 9366, la mémoire est rafraîchie toutes les deux lignes de visualisation (128 accès). Avec le 9367, elle l'est toutes les quatre lignes (256 accès).

Figures 9a...9c. Pour le multiplexage des adresses, celles-ci sont distribuées sur les lignes DAD et MSL d'une manière qui, à première vue, n'est pas facile à comprendre. En 9a, il s'agit d'un accès collectif, alors qu'en 9b c'est un accès bit par bit. On notera que dans un cas somme dans l'autre, un des bits de l'adresse verticale (V0 ou Y0) sort en même temps que les bits de l'adresse horizontale sur les lignes DAD. La correspondance entre les bits h et X d'une part, et V et Y d'autre part, est donnée par la figure 8. Les bits h et V sont gérés par le contrôleur de visualisation du GDP (figuré 4) tandis que les bits X et Y sont ceux des registres du même nom.

La mémoire vive dynamique

Pour bien comprendre le fonctionnement du circuit et saisir l'importance d'une chronologie rigoureuse de tous les signaux, il importe de connaître les principes essentiels d'une mémoire vive dynamique comme celle que l'on utilise ici.

Chacun des huit circuits intégrés du type 4164 comporte 65 536 cellules de mémoire de 1 bit. Le contenu d'une telle cellule doit être rafraîchi périodiquement, à défaut de quoi il se perd. Les constructeurs indiquent une période maximale de rafraîchissement de 2 ms. Ce qui signifie que toutes les cellules de la mémoire doivent avoir été rafraîchies en l'espace de 2 ms. Pour obtenir ce rafraîchissement, une simple opération de lecture suffit.

En principe, pour accéder à chacune des 65 536 cellules, il faut 16 lignes d'adresses distinctes. Cela signifierait-il qu'un circuit intégré de ce type compte 16 broches pour les adresses, deux pour l'alimentation, une pour les données, etc? Où irions-nous?

Fort heureusement, les 64 K × 1 bit sont organisés en matrice à adressage multiplexé. C'est-à-dire que l'on donne d'abord l'adresse de la ligne de la matrice (*row adress*) sur laquelle se trouve la cellule à laquelle on souhaite accéder, puis, sur les mêmes broches du circuit intégré, l'adresse de la colonne sur laquelle se trouve cette cellule. De sorte que l'on peut se contenter de huit broches d'adresses, plus deux broches sur lesquelles on appliquera les signaux d'échantillonnage, indiquant l'un que les adresses présentes sur les broches du circuit intégré sont des addresses de ligne de la matrice, et l'autre que ces adresses concernent une colonne de la matrice. Ces signaux sont appelés $\overline{RAS}$ et $\overline{CAS}$ (*row address strobe* = impulsion de validation de l'adresse de ligne; *column address strobe* = impulsion de validation de l'adresse de colonne). Comme ces signaux sont actifs au niveau logique bas, ils sont surmontés de la barre d'inversion.

Pour le rafraîchissement, il n'est pas nécessaire d'adresser les 65536 cellules individuellement. On parle de rafraîchissement collectif. Il suffit d'adresser les lignes de la matrice qui sont au nombre de 256. Et comme les fabricants de circuits intégrés ont fait l'effort de concevoir des circuits de mémoire de 64 K compatibles avec leurs prédécesseurs de 16 K qui ne comptaient que 128 lignes et 128 colonnes (128 × 128 = 16 384 cellules), nous disposons même de circuits de 64 K qui se contentent d'un rafraîchissement sur 128 lignes au lieu de 256. Lorsque l'on consulte les fiches de caractéristiques des circuits intégrés, on reconnaît ce type de mémoires à la particularité suivante: le constructeur indique *"128 refresh cycles/2 ms"*, et le diagramme des signaux de rafraîchissement comporte la mention *"A7: don't care"*.

La chronologie des signaux $\overline{RAS}$ et $\overline{CAS}$ avec les signaux d'adresses correspondants est extrêmement rigoureuse. Lors d'une opération de lecture ou d'écriture, elle est la suivante (voir **figures 10 et 11**):

Le premier signal à devenir actif est $\overline{RAS}$ (broche 4 d'un 4164); le circuit intégré prend en compte les huit niveaux logiques présents sur les broches A0...A7 à ce moment précis, et il en fait l'adresse de ligne de la matrice. Un peu plus tard, alors que $\overline{RAS}$ reste actif au niveau logique bas, c'est le signal $\overline{CAS}$ qui devient actif. A ce moment, le circuit intégré prend en compte comme adresse de colonne les huit niveaux logiques présents sur les broches A0...A7: ce sont en fait les bits d'adresse A8...A15 dans l'espace mémoire de 64 K. Il aura donc fallu, après $\overline{RAS}$ et avant $\overline{CAS}$, appliquer sur les broches A0...A7 les bits A8...A15. On notera cependant qu'avec les RAM dynamiques, il n'est pas nécessaire que les signaux d'adresses (A0...A7) soient établis avant l'impulsion de validation $\overline{RAS}$ ou $\overline{CAS}$ (t_{ASR} et t_{ASC} peuvent être nuls!).

S'il s'agit d'une opération de lecture, quelques dizaines de nanosecondes après l'échantillonnage des signaux d'adresse, le niveau logique du bit adresse apparaît sur la broche D_{out} du circuit intégré. S'il s'agit d'une opération d'écriture dans la mémoire, il aura fallu activer la ligne $\overline{WRITE}$ (broche 3 du 4164) et appliquer sur l'entrée D_{in} le niveau logique du bit adressé, un peu avant d'activer $\overline{CAS}$.

Pour le rafraîchissement, il suffit d'appliquer l'adresse de ligne et d'activer $\overline{RAS}$, en respectant le rapport cyclique imposé au rafraîchissement d'une part (c'est-à-dire que les 128 lignes doivent avoir été adressées chacune toutes les 2 ms), mais aussi le rapport cyclique imposé au signal $\overline{RAS}$ lui-même. En effet, avant que $\overline{RAS}$ ne devienne actif au niveau logique bas et puisse le rester pour un temps t_{RAS}, il faut qu'il soit resté au niveau logique haut pendant un minimum t_{RP} — *precharge time* — qui est en quelque sorte une pause nécessaire au circuit intégré pour..."retrouver ses esprits"...! Cette révision ou découverte (pour certains) des principes de fonctionnement de la RAM dynamique était nécessaire, comme on va le voir, pour aborder un mode de fonctionnement très particulier de la carte graphique. Ce mode fait appel à une caractéristique extraordinaire des RAM dynamiques, puisque l'on effectue, au cours du même cycle d'accès, une opération de lecture et d'écriture à une adresse donnée. Et lorsque l'on saura qu'en plus, entre la lecture de la donnée et l'écriture, prend place une opération de modification de cette donnée, on aura compris que l'affaire est périlleuse. Il s'agit du mode "lecture-modification-écriture" appelé RMW-mode.

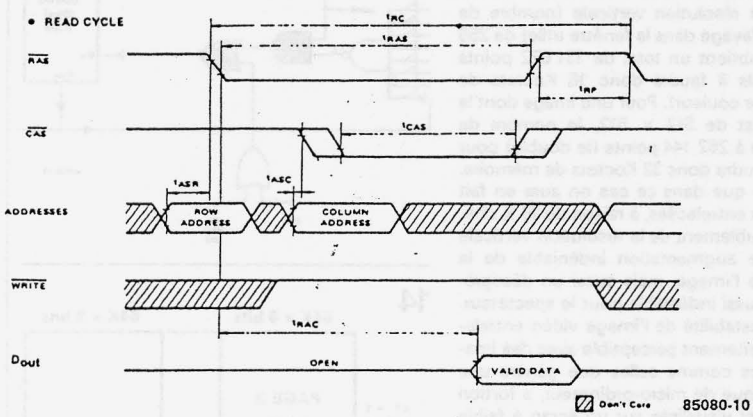
Read/Modify/Write mode

Pour commencer, rappelons que toute chose en ce bas monde, aussi compliquée fût-elle, n'est jamais qu'une combinaison plus ou moins obscure de choses simples...*Ad augusta per angusta*, chère Augustine.

