SPPU004

Video Display Processors

Guide du Programmeur

Video Display Products

TEXAS INSTRUMENTS

Video Display Processors

Guide du Programmeur

TEXAS INSTRUMENTS

1 INTRODUCTION

Cet ouvrage est le premier d'une série traitant de la programmation des processeurs d'affichage vidéo (VDP) de Texas Instruments. Ce guide du programmeur prêtera une attention particulière aux fondamentaux d'initialisation et de création d'un affichage avec les VDP TMS 9918/28/29. Ce livre sert aussi pour leurs prédécesseurs, les TMS 9918A/28A/29A, et sert de pré-requis aux futures publications de la génération suivante de VDP avancés de Texas Instruments. Les différences matérielles seront notées dans un souci de commodité.

La technique de programmation choisie dans cet ouvrage est l'assembleur. La plupart des exemples de programmes sont très généraux, dans un souci de clarté.

Tous les sujets nécessaires pour la programmation du VDP sont abordés dans ce guide de programmeur. Si un sujet n'est pas au premier abord expliqué suffisamment en profondeur ou si vous désirez plus d'informations sur un sujet particulier, laissez la table des matières vous guider vers une explication plus détaillée de cette question.

1.1 Fonctionnement général du VDP

Le VDP rapporte des données depuis la mémoire vidéo (VRAM) et les convertit en une série de données utilisées pour contrôler le faisceau du tube cathodique qui balaie l'écran. Le VDP effectue cette opération à plusieurs reprises, à l'instar d'un programme qui exécuterait une boucle. Cependant, le VDP exécute de nombreuses autres fonction lors de cette simulation de boucle.

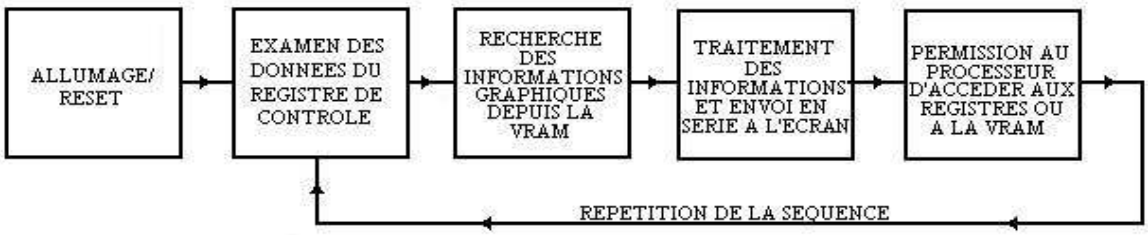


FIGURE 1.1 - DEROULEMENT DES OPERATION DU VDP

Une grande partie de la polyvalence du VDP découle du fait qu'il n'est pas restreint à récupérer les données du même endroit pendant la même séquence. Le VDP possède neuf registres internes, huit d'entre eux possèdent des bits d'option et de contrôle qui peuvent être programmés par l'utilisateur. Le neuvième registre est le registre de statut, qui peut être lu par l'utilisateur afin de déterminer certaines choses qui se produisent dans le VDP. Grâce aux informations de programmation contenues dans les 8 registres de contrôle, le VDP peut être amené à rapporter des données depuis plusieurs endroits différents de la VRAM, et ce au cours de différentes séquences.

Le VDP prend du temps pendant quelques microsecondes pour voir si le microprocesseur a besoin d'accéder à l'un des registres internes ou à la VRAM. Si le VDP ne dispensait pas cette fonction, il ne serait pas possible de programmer les registre, lire le statut, ou même charger un chef-d'œuvre artistique dans la VRAM pour l'affichage.

1.2 Références

1) TMS9918A/28A/29A Video Display Processors Data Manual (MP010A)
2) TMS9118/28/29 Video Display Processors Data Manual (SPPS002)
3) TMS9928/29 and TMS9128/29 Interface to Color Monitors Application (SPPA004)
4) TMS9118/TMS9128/TMS9129 Evaluation Module User’s Guide (SPPU003)
5) 5)Dual Video Display Processor Application Report (SPPA005)

2 FONCTIONS

2.1 Plans d'affichage

Le VDP affiche une image à l'écran qui peut être au mieux symbolisée par un ensemble de 35 plans d'affichages superposés les uns aux autres (voir figure 2-1). En regardant le moniteur ou l'écran de télévision, on peut visualiser le plan de plus haute priorité comme étant le plus proche de nous, et inversement le plan de plus basse priorité comme étant le plus éloigné.

S'il arrive que des motifs de différents plans occupent la même place sur l'écran, alors ce sera le motif appartenant au plan de plus haute priorité qui sera affiché à cet endroit. Si on veut voir un motif sur un plan de plus basse priorité, il faut que le ou les motifs le couvrant soient mis à la couleur transparente du VDP. Voir TMS9118/28/29 Video Display Processors Data Manual (SPPS002) pour plus de détails.

Les 35 plans priorisés sont montrés sur la figure 2-1, avec chacun des 32 premiers plans contenant un seul sprite. Un sprite est un objet définissable dont la position à l'écran est relative à ses coordonnées X et Y. Ces coordonnées sont composées de deux octets dans la VRAM. En changeant les valeurs de ces deux octets, on peut facilement déplacer un sprite dans l'écran. Les sprites sont disponibles en 2 tailles : 8x8 pixels ou 16x16 pixels. Ces sprites peuvent aussi être agrandis en 16x16 ou 32x32 pixels.

Derrière ces plans de sprites se trouve le plan des motifs (ou patrons). Ce plan est utilisé pour afficher soit des graphiques, soit du texte. Le VDP peut afficher des patrons sur ce plan dans chacun des différents modes d'affichage disponibles : texte, graphique 1, graphique 2 ou multicolore.

2.2 Modes d'affichage

Le mode texte divise l'écran en blocs de 6x8 pixels conçus spécifiquement pour l'affichage de texte. En mode graphique 1, l'écran est divisé en 32 blocs horizontaux sur 24 blocs verticaux. Chaque bloc du mode graphique 1 est constitué de 8x8 pixels, on atteint ainsi une résolution de 256x192 pixels. En mode graphique 2, la division et la résolution de l'écran sont les mêmes qu'en mode graphique 1, mais les possibilités d'affichage de patrons et de couleurs sont plus complexes. Le mode multicolore est un mode d'affichage basse résolution qui divise l'écran en 64 blocs horizontaux par 48 blocs verticaux. Chaque bloc du mode multicolore est formé de 4x4 pixels de l'une des 16 couleurs disponibles.

Derrière le plan des patrons se trouve le plan de fond, d'une surface plus grande que les autres plans. Il forme donc une bordure autour de ceux-ci. La couleur de fond est définie par 4 bits dans le registre 7 du VDP.

Le 35ème – et donc de plus basse priorité – plan est le plan VDP externe. Si un VDP esclave externe est détecté par le VDP maître, alors les plans seront affichés au-dessus du plan externe. Si l'on veut voir une portion de ce plan externe, il faut mettre les éléments couvrants des 34 plans en transparent.