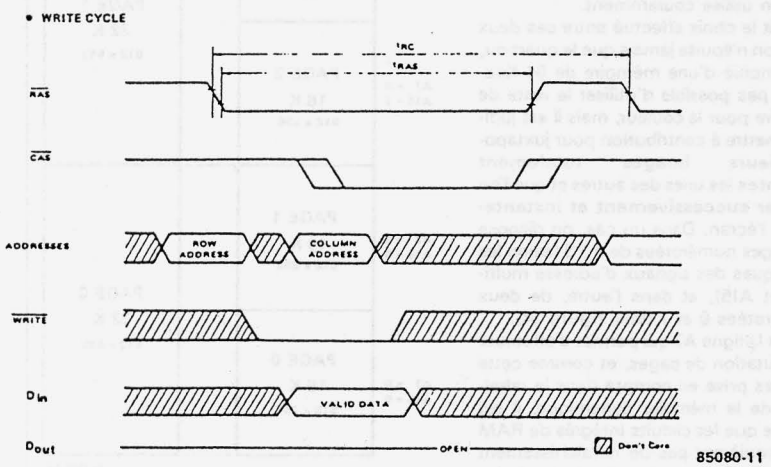
Dans le paragraphe "Noir&Blanc ≠ couleurs", nous avons vu comment les niveaux logiques de trois bits dans trois bancs de mémoire donnaient huit combinaisons de couleurs selon leurs niveaux logiques. Nous avons également noté qu'accessoirement, et pour des raisons techniques, un point allumé sur l'écran correspondait à un niveau logique bas dans la mémoire, tandis qu'un niveau logique haut donne lieu à l'extinction du point correspondant sur l'écran. Abandonnons un instant la notion de couleur pour revenir au Noir&Blanc, afin de simplifier la question du mode de lecture/modification/écriture.

Lorsque l'on a un dessin (un fond) sur lequel se déplace un objet, cet objet efface forcément la partie du fond sur laquelle il se trouve; il convient donc, à chaque déplacement de l'objet, de reconstituer le fond tel qu'il était avant l'apparition de l'objet à cet endroit. Cette opération est complexe, et demande par conséquent un temps d'exécution relativement long. C'est pour contourner ce gros obstacle que l'on fait appel au mode que nous nous proposons d'étudier ici. Pour le mettre en oeuvre, il suffit de quelques opérateurs logiques élémentaires supplémentaires (une porte NAND à 8 entrées, deux portes AND et deux portes OR comme on les voit sur la **figure 12**). Nous savons que pour écrire dans la mémoire vidéo, il nous faut valider le signal d'écriture ($\overline{WRITE}$) et définir le niveau logique de la donnée à écrire (D_{in}). En mode RMW on définit le niveau logique de ce bit non seulement en fonction de l'effet souhaité (point allumé ou éteint), mais **aussi en fonction de l'état antérieur de ce bit!** Si le point était allumé, nous allons l'éteindre. S'il était éteint, nous allons l'allumer. A présent le point est apparu sur le fond. Si le fond était noir, le point est apparu en blanc; si le fond était blanc, le point est apparu en noir. Et ce qui est formidable, c'est que pour effacer ce point et restituer le fond dans son état antérieur à l'apparition du point, il va suffire de **redessiner le point au même endroit.** S'il était allumé, il s'éteint, s'il était éteint, il s'allume, se confondant ainsi avec le fond (c'est le cas de le dire!). Ce qui vaut ici pour un seul

10



11



point vaut également pour tout dessin (qui n'est rien d'autre qu'un ensemble de points) aussi complexe soit-il. En résumé, lorsque l'on dessine un objet sur un fond, on procède en fait à l'inversion des points du fond à l'endroit où doit apparaître l'objet; puis, pour reconstituer le fond et faire disparaître l'objet, on refait cette opération une seconde fois. Et comme nous le savons, deux inversions successives s'annulent. On peut se représenter cela sous la forme d'un OU-Exclusif (EXOR) effectué entre la plume et le papier: là où le papier est blanc, la plume sera noire; là où le papier est noir, la plume sera blanche.

Un cycle RMW commence par un accès en lecture et se termine par un accès en écriture. Entre les deux, le bit adressé est modifié.

La porte NAND à entrées multiples restitue le niveau logique du seul bit adressé (les sept autres sont forcés au niveau logique haut) **inversé**. Nous l'appelons ici Σ. Le niveau logique de la ligne RMWS (*read/modify/write select*) indique, lorsqu'il est haut, que le mode RMW est actif. Lorsque cette ligne est au niveau logique bas, le bit de la ligne DIN venant du processeur graphique est chargé dans la mémoire d'image grâce à l'impulsion $\overline{DW}$ (*data write*). On ne tient donc pas compte de l'état du fond sur lequel on dessine, puisque l'on n'est pas en mode RMW. La combinaison du signal RMWS et du signal $\overline{LD}$ est nécessaire pour inhiber, en mode RMW, la procédure d'écriture, amorcée par $\overline{DW}$, jusqu'à l'apparition, en sortie de la porte NAND, du bit lu dans la mémoire, filtré et inversé par la porte NAND à 8 entrées et renvoyé sur l'entrée Din de la RAM concernée. Cette combinaison est donc motivée par des problèmes de chronologie des signaux.

Si le bit Σ est haut, c'est que dans la mémoire il était bas: d'un point allumé (niveau bas dans la mémoire), nous faisons un point éteint (niveau haut sur D_{in}), quel que soit le niveau logique que l'on veut écrire (DIN). Si le bit Σ est bas, c'est que le point que nous adressons était éteint (niveau haut dans la mémoire). Le fond, à cet endroit, est noir. Si à présent il y a lieu d'allumer un point, la ligne DIN est basse: ce niveau logique est chargé par D_{in}. Si à cet endroit du dessin il n'y a pas lieu d'allumer un point, la ligne DIN est haute, et le niveau logique haut du bit correspondant est et reste haut dans la mémoire (point éteint).

Voilà ce qui se cachait derrière l'effrayant vocable RMW... Si vous n'en avez pas saisi tous les détails du premier coup, ne vous inquiétez pas, vous n'êtes pas seul dans ce cas. Et de toutes façons, l'essentiel est que ça marche.

Or, ça marche, à condition que le retard introduit par la porte NAND et les portes AND et OR ne compromettent pas la chronologie du cycle de lecture/modification/écriture de la RAM. Plus le temps t_{RAC} (à compter du début de $\overline{RAS}$ jusqu'à l'apparition de la donnée sur D_{out}) est court (ce temps défini comme temps d'accès est marqué sur le boîtier du circuit intégré), plus il reste de temps à la logique RMW pour modifier la donnée et la transférer vers l'entrée D_{in} dans les délais, c'est-à-dire avant l'arrivée de l'impulsion $\overline{WRITE}$. Les calculs indiquent que le temps d'accès ne devrait pas excéder 150 ns (NEC4164-3, Hitachi 4864-2, Toshiba 4164-3, OKI 3764-15, MOSTEK 4564-15, etc). Mais l'expérience a montré que dans de nombreux cas, le circuit RMW donnait satisfaction même avec des temps d'accès typiques de 300 ns...