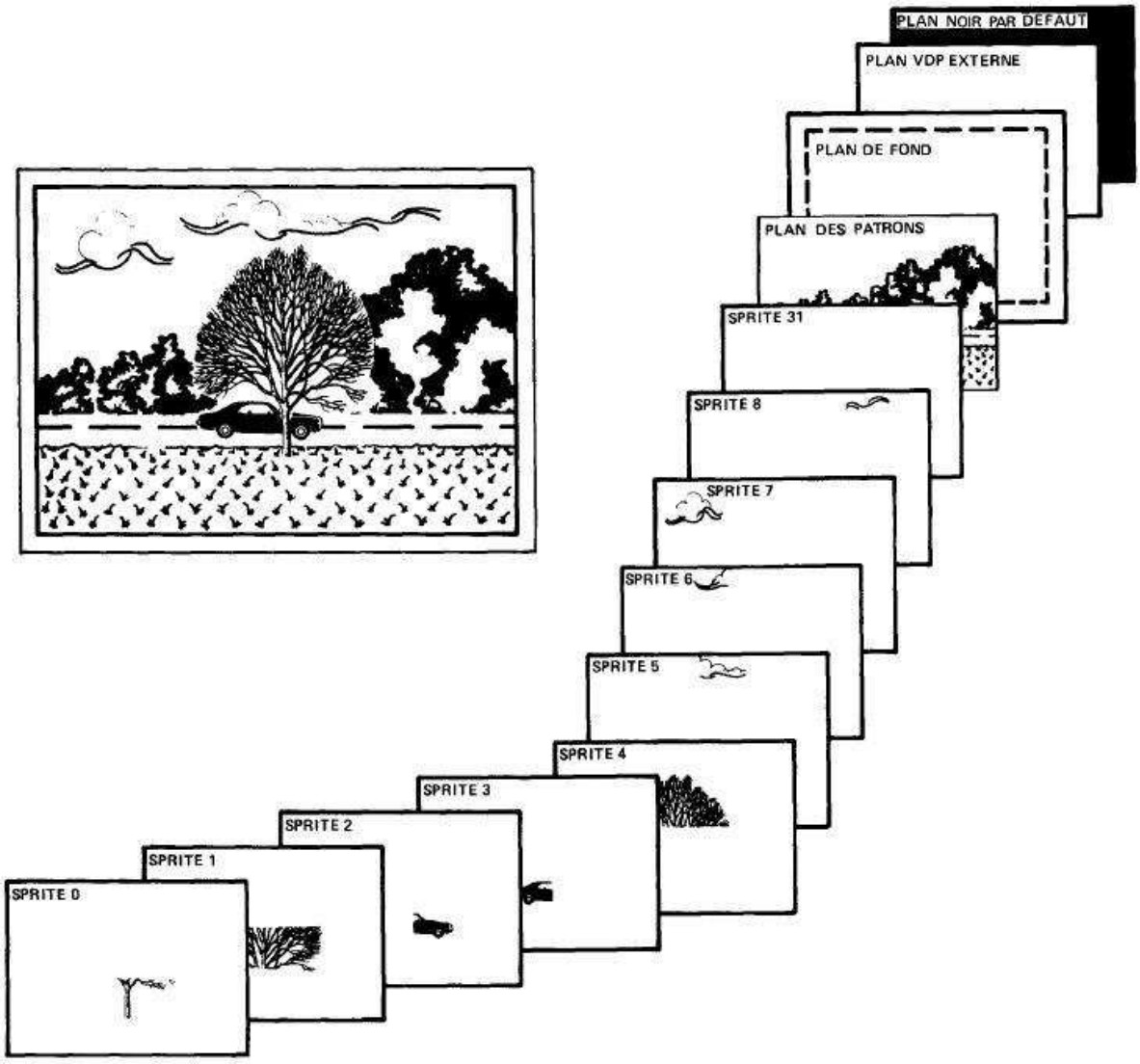


FIG 2-1 - PLANS D'AFFICHAGE DU VDP

2.3 Couleurs disponibles

Le VDP peut afficher 16 couleurs (incluant le transparent) comme indiqué dans le tableau 1-2. Le VDP peut aussi afficher 15 niveaux de gris différents sur les moniteurs monochromes.

TABLEAU 2-1 – Affectation des couleurs du VDP

Numéro de couleur (hexadécimal)	Couleur correspondante	Numéro de couleur (hexadécimal)	Couleur correspondante
0	Transparent	8	Rouge moyen
1	Noir	9	Rouge clair
2	Vert moyen	A	Jaune foncé
3	Vert clair	B	Jaune clair
4	Bleu foncé	C	Vert foncé
5	Bleu clair	D	Magenta
6	Rouge foncé	E	Gris
7	Bleu émeraude	F	Blanc

3 CIRCUIT DE COMMUNICATION

Le circuit présenté dans la figure 3-1 est en réalité une partie du module d'évaluation Texas Instruments TMS 9118/28/29 (disponible en démonstration chez votre commerçant TI). Nous utiliserons ce circuit pour aider à décrire comment le processeur et le VDP communiquent. Ce circuit est un système de travail complet.

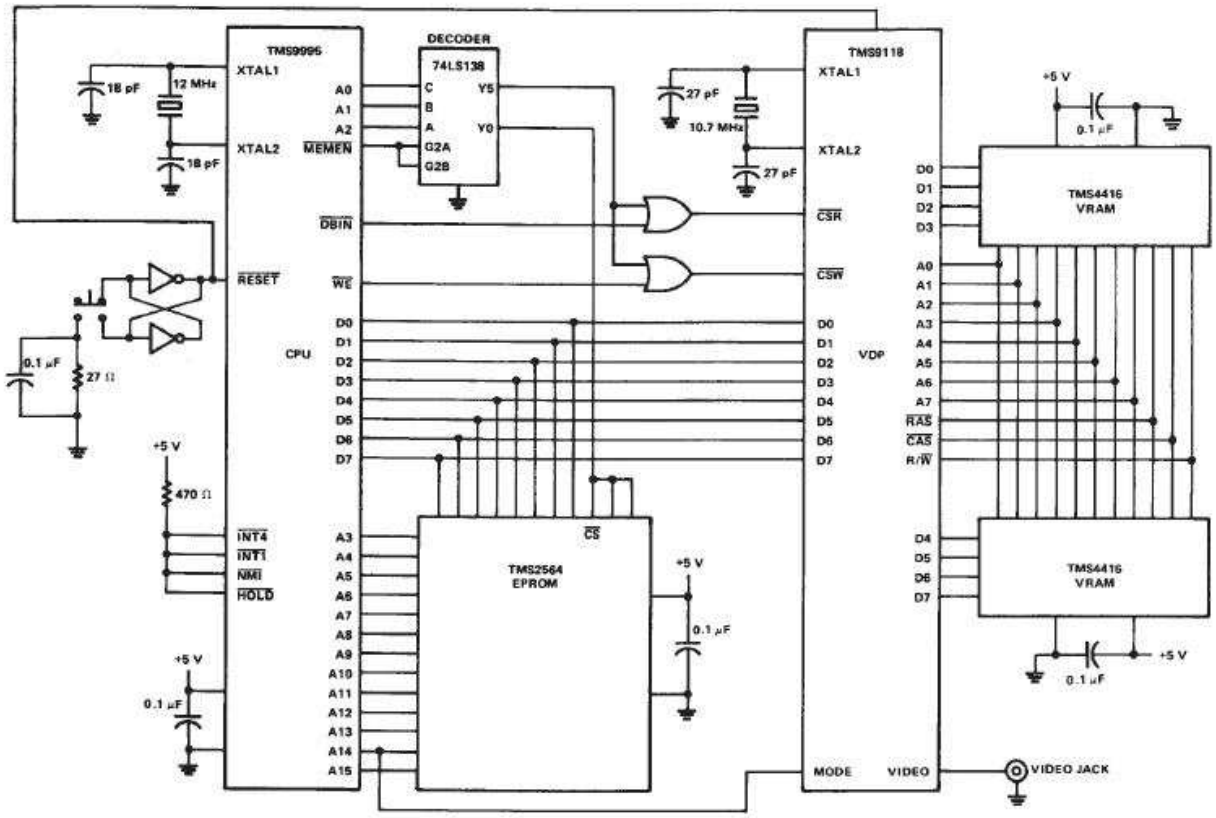


FIG 3-1 - INTERFACE PROCESSEUR VERS VDP

3.1 Interface processeur vers VDP

Le processeur communique avec le VDP à travers un bus 8 bits bidirectionnel, 3 lignes de contrôles et une ligne d'interruption. Les trois signaux de contrôles sont $\overline{\text{CSR}}$ (chip select read), $\overline{\text{CSW}}$ (chip select write) et MODE. $\overline{\text{CSR}}$ et $\overline{\text{CSW}}$ déterminent si le VDP prend des informations depuis le bus de données ou en envoie dessus. Si $\overline{\text{CSR}}$ est actif, le VDP enverra des informations au processeur par le bus de données. Si $\overline{\text{CSW}}$ est actif, le VDP prendra les informations envoyées par le processeur depuis le bus de données.

Le signal MODE détermine la source ou la destination d'un transfert de données. Si le signal MODE est bas, le VDP réalisera une opération sur la VRAM. Si le signal MODE est haut, le VDP réalisera une opération sur les registres du VDP.

Un des moyens les plus simples pour concevoir l'interface matérielle est de réserver 2 adresses dans le plan de mémoire du processeur hôte dédiés à la communication avec le VDP. Dans le circuit de la figure 3-1, les deux adresses réservées sont les adresses C000H et C002H. Une opération faite par le processeur à l'adresse C000H fera passer le signal MODE en signal bas. Une opération faite à l'adresse C002H fera passer le signal MODE en signal haut. $\overline{CSR}$ et $\overline{CSW}$ sont contrôlés par la logique d'entrée/sortie du processeur. Si une opération de lecture est effectuée, $\overline{CSR}$ sera actif (bas), et si une opération d'écriture est effectuée, $\overline{CSW}$ sera actif (bas).