Sur la **figure 13**, nous trouvons un circuit RMW typique pour trois bancs de mémoire-couleur. En grisé, ce sont les portes spécifiques à ce mode. Elles sont au nombre de sept, mais l'une d'entre elles est même réutilisée pour une autre fonction dont nous reparlerons. En tout état de cause, le rendement de l'adjonction de ces quelques portes est élevé. Leur présence n'est guère gênante dans un circuit de toutes façons très touffu; elle ne pose de problèmes qu'au niveau du retard introduit par elles dans le trajet du bit de donnée de la sortie d'une mémoire vers l'entrée.

Le circuit reste en fait le même qu'en N&B, à ceci près qu'il faut rajouter ici les signaux de sélection de couleur. Ce sont RS-GS-BS (pour *red select*, *green select* et *blue select*) et RWS-GWS-BWS (pour *red write select*, *green write select* et *blue write select*). Contrairement à DIN et $\overline{DW}$ qui sont issus du GDP, les signaux mentionnés ci-dessus sont commandés par l'utilisateur via le registre COLOR.

Si RS, GS ou BS sont au niveau logique haut, le point adressé sera éteint dans la couleur correspondante, à condition que la ligne RWS, GWS ou BWS permette par son niveau logique bas, l'accès à ce banc de mémoire. Si RS, GS ou BS sont au niveau logique bas, le point adressé sera allumé dans la couleur correspondante, à condition que la ligne RWS, GWS ou BWS permette par son niveau logique bas, l'accès à ce banc de mémoire. Le **tableau 2** donne les combinaisons possibles avec et sans RMW, mais en N&B uniquement. Nous reviendrons là-dessus pour la couleur lorsqu'il en sera question pour de bon.

Tableau 2

RMWS	$\overline{(DW = \emptyset)}$ Dout = Σ	DIN	Din/points
Ø	X	1	point éteint
Ø	X	Ø	point allumé
1	Ø	Ø	point allumé (il était éteint)
1	Ø	1	point éteint (il était éteint)
1	1	X	point éteint (il était allumé)

Capacité de la mémoire et résolution de l'image

Si l'on considère une image dont la résolution horizontale (nombre de points d'une ligne de balayage dans la fenêtre utile) est de 512 points, et la résolution verticale (nombre de lignes de balayage dans la fenêtre utile) de 256 points, on obtient un total de 131 072 points pour lesquels il faudra donc 16 Koctets de mémoire (par couleur). Pour une image dont la résolution est de 512 × 512, le nombre de points passe à 262 144 points (le double) pour lesquels il faudra donc 32 Koctets de mémoire. Il faut noter que dans ce cas on aura en fait deux images entrelacées, à raison de 16 K chacune. Le doublement de la résolution verticale apporte une augmentation indéniable de la définition de l'image, mais aussi un désagrément tout aussi indéniable pour le spectateur. En effet, l'instabilité de l'image vidéo entrelacée est parfaitement perceptible avec des images statiques comme celles que génère une carte graphique de micro-ordinateur, a fortiori lorsqu'elle est visualisée sur un écran à faible rémanence comme celui des moniteurs couleurs que l'on utilise couramment.

Quel que soit le choix effectué entre ces deux résolutions, on n'épuise jamais que le quart ou, au plus, la moitié d'une mémoire de 64 Koctets. Il n'est pas possible d'utiliser le reste de cette mémoire pour la couleur, mais il est judicieux de la mettre à contribution pour juxtaposer **plusieurs images totalement indépendantes** les unes des autres et que l'on peut **afficher successivement et instantanément** sur l'écran. Dans un cas, on dispose de quatre pages numérotées de Ø à 3 (selon les niveaux logiques des signaux d'adresse multiplexés A7 et A15), et dans l'autre, de deux pages numérotées Ø et 1 (voir **figure 14**).

Comme c'est la ligne A7 qui permet d'effectuer cette commutation de pages, et comme cette ligne n'est pas prise en compte dans le rafraîchissement de la mémoire dynamique, il est indispensable que les circuits intégrés de RAM utilisés ne requièrent pas de rafraîchissement sur A7 et A15. Sont exclus par exemple les HYB4164 de Siemens.

Figure 10. Chronologie des signaux du cycle de lecture d'une RAM dynamique 4164.

Figure 11. Chronologie des signaux d'écriture du cycle d'écriture d'une RAM dynamique 4164.

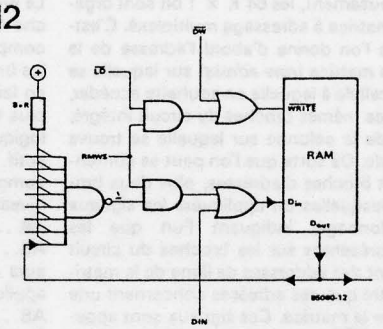
Figure 12. La logique RMW en N&B.

Figure 13. La logique RMW en couleurs.

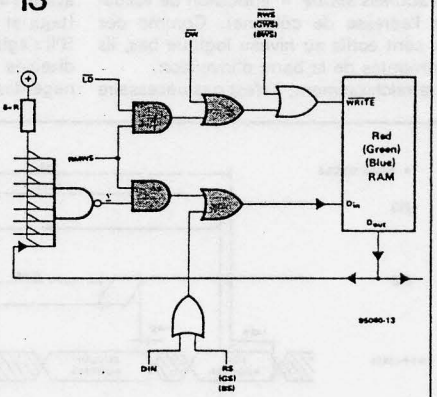
Figure 14. Cartographie de la mémoire en mode non entrelacé (à gauche) et en mode entrelacé (à droite).

Tableau 2. Table de vérité de la logique RMW et N&B.

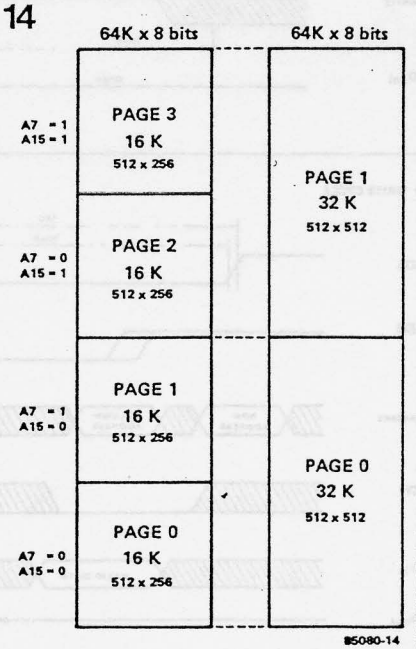
12



13



14



Le circuit — figure 15

Le décodage d'adresses

Les huit lignes d'adresses de poids fort A8...A15 sont appliquées aux entrées P du comparateur à 8 bits IC1. Ses entrées Q sont polarisées au moyen de résistances et forcées au niveau logique bas par des interrupteurs lorsque ceux-ci sont fermés. Quand l'octet d'adresse de poids fort présent sur le bus d'adresses est identique au mot binaire programmé par l'utilisateur sur S1...S8, la sortie $\overline{P=Q}$ d'IC1 passe au niveau logique bas, autorisant ainsi IC2 à décoder les lignes d'adresses A4...A7. Comme A7 est relié à l'entrée de validation $\overline{G2A}$ d'IC2, le 74LS138 ne sera activé que pour les adresses comprises entre XX0Y et XX7Y, où XX est défini par IC1 (A8...A15) et Y par IC3. Notez au passage la présence de Φ 2 sur l'entrée de validation G1 d'IC2. Ceci garantit la synchronisation du décodage d'adresses avec la chronologie des signaux d'adresses et de données du 6502. Un circuit compatible avec le bus du Z80 fera l'objet d'une publication ultérieure.