NOTE

Les adresses qui seront utilisées seront probablement différentes
de C000H et C002H suivant le système utilisé mais les fonctions resteront identiques.

Afin d'utiliser la pleine capacité de chaque mode graphique, notre VDP doit avoir au moins 16 Ko de VRAM disponible. C'est aussi le montant de VRAM le plus populaire qu'on peut trouver dans les systèmes VDP. La VRAM est localisée dans le VDP entre les adresses 0000H à 3FFFH. Comme expliqué précédemment, la VRAM ne peut être adressée par le processeur pour la lecture ou l'écriture qu'à travers les adresses réservées C000H et C0002H.

Une autre remarque importante concerne les exemples utilisant les adresses et les lignes de données. Les exemples de cet ouvrage donnent le bit le plus significatif (MSB) comme étant le bit D0, et le bit le moins significatif (LSB) comme étant le bit D7. Il en va de même pour le bus d'adresses de 14 bits, le bit A0 étant le MSB, et le bit A13 le LSB.

3.2 Opérations logicielles

Le processeur peut être programmé pour réaliser l'une de ces 4 opérations :

1) Ecrire un octet de donnée dans la VRAM
2) Lire un octet de donnée depuis la VRAM
3) Ecrire dans l'un des 8 registres internes de contrôle du VDP, ou initialiser l'adresse de la VRAM en écrivant dans le registre d'adresse de 14 bits.
4) Lire le registre de statut du VDP

Chacune de ces opérations requiert qu'au moins un transfert de données ait lieu du processeur vers le VDP. Le VDP détermine lequel de ces quatre transferts de données est exécuté par l'état des trois signaux de contrôle (CSR, CSW et le MODE) comme indiqué dans le tableau 3-1.

TABLEAU 3-1 – Transfert de données du processeur vers le VDP

OPERATION	$\overline{CSW}$	$\overline{CSR}$	MODE	ADRESSE DU PORT
Ecrire dans la VRAM	0	1	0	C000H
Lire depuis la VRAM	1	0	0	C000H
Ecrire dans un registre du VDP	0	1	1	C002H
Lire le registre de statut	1	0	1	C002H

NOTE

Les adresses C000H et C002H sont des adresse arbitraires choisies pour ce guide.

4 PARLER AU VDP

4.1 Ecrire dans les registres du VDP

Le VDP possède 8 registres en écriture seule et un registre de statut en lecture seule. Les registres en écriture seule contiennent des informations qui contrôlent les opérations du VDP, incluant la manière dont la VRAM est allouée. Le registre de statut contient des informations sur les interruptions et sur les sprites.

Un registre à écriture seule est chargé en utilisant 2 transferts de données 8 bits depuis le processeur. Le premier octet écrit est la valeur de la donnée, et le second octet est le numéro de registre qui indique au VDP où envoyer la donnée à écrire. Le MSB du deuxième octet doit être un 1, les 4 bits suivant à 0, et les 3 bits les plus bas codent le registre à traiter (valeurs de 0 à 7). Le tableau 4-1 explicite le format pour les 8 registres en écriture seule.

TABLEAU 4-1 – Ecriture dans les registres du VDP

OPERATION	MSB 0	1	2	3	4	5	6	LSB 7	$\overline{CSR}$	$\overline{CSW}$	MODE
Ecriture de donnée (octet 1)	D0	D1	D2	D3	D4	D5	D6	D7	1	0	1
Sélection du registre (octet 2)	1	0	0	0	0	Rn	Rn	Rn	1	0	1

EXEMPLE 4-1 :

Disons que nous voulons initialiser le registre 0 (R0) à la valeur 0. Le premier octet envoyé à l'adresse C002H sera 00H, le second sera 80H (souvenez-vous que le MSB doit être mis à 1). Si nous avions voulu écrire 0 dans le registre R7, alors le second octet envoyé aurait été 87H.

4.2 Lire le registre de statut

Le contenu du registre de statut peut être lu par un seul transfert de données, en réalisant simplement une lecture de l'adresse C0002H (voir tableau 4-2).

TABLEAU 4-2 – Lecture du registre de statut du VDP

OPERATION	MSB 0	1	2	3	4	5	6	LSB 7	$\overline{CSR}$	$\overline{CSW}$	MODE
Lecture de donnée (octet 1)	D0	D1	D2	D3	D4	D5	D6	D7	0	1	1

4.3 Ecrire et lire dans la VRAM

Le VDP est connecté à la VRAM par un registre d'adresse de 14 bits auto-incrémental. Une fois que l'adresse à lire ou sur laquelle écrire est définie (transfert de données de 2 octets), on peut lire ou écrire un octet de donnée en utilisant un transfert d'un octet. Continuer de lire ou écrire sur le VDP implique l'incrément automatique de l'adresse. Donc, écrire ou lire tout un bloc de données peut être exécuté très rapidement. Le signal MODE est haut (MODE1) pour le premier transfert de 2 octets (définition de l'adresse), puis bas (MODE0) pour l'écriture ou la lecture de la VRAM.

Les séquences suivantes illustrent les étapes exactes à suivre pour l'écriture et la lecture dans la VRAM. Consultez les tableaux 4-3 et 4-4 pour les détails.

Ecrire dans la VRAM :

1) Transférer les 8 bits les plus bas de l'adresse (en MODE haut)
2) Transférer les 8 bits les plus hauts de l'adresse (en MODE haut). Les 2 MSB doivent être respectivement mis à 0 et 1.
3) Ecrire l'octet en MODE bas
4) Ecrire l'octet suivant

TABLEAU 4-3 – Ecriture dans VRAM

OPERATION	MSB 0	1	2	3	4	5	6	LSB 7	$\overline{CSR}$	$\overline{CSW}$	MODE
Définition de l'adresse (octet 1)	A6	A7	A8	A9	A10	A11	A12	A13	1	0	1
Définition de l'adresse (octet 2)	0	1	A0	A1	A2	A3	A4	A5	1	0	1
Ecriture de la donnée (octet 3)	D0	D1	D2	D3	D4	D5	D6	D7	1	0	0

Lire la VRAM :

1) Transférer les 8 bits les plus bas de l'adresse (en MODE haut)
2) Transférer les 8 bits les plus hauts de l'adresse (en MODE haut). Les 2 MSB doivent être mis à 0.
3) Lire un octet en MODE bas
4) Lier l'octet suivant

TABLEAU 4-4 – Lecture de la VRAM

OPERATION	MSB 0	1	2	3	4	5	6	LSB 7	$\overline{CSR}$	$\overline{CSW}$	MODE
Définition de l'adresse (octet 1)	A6	A7	A8	A9	A10	A11	A12	A13	1	0	1
Définition de l'adresse (octet 2)	0	0	A0	A1	A2	A3	A4	A5	1	0	1
Lecture de la donnée (octet 3)	D0	D1	D2	D3	D4	D5	D6	D7	0	1	0

EXEMPLE 4-2 :

Ecrire dans la VRAM

Supposons que nous voulions écrire la donnée 00H à l'adresse 20A0H de la VRAM. Le premier octet à transférer à l'adresse C002H sera la partie basse de l'adresse, soit A0H. L'octet suivant est la partie haute de l'adresse, avec les MSB mis à 1 et 0, donc on a 60H au lieu de 20H à transférer à l'adresse C002H. L'adresse étant définie, on peut transférer la donnée par l'adresse C000H.

EXEMPLE 4-3 :

Lire la VRAM

Supposons que nous souhaitions lire l'octet de l'adresse 20A0H de la VRAM. Le premier octet à transférer à l'adresse C002H sera la partie basse de l'adresse, soit A0H. L'octet suivant est la partie haute de l'adresse, avec les 2 MSB mis à 0. On envoie donc l'octet 20H à l'adresse C002H. L'adresse étant définie, on peut lire la donnée via l'adresse C000H.