Seules deux des huit sorties d'IC2, définissant chacune un bloc de seize adresses, sont utilisées. L'une pour activer l'entrée $\overline{E}$ (*enable*) du GDP (qui est donc décodé entre XX50 et XX5F), et l'autre pour activer IC3, qui se charge des seize octets entre XX60 et XX6F. La première adresse utilisée dans ce bloc est XX64. La présence de la ligne R/$\overline{W}$ (*read/write*) sur l'entrée A d'IC3 (au niveau logique bas lors des opérations d'écriture et au niveau logique haut lors des opérations de lecture) permet de faire des économies. On obtient deux signaux de décodage différents à la même adresse, selon que l'opération en cours est de lecture ou d'écriture. Ainsi, lorsque l'on écrit en XX64, c'est dans IC12 (FF1, FF2); lorsque l'on lit à l'adresse XX64, c'est dans IC13. A l'adresse XX65 (rappelons que c'est l'utilisateur qui à l'aide de S1...S8, définit l'octet XX), on trouve le registre IC6, à écriture seule. L'adresse de lecture XX65 n'est pas utilisée. A l'adresse XX66 se trouve le registre IC11, à écriture seule également. Là non plus, le signal de décodage en mode lecture n'est pas utilisé.

Lorsque le GDP IC5 est activé, le tampon de données IC4 l'est aussi; le sens de transfert (bus de données → GDP pour les opérations d'écriture; ou GDP → bus de données pour les opérations de lecture) est défini par le niveau logique de la ligne R/$\overline{W}$ du 6502, appliquée à la broche 1 d'IC4. Les autres registres de la carte graphique communiquent directement avec le bus de données. Il y a à cela une raison bien précise que nous évoquerons à propos des circuits de couleur. Contentons-nous de le noter pour l'instant. On trouvera une récapitulation du décodage d'adresses dans le **tableau 3.**

Le GDP et les signaux de commande

Avec ses 11 registres adressables et accessibles depuis le bus du microprocesseur, le GDP est la vedette de cette distribution. Son fonctionnement paraît compliqué, au début notamment, et à juste titre d'ailleurs, car ce n'est pas un circuit intégré simple.

Sur les broches DAD0...DAD6, il sort les adresses des octets auxquels il lui faut accéder, pour y lire (visualisation), y écrire ou pour les rafraîchir. Comme il convient, le premier mot d'adresse apparaissant sur ces broches est A0...A6, c'est-à-dire l'adresse de ligne de la mémoire dynamique (RAS). Le deuxième mot d'adresse arrive un instant après, et c'est A8...A14 de la RAM. Pour l'instant, il faut voir le circuit comme si IC8 et IC9 n'existaient pas, et comme si les sorties DAD du GDP étaient reliées directement aux entrées A0...A6 de la mémoire.

L'apparition des mots d'adresse multiplexés sur les broches DAD est cadencée par le signal CK. C'est à partir de lui qu'est construite toute la chronologie de la carte. Lui-même est dérivé de l'horloge-points générée par l'oscillateur à quartz construit autour de N16...N18. La fréquence de cet oscillateur est de 12 MHz avec le 9367 et de 14 MHz avec le 9366. Pour l'utilisateur, cela se traduit sur l'écran par une image un peu plus large dans le premier cas que dans le second. C'est tout.

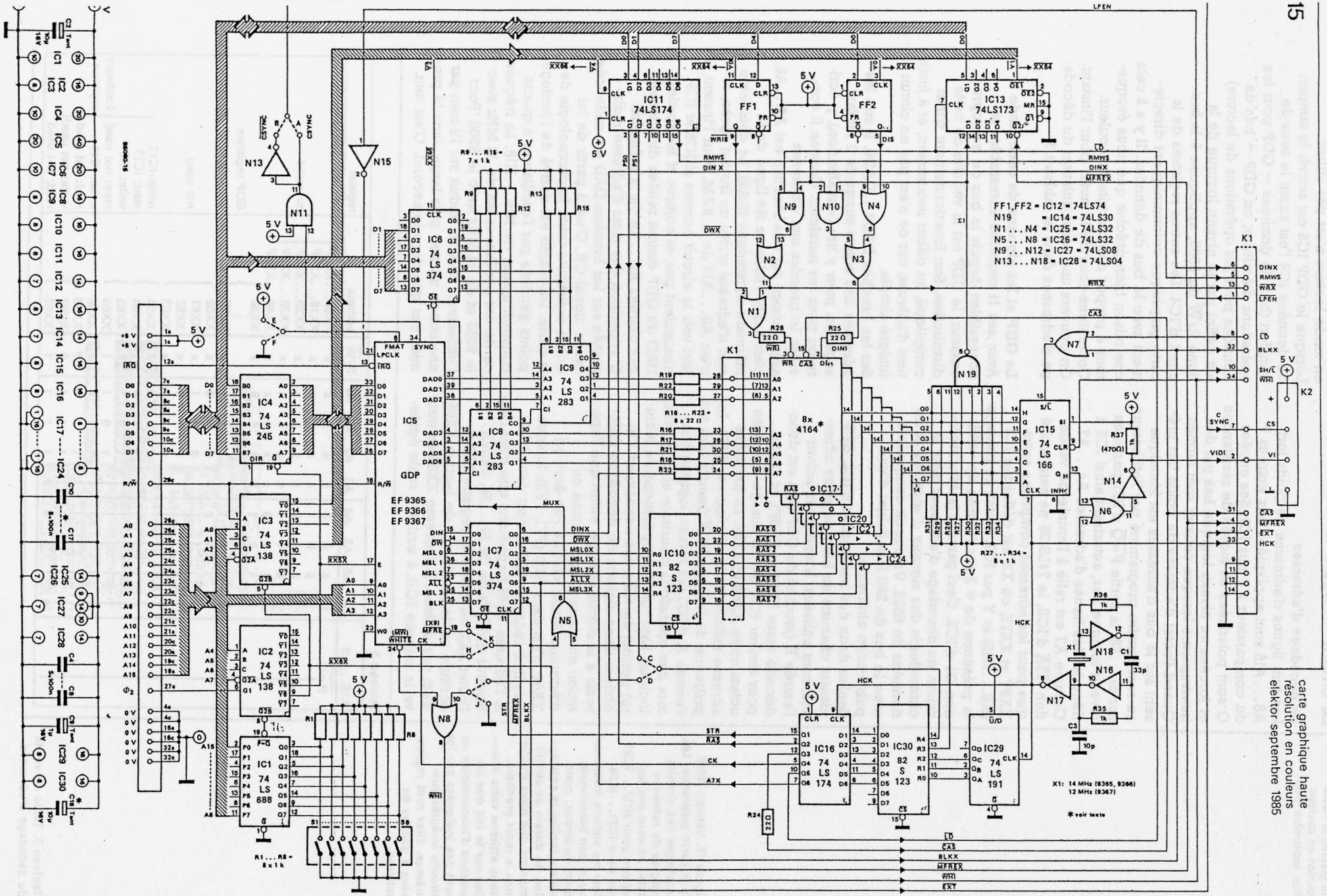
Figure 15. Version N&B de la carte graphique. Les brochages des circuits intégrés de mémoire (4164) ne sont pas les mêmes pour IC17...IC20 que pour IC21...IC24. Ceci n'a aucune influence sur le bon fonctionnement du circuit, mais nous a facilité la conception du dessin de circuit imprimé que nous publierons le mois prochain. Nous attirons votre attention sur le fait que les broches d'alimentation ne sont pas indiquées sur les circuits intégrés eux-mêmes. Elles sont regroupées à gauche du schéma.

Tableau 3

A15...A8	A7	A6	A5	A4	A3	A2	A1	A0	R/$\overline{W}$	address	register
$\overline{P=Q}$	0	0	0	0	x	x	x	x	x	XX00...XX0F	
		0	0	1	x	x	x	x	x	XX10...XX1F	
		0	1	0	x	x	x	x	x	XX20...XX2F	not used
		0	1	1	x	x	x	x	x	XX30...XX3F	
		1	0	0	x	x	x	x	x	XX40...XX4F	
		1	0	1	0	0	0	0	x	XX50	
										⋮	GDP registers
		1	0	1	1	1	1	1	x	XX5F	
		1	1	0	0	0	0	0	x	XX60	
						0	0	1	x	XX61	not used
						0	1	0	x	XX62	
						0	1	1	x	XX63	
						1	0	0	0	XX64	write: IC12
						1	0	0	1	XX64	read: IC13
						1	0	1	0	XX65	write: IC6
					1	0	1	1		XX65	read: not used
					1	1	0	0	0	XX66	write: IC11
					1	1	0	1		XX66	read: not used
					1	1	1	0	0	XX67	write: not used
	0	1	1	0	0	1	1	1	1	XX67	read: not used

hardware registers

Tableau 3. Table de vérité du décodage d'adresses.



carte graphique haute résolution en couleurs
elektor septembre 1985

Tableau 4

PROM 82S123 (IC30)

PS1	PS0	QC	QB	QA	ADDRESS HEX		N.C.	N.C.	A7	LD	CK	CAS	RAS	STR	DATA HEX	
R4	R3	R2	R1	R0			D7	D6	D5	D4	D3	D2	D1	D0		
0	0	0	0	0	00		0	0	0	1	0	1	1	0	16	page 0
0	0	0	0	1	01		0	0	0	0	0	0	0	0	00	
0	0	0	1	0	02		0	0	0	1	0	0	0	0	10	
0	0	0	1	1	03		0	0	0	1	0	0	0	0	10	
0	0	1	0	0	04		0	0	0	1	1	1	0	0	1C	
0	0	1	0	1	05		0	0	0	1	1	1	0	0	1C	
0	0	1	1	0	06		0	0	0	1	1	1	0	0	1C	
0	0	1	1	1	07		0	0	0	1	0	1	1	1	17	
0	1	0	0	0	08		0	0	0	1	0	1	1	0	16	page 1
0	1	0	0	1	09		0	0	0	0	0	0	0	0	00	
0	1	0	1	0	0A		0	0	0	1	0	0	0	0	10	
0	1	0	1	1	0B		0	0	0	1	0	0	0	0	10	
0	1	1	0	0	0C		0	0	0	1	1	1	0	0	1C	
0	1	1	0	1	0D		0	0	1	1	1	1	0	0	3C	
0	1	1	1	0	0E		0	0	1	1	1	1	0	0	3C	
0	1	1	1	1	0F		0	0	1	1	0	1	1	1	37	
1	0	0	0	0	10		0	0	1	1	0	1	1	0	36	page 2
1	0	0	0	1	11		0	0	1	0	0	0	0	0	20	
1	0	0	1	0	12		0	0	1	1	0	0	0	0	30	
1	0	0	1	1	13		0	0	1	1	0	0	0	0	30	
1	0	1	0	0	14		0	0	1	1	1	1	0	0	3C	
1	0	1	0	1	15		0	0	0	1	1	1	0	0	1C	
1	0	1	1	0	16		0	0	0	1	1	1	0	0	1C	
1	0	1	1	1	17		0	0	0	1	0	1	1	1	17	
1	1	0	0	0	18		0	0	1	1	0	1	1	0	36	page 3
1	1	0	0	1	19		0	0	1	0	0	0	0	0	20	
1	1	0	1	0	1A		0	0	1	1	0	0	0	0	30	
1	1	0	1	1	1B		0	0	1	1	0	0	0	0	30	
1	1	1	0	0	1C		0	0	1	1	1	1	0	0	3C	
1	1	1	0	1	1D		0	0	1	1	1	1	0	0	3C	
1	1	1	1	0	1E		0	0	1	1	1	1	0	0	3C	
1	1	1	1	1	1F		0	0	1	1	0	1	1	1	37	

Tableau 4. Les signaux de la figure 16 sont générés par la lecture cyclique ininterrompue de l'un des blocs de huit adresses de la PROM IC30. Quel que soit le bloc concerné (selon la sélection de la page affichée sur l'écran), la lecture se fait de l'adresse supérieure vers l'adresse inférieure (par exemple 07 06 05 04 03 02 01 00 07 06...01 00 07 etc).

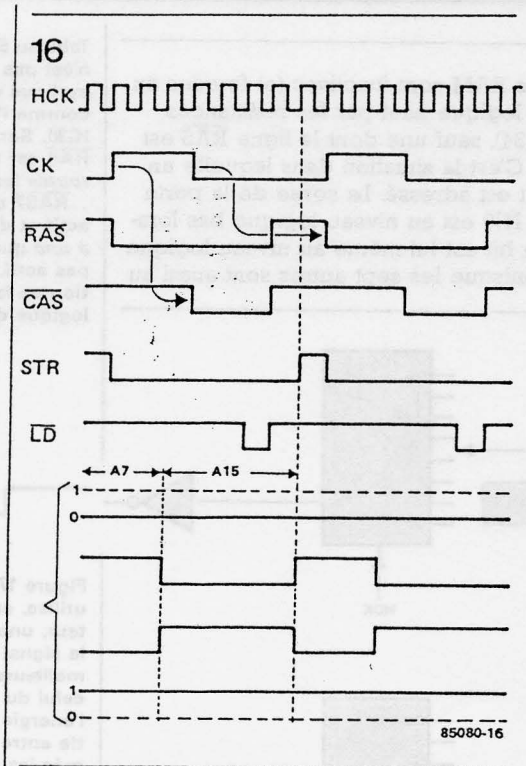


Figure 16. La chronologie des signaux de gestion de la mémoire vive dynamique est, on s'en doute, d'une importance capitale. Les flancs actifs des signaux RAS et CAS sont déterminés par les flancs respectivement descendant et montant du signal CK. La partie inférieure du chronogramme donne les quatre combinaisons de niveaux logiques possibles sur A7 à l'intérieur d'un cycle. Que l'on ne s'y trompe pas: ces combinaisons s'excluent l'une l'autre et ne peuvent donc jamais apparaître en même temps.

A7/A15

MSL3X/PS1	PS0	A15	A7
0	0	0	0
0	1	0	1
1	0	1	0
1	1	1	1

Le signal d'horloge HCK (*high clock*) cadence directement le registre à décalage de sortie IC15. Par ailleurs, il est appliqué à IC29, un décompteur qui répète inlassablement sur ses broches QA...QC toutes les configurations binaires de 7 à 0. C'est ainsi que l'on obtient un cycle de huit impulsions d'horloge (= huit points = un octet) à l'intérieur duquel les signaux de commande (CK, RAS, CAS, STR, LD et A7) sont établis à l'aide d'une PROM de 32 x 8 bits (IC30; 82S123). Les trois lignes d'adresse de poids faible de cette PROM sont commandées par IC29, de sorte que les adresses défilent sans interruption de sept à zéro. Les lignes d'adresse de poids fort d'IC30 sont commandées soit par les lignes PS0 et PS1, soit par les lignes PS0 et MSL3X. En mode non entrelacé, PS0 et PS1 permettent d'accéder à 4 x 8 adresses différentes de la PROM. Tous les signaux sont les mêmes dans chacun de ces quatre blocs, à l'exception de A7. En mode entrelacé, MSL3X remplace PS1; dans ce cas, PS0 ne permet plus que la commutation entre deux pages de mémoire, mais du coup la résolution verticale de l'écran est passée de 256 à 512 lignes.