5 DESCRIPTION DES REGISTRES DU VDP

5.1 Registres du VDP en écriture seule

Les 8 registres du VDP seront décrits dans les paragraphes suivants. Les registres 0 et 1 contiennent des bits permettant d'activer ou désactiver diverses fonctions et modes. Les registres 2 à 6 contiennent des valeurs qui spécifient les adresses de début des diverses tables en VRAM. Ces tables sont utilisées pour générer des affichages sur le plan des patrons ou sur les plans de sprites. Le registre 7 contient la couleur du texte (si on est en mode texte) et la couleur de fond pour tous les modes graphiques.

Dans certains registres, tous les bits ne sont pas utilisés. Afin d'assurer la compatibilité logicielle avec la génération future de VDP, ces bits inutilisés doivent être mis à 0.

NOTE

Le bit 0 est le MSB, le bit 7 le LSB.

5.1.1 Registre 0 (contient 2 bits de contrôle du VDP)

Registre 0

MSB							LSB
D0							D7
0	0	0	0	0	0	M3	EXT. VID.

Bit 6 = M3 (Pattern Mode Bit 3)

C'est un des 3 bits qui, lorsqu'ils sont spécifiés, déterminent le mode d'affichage du VDP.

Les autres bits de mode sont dans le registre 1.

M1	**M2**	**M3**	**Mode graphique**
0	0	0	Mode graphique 1
0	0	1	Mode graphique 2
0	1	0	Mode multicolore
1	0	0	Mode texte

Bit 7 = Plan VDP externe activé/désactivé

0 - désactive le plan VDP externe
1 - active le plan VDP externe

5.1.2 Registre 1 (contient 8 bits de contrôle du VDP)

Registre 1

MSB D0							LSB D7
4/16K	BLK. SCRN	IE	M1	M2	0	SPR. SIZE	SPR. MAG.

Bit 0 = 4/16 Ko Selection

0 – sélectionne 4 Ko de VRAM

1 – sélectionne 16 Ko de VRAM

NOTE

Ce bit n'est utilisé qu'avec les TMS 9918A/28A/29A. Lors de l'utilisation du TMS 9918/28/29, ce bit n'est pas utilisé et peut donc prendre n'importe quelle valeur. Le TMS 9918/28/29 part du principe que 16 Ko de VRAM sont présents.

Bit 1 = Display Blank Enable/Disable

0 – Met l'affichage actif en « blanc »

1 – Rend l'affichage normal

La mise en « blanc » de l'affichage consiste à désactiver les patrons et les sprites, ne laissant alors apparaître que le plan de fond. La mise en « blanc » ne supprime pas le contenu des différentes tables.

Bit 2 = IE (Interrupt Enable)

0 – désactive les interruptions

1 – active les interruptions

Si le système d'interruption du VDP est connecté et que ce bit est mis à 1, l'interruption se produira à la fin de l'aire d'affichage actif de l'écran, avant de recommencer le traçage vertical. Extraordinairement pratique, le nettoyage des patrons ou les mouvements de sprites peuvent être recalculés en écrivant dans le VDP le temps de l'interruption.

Bit 3,4 = M1, M2 (Pattern mode bit 1 et 2)

Voir la description du bit 6 du registre 0.

Bit 5 = bit réservé (doit rester à 0)

Bit 6 = Sprite Size Select

0 – Sélectionne une taille de sprites de 8x8 pixels (taille 0)

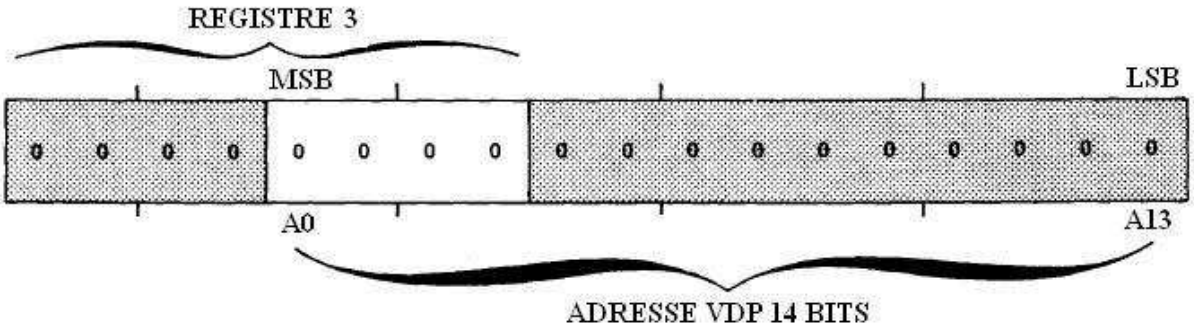
1 – Sélectionne une taille de sprites de 16x16 pixels (taille 1)