Les signaux de sortie de la PROM IC30 sont synchronisés avec l'horloge HCK grâce aux six bascules contenues dans IC16. On trouvera dans le **tableau 4** un listing complet du contenu de la PROM IC30. Sur la **figure 16** nous indiquons la chronologie correspondante pour les signaux CK, RAS, CAS, STR et LD quelle que soit la page, et la chronologie de A7 selon l'état des lignes PS0 et PS1/MSL3X. Quand CK est au niveau logique haut, les adresses de lignes sont échantillonnées par RAS; quand CK est bas, ce sont les adresses de colonnes qui le sont par CAS. Mais si CAS est appliqué directement aux mémoires, il n'en va pas de même pour RAS. On se doute qu'il s'agit de différencier les accès collectifs des accès individuels. C'est en effet pour cette raison que l'impulsion RAS est combinée avec le signal ALL et les signaux MSL0...MSL2. Du signal ALL, nous avons vu qu'il était au niveau logique bas pour indiquer les accès collectifs à la mémoire. Les niveaux logiques des lignes MSL0...2, au contraire, indiquent lequel des huit bits est adressé. Ce décodage des signaux RAS, ALL et MSL est effectué là encore par une PROM du type 82S123 (IC10). On en trouve le contenu dans le **tableau 5**, où l'on voit que les sorties RAS0...RAS7 ne sont activées toutes ensembles que lorsque l'entrée RAS et l'entrée ALL sont toutes deux au niveau logique actif bas. Lorsque seul RAS est bas, le numéro de la sortie RAS0...RAS7 activée est déterminé par les lignes MSL0...MSL2.

Lorsque l'on sait par ailleurs que l'impulsion RAS à l'entrée d'un circuit intégré de mémoire dynamique ne sert pas seulement à échantillonner les adresses A0...A7, mais aussi comme signal de validation pour l'opération d'échantillonnage des adresses A8...A15 effectuée avec l'impulsion CAS, on comprend que la

élection d'un signal $\overline{RAS}$ parmi huit équivaut à la sélection d'un bit parmi huit.
'impulsion $\overline{CAS}$ appliquée à un circuit ui n'a pas reçu au préalable le signal $\overline{CAS}$, reste donc sans effet sur lui.
e signal STR active, avec son flanc ascenlant, le tampon IC7 qui verrouille les iveaux logiques des lignes MSL, $\overline{ALL}$, LK, DIN et $\overline{DW}$ au début de chaque nouveau cycle. Il se trouve en effet que ces iveaux logiques ne sont valables en sorie du GDP que pendant un laps de temps eaucoup plus court que la période du ignal CK; d'où la présence de ce registre ntermédiaire.
Nous verrons le rôle de N5, IC6, IC8 et C9 lorsqu'il sera question de l'échappement vertical (*scroll*).

La mémoire et sa périphérie

La mémoire est constituée, sur la carte nère, de huit circuits intégrés du type 4164 (64 K × 1 bit). Nous avons vu pourquoi, si leur ligne $\overline{CAS}$ est commune, ce n'est pas le cas pour leur entrée $\overline{RAS}$. Nous avons également vu pourquoi, si les lignes d'adresses A0...A6 sont commandées par le GDP, ce n'est pas le cas pour A7 qui est commandée par IC30 via IC16. Chaque adresse apparaissant en bonne et due forme sur les lignes d'adresses multiplexées A0...A6 + A7 concerne en principe un octet entier. Si des huit lignes $\overline{RAS}$ une seule est active, l'adresse ne concerne plus qu'un seul bit. Sur les sorties Q0...Q7 de la mémoire, il y a donc deux cas de figure possibles. Elles sont actives toutes les huit (accès collectif) lors du chargement du registre à décalage IC15. L'application du signal $\overline{LD}$ à l'entrée SH/$\overline{L}$ (*shift/load*) d'IC15 via N7 a pour effet que le registre à décalage est chargé à la fin de chaque cycle de la PROM IC30 (voir **figure 16**), mais la présence de BLKX (*blanking*) sur cette même entrée empêche le chargement du registre à décalage en dehors de la fenêtre de visualisation. Il est d'ailleurs intéressant, à titre expérimental, de supprimer ce signal BLKX à cet endroit: on "voit" alors le GDP manipuler la mémoire sur les bords de l'écran... Malheureusement, les limites de cet article ne laissent pas de place pour ce genre de digressions.
La sortie du registre à décalage IC15 (Q_H) délivre le signal vidéo obtenu à travers la sérialisation du contenu de la mémoire. Si deux bits voisins dans un octet sont au niveau logique haut, le signal vidéo reste au niveau logique haut lui aussi d'un point à l'autre. Ceci a pour effet de réduire le piqué de l'image sur certains moniteurs ou les postes TV modifiés où elle devient "baveuse". C'est pourquoi on hache le signal décalé à l'aide de l'horloge-points dans N6, afin de compenser l'excédent d'énergie (voir **figure 17**). Ceci présente l'inconvénient d'exiger une bande passante beaucoup plus large du moniteur utilisé. Là encore, il est intéressant, à titre expérimental, de supprimer le signal HCK sur N6, pour "voir la différence"...
Dans l'autre cas de figure, toutes les sorties des RAM sont inactives (et forcées au niveau logique haut par les résistances R27...34), sauf une dont la ligne $\overline{RAS}$ est active. C'est la situation dans laquelle un seul bit est adressé. La sortie de la porte NAND N19 est au niveau logique bas lorsque ce bit est lui-même au niveau logique haut (puisque les sept autres sont aussi au

Tableau 5

carte graphique haute résolution en couleurs elektor septembre 1985

PROM 82S123 (IC10)																
$\overline{RAS}$	$\overline{ALL}$	MSL2	MSL1	MSL0	ADDRESS HEX		$\overline{RAS7}$	$\overline{RAS6}$	$\overline{RAS5}$	$\overline{RAS4}$	$\overline{RAS3}$	$\overline{RAS2}$	$\overline{RAS1}$	$\overline{RAS0}$	DATA HEX	
R4	R3	R2	R1	R0			D7	D6	D5	D4	D3	D2	D1	D0		
0	0	0	0	0	00		0	0	0	0	0	0	0	0	00	accès collectif
0	0	0	0	1	01		0	0	0	0	0	0	0	0	00	
0	0	0	1	0	02		0	0	0	0	0	0	0	0	00	
0	0	0	1	1	03		0	0	0	0	0	0	0	0	00	
0	0	1	0	0	04		0	0	0	0	0	0	0	0	00	
0	0	1	0	1	05		0	0	0	0	0	0	0	0	00	
0	0	1	1	0	06		0	0	0	0	0	0	0	0	00	
0	0	1	1	1	07		0	0	0	0	0	0	0	0	00	
0	1	0	0	0	08		1	1	1	1	1	1	1	0	FE	accès bit par bit
0	1	0	0	1	09		1	1	1	1	1	1	0	1	FD	
0	1	0	1	0	0A		1	1	1	1	1	0	1	1	FB	
0	1	0	1	1	0B		1	1	1	1	0	1	1	1	F7	
0	1	1	0	0	0C		1	1	1	0	1	1	1	1	EF	
0	1	1	0	1	0D		1	1	0	1	1	1	1	1	DF	
0	1	1	1	0	0E		1	0	1	1	1	1	1	1	BF	
0	1	1	1	1	0F		0	1	1	1	1	1	1	1	7F	
1	0	0	0	0	10		1	1	1	1	1	1	1	1	FF	pas d'accès
1	0	0	0	1	11		1	1	1	1	1	1	1	1	FF	
1	0	0	1	0	12		1	1	1	1	1	1	1	1	FF	
1	0	0	1	1	13		1	1	1	1	1	1	1	1	FF	
1	0	1	0	0	14		1	1	1	1	1	1	1	1	FF	
1	0	1	0	1	15		1	1	1	1	1	1	1	1	FF	
1	0	1	1	0	16		1	1	1	1	1	1	1	1	FF	
1	0	1	1	1	17		1	1	1	1	1	1	1	1	FF	
1	1	0	0	0	18		1	1	1	1	1	1	1	1	FF	
1	1	0	0	1	19		1	1	1	1	1	1	1	1	FF	
1	1	0	1	0	1A		1	1	1	1	1	1	1	1	FF	
1	1	0	1	1	1B		1	1	1	1	1	1	1	1	FF	
1	1	1	0	0	1C		1	1	1	1	1	1	1	1	FF	
1	1	1	0	1	1D		1	1	1	1	1	1	1	1	FF	
1	1	1	1	0	1E		1	1	1	1	1	1	1	1	FF	
1	1	1	1	1	1F		1	1	1	1	1	1	1	1	FF	

Tableau 5. La PROM IC10 n'est pas lue de façon cyclique et interrompue comme l'est la PROM IC30. Son rôle, quand $\overline{RAS}$ est bas, est d'activer *toutes* les lignes $\overline{RAS0}$... $\overline{RAS7}$ quand $\overline{ALL}$ est actif et de les activer *une à une* quand $\overline{ALL}$ n'est pas actif, et ceci en fonction de la combinaison logique des lignes MSL.

17

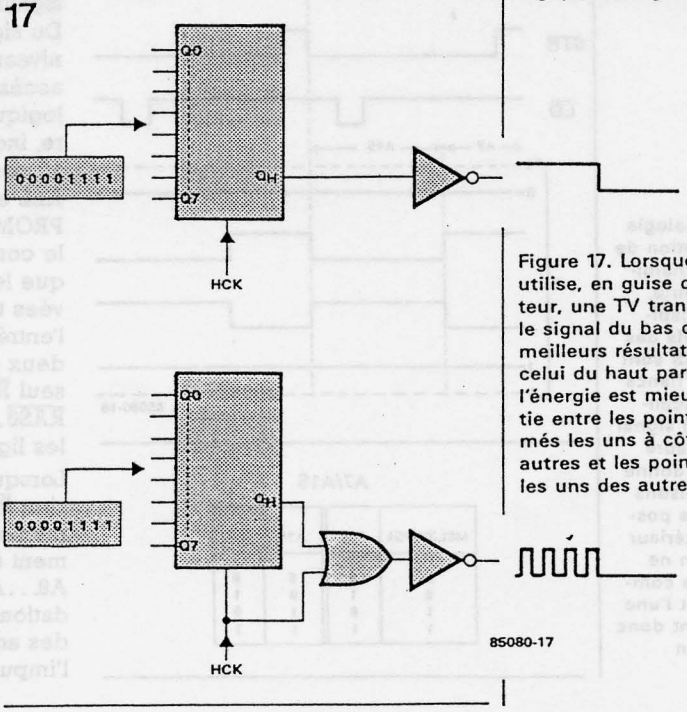


Figure 17. Lorsque l'on utilise, en guise de moniteur, une TV transformée, le signal du bas donne de meilleurs résultats que celui du haut parce que l'énergie est mieux répartie entre les points allumés les uns à côté des autres et les points isolés les uns des autres.

niveau logique haut), et elle est au niveau logique haut lorsque ce bit est lui-même au niveau logique bas. **Nous disposons donc sur la ligne Σ (sigma) du niveau logique inversé du bit adressé.** Chargé dans le registre IC13, ce bit peut donc être lu par le microprocesseur hôte via le bus de données (DØ), à l'adresse XX64. On remarque que le chargement dans le registre 74LS173 est provoqué par l'apparition simultanée du signal $\overline{LD}$ qui indique la validité de l'accès et du signal $\overline{MFREX}$ qui indique que le bit en présence est celui dont le microprocesseur a donné les coordonnées au GDP.
Par ailleurs, le bit Σ est appliqué au circuit de sélection de couleur qui comporte la logique RMW dont nous avons déjà parlé.

Ecriture dans la mémoire d'image
La condition sine qua non pour écrire dans la mémoire est l'activation (au niveau logique bas) de la ligne $\overline{DW}$ par le GDP. L'application de ce signal à la broche $\overline{WRITE}$ des RAM est conditionnée par la ligne $\overline{WRIS}$ (*write I select*) qui n'est active que lorsqu'il y a lieu, en fonction de la gestion des couleurs, d'écrire dans le plan de mémoire I. Dans la version N&B, la ligne $\overline{WRIS}$ est donc en principe toujours active.
Cependant, l'application du signal $\overline{DW}$ à la RAM est également conditionnée par le niveau logique de sortie de la porte AND N9, qui appartient à la logique RMW. Admettons pour l'instant que cette sortie est au niveau logique bas, permettant ainsi à la broche $\overline{WR}$ d'être activée par le signal $\overline{DW}$.
La donnée à écrire dans la mémoire apparaît sur la ligne DINX. Elle ne peut être acheminée vers l'entrée Din des RAM via N3 et N4 qu'en combinaison avec le niveau logique de la ligne de sélection de couleur DIS (*data I select*), et le niveau logique de sortie de la porte N10. Supposons, pour l'instant, que la sortie de la porte AND N10 (logique RMW) est au niveau logique bas. Si DINX est au niveau logique haut, c'est que le GDP veut éteindre le point adressé. A cela, l'état de la ligne de sélection DIS ne peut rien changer; le seul moyen d'empêcher, de l'extérieur, l'extinction de ce point est d'interdire l'écriture dans le plan I en mettant au niveau logique haut la ligne $\overline{WRIS}$ dans le registre IC12.
Si DINX est au niveau logique bas, c'est que le GDP veut allumer le point adressé. Si au contraire, la gestion des couleurs exige que ce point soit éteint (plume d'une couleur sur fond d'une autre couleur), il faut que la ligne DIS passe au niveau logique haut (tandis que $\overline{WRIS}$ est actif). Comme on le remarque, tout ceci est valable pour la couleur et n'a donc pas d'utilité réelle en N&B. Il a cependant fallu prévoir cette logique combinatoire dès le début, afin de faciliter l'adjonction ultérieure de la couleur (voir **tableau 6**)
Venons-en à présent à la logique de lecture/modification/écriture telle que nous l'avions déjà abordée avec les figures 12 et 13. Les portes nécessaires à la sélection des couleurs sont N1 et N4 associées au registre IC12. Pour la logique RMW, il a fallu rajouter N2, N3, N9 et N10, associées à la ligne RMWS (*read/modify/write select*) issue du registre IC11. Quand cette ligne est au niveau logique bas, c'est comme si les portes N2, N3, N9 et N10 n'existaient pas. Quand elle est au niveau logique haut par contre, nous sommes en mode RMW. Là encore, lorsque le GDP veut éteindre un point (DINX au niveau logique haut), rien ne peut l'en empêcher, si ce n'est la logique de sélection de couleur qui peut interdire l'écriture dans un ou plusieurs plans de mémoire. Cependant, la combinaison du signal $\overline{LD}$ interdit l'écriture en mode RMW, sauf au moment précis où le bit à modifier est disponible en sortie de N19 (voir figure 16).
Les combinaisons possibles de la logique RMW en N&B ont été données dans le tableau 2. Nous les reprendrons en détail lorsque nous aurons affaire à la couleur.

Tableau 6. Quelques exemples de combinaisons logiques des lignes de sélection de la couleur (sans mode RMW).