Bit 7 = Sprite Magnify Option

0 – Indique qu'aucun agrandissement n'est effectué

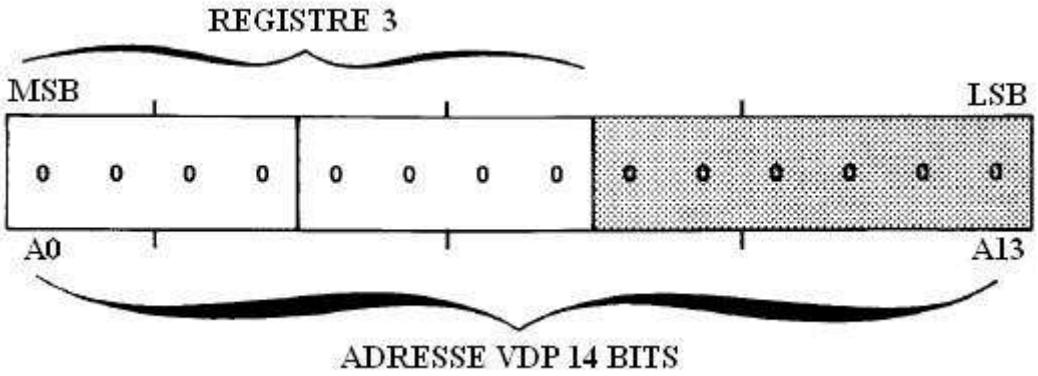
1 – Indique un agrandissement de 1, soit les sprites 8x8 qui deviennent 16x16 et les 16x16 qui deviennent des 32x32.

5.1.3 Registre 2



Le registre 2 indique au VDP à quel emplacement mémoire est située la table des noms de patrons. Son contenu peut varier de 0 à FH. Le contenu du registre forme les 4 premiers bits de l'adresse VDP 14 bits, définissant ainsi l'emplacement de la table des noms dans la VRAM égal à (registre 2) * 400H.

5.1.4 Registre 3

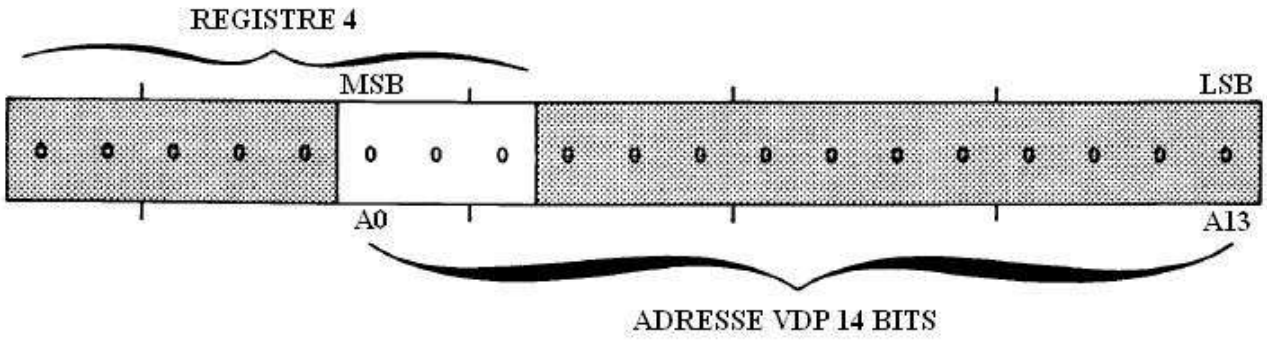


Le registre 3 indique au VDP à quel emplacement mémoire est située la table des couleurs. Son contenu peut varier de 0 à FFH. Le contenu de ce registre forme les 8 premiers bits de l'adresse VDP 14 bits, définissant ainsi l'emplacement de la table des couleurs dans la VRAM égal à (registre 3)* 40H.

NOTE

Le registre 3 fonctionne différemment quand le VDP est en mode