Tableau 6

write enable signals on color planes					data signals on color planes				GDP signals		
RMWS	R	G	B	I	R	G	B	I	DWX	DINX	PIXEL
Ø	X	X	X	X	X	X	X	X	1	X	no write operation
Ø	1	1	1	1	X	X	X	X	Ø	X	writing not allowed on this pixel
Ø	Ø	1	1	1	X	X	X	X	Ø	1	turn off red dot; no change on green, blue and I planes
Ø	Ø	Ø	1	1	X	X	X	X	Ø	1	turn off red and green dots; no change on blue and I planes
Ø	Ø	Ø	Ø	Ø	X	X	X	X	Ø	1	turn off all dots
Ø	Ø	Ø	Ø	Ø	Ø	1	Ø	1	Ø	Ø	turn on red and blue dots; turn off green and I dots
Ø	Ø	1	Ø	Ø	1	X	1	Ø	Ø	Ø	turn off red and blue dots; turn off I dot; no change on green plane
Ø	1	Ø	Ø	Ø	X	1	Ø	1	Ø	Ø	turn on blue dots; turn off green and I dots; no change on red plane
Ø	Ø	Ø	Ø	Ø	1	1	1	1	Ø	Ø	turn off all dots

Scroll
La dernière partie du circuit à examiner est la logique d'échappement vertical. Sont concernés la porte OR N5, les additionneurs rapides à 4 bits IC8 et IC9, et enfin le registre IC6.
Nous avons vu comment le GDP établissait la correspondance entre les adresses de visualisation, c'est-à-dire les "adresses sur l'écran", et les adresses dans la mémoire. Si nous voulons obtenir un échappement vertical du contenu de l'écran, intéressant surtout dans les applications alphanumériques, il nous faut modifier d'une manière ou d'une autre **les adresses verticales** fournies par le GDP qui ne connaît pas cette fonction. C'est donc une opération que nous faisons à son insu, comme c'est le cas pour la couleur ou la lecture/modification/écriture.
On peut opter pour deux types d'échappement: l'un déplace le contenu de

l'écran par rapport au contenu de la mémoire et demande une manipulation assez complexe. Le second, plus simple, se contente de l'effet visuel du rouleau sans fin qui défile devant le spectateur. C'est ce type d'échappement que nous avons retenu ici.

Nous savons que les adresses verticales (axe Y) ne sont présentes que pendant une partie du cycle d'adressage, puisqu'elles sont multiplexées avec les adresses horizontales (axe X). Or ces dernières ne doivent pas être modifiées, à défaut de quoi nous obtiendrions un échappement horizontal (qui est intéressant aussi, soit dit en passant), voire oblique.

Plutôt que de procéder à un démultiplexage réel des adresses, on fait appel ici à une caractéristique élémentaire de la logique binaire. Comme on va le voir, c'est très simple, mais il fallait y penser!

Les additionneurs IC8 et IC9 reçoivent sur leurs entrées A les bits d'adresse du GDP. Sur leurs entrées B ils reçoivent un mot binaire de 7 bits qu'ils additionnent au mot binaire des entrées A. Sur les sorties Q apparaît le résultat qui est l'adresse verticale modifiée pour obtenir le défilement du contenu de l'écran, ou l'adresse horizontale non modifiée. Le mot de 7 bits fourni par le registre IC6 est calculé à l'aide du logiciel en fonction de la position du curseur par rapport à la limite inférieure de l'écran.

Le démultiplexage entre les adresses verticales et les adresses horizontales qui se succèdent sur les broches DAD0. . . DAD6 est effectué à l'aide de N5. Cette porte combine deux signaux: l'un est BLKX, dont nous savons qu'il est actif au niveau logique haut en dehors de la fenêtre de visualisation, et l'autre est $\overline{RAS}$, dont nous savons qu'il sert à échantillonner les adresses horizontales. Or, le signal MUX issu de cette combinaison commande l'entrée de validation du registre IC6 (*output enable*) dont les sorties présentent une haute impédance lorsque le circuit est inactivé, mais aussi l'entrée $Carry_{in}$, c'est-à-dire l'entrée de retenue du premier additionneur. Ce qui signifie que lorsque la ligne MUX est au niveau logique haut, la valeur du mot binaire sur les entrées B des additionneurs est "111 1111" (grâce aux sept résistances R9...R15) tandis que la retenue est 1. **Quelle que soit la combinaison présente dans ces conditions sur les entrées A des additionneurs, on la retrouvera inchangée sur les sorties Q.** Il suffit donc de mettre la ligne MUX au niveau logique haut toutes les fois que l'addition ne doit pas avoir lieu. C'est d'une part lorsque l'on est en dehors de la fenêtre de visualisation et d'autre part lorsque dans la fenêtre apparaissent les adresses de visualisation horizontales, qui doivent rester inchangées. C'est BLKX qui s'occupe de verrouiller les additionneurs en dehors de la fenêtre, et $\overline{RAS}$ qui ne les déverrouille, dans la fenêtre, que pour les adresses verticales.

Le lecteur attentif aura noté la subtilité de cette manoeuvre. Mais il aura certainement noté également une contradiction flagrante entre la volonté d'inhiber l'addition sur les adresses horizontales, et le fait d'utiliser le signal $\overline{RAS}$ justement pour autoriser cette addition, alors que l'on sait par ailleurs que le signal $\overline{RAS}$ sert à échantillonner précisément les adresses horizontales. La contradiction n'est qu'apparente. En effet, le signal $\overline{RAS}$ est appelé, à juste titre, signal **d'échantillonnage** des adresses horizontales. C'est-à-dire que celles-ci sont prises en compte par les circuits de RAM dans l'état où elles sont au moment du flanc descendant sur la ligne $\overline{RAS}$. C'est là une caractéristique des RAM **dynamiques**, leur nom le dit bien. Donc les niveaux logiques de sortie des additionneurs sont encore ceux des entrées A au moment où $\overline{RAS}$ met l'entrée de retenue au niveau logique bas. Les adresses horizontales sont restées inchangées, mais par contre les adresses verticales qui apparaissent ensuite sur les entrées A des additionneurs seront affectées par le mot binaire présent sur les entrées B. Lorsque les additionneurs seront remis en mode "by pass" par la ligne $\overline{RAS}$ revenant au niveau logique haut à la fin du cycle d'adressage, les adresses verticales modifiées auront été échantillonnées à leur tour par le signal $\overline{CAS}$. Une remarque finale concernant le registre IC6: lorsque son contenu est FF_{HEX} (= "1111 1111"), les adresses verticales aussi bien que les adresses horizontales restent inchangées.

A suivre

Avec ce passage en revue du circuit de la carte mère, nous en arrivons à la fin du premier article consacré à la carte graphique. Le mois prochain, il sera question de sa réalisation et de sa mise en oeuvre. Puis, nous aborderons la carte d'extension et le logiciel, que nous avons déjà évoqués l'une et l'autre au fil de ce premier article. D'ores et déjà, nous attendons vos réactions, vos suggestions et, pourquoi pas, vos contributions. De notre côté, nous vous proposerons, le moment venu, des exemples d'application de ce système graphique, des programmes, des trucs et des ficelles en tous genres, et aussi des circuits périphériques comme par exemple l'incrustation d'images graphiques dans un signal vidéo, etc. Sans parler d'une combinaison de la carte graphique et de la carte CPU universelle qui pourrait nous donner un véritable terminal graphique universel, que l'on attaquera tout simplement via une entrée Centronics ou sérielle, à partir de n'importe quel ordinateur! Plus de problèmes d'interfaçage de bus, plus de problèmes d'adaption du logiciel... et cela sans compromettre le moins du monde la vitesse d'exécution. Au contraire.

En attendant, soyez remerciés, chers lecteurs, pour votre confiance et votre patience que nous nous efforcerons, plus que jamais, de ne pas décevoir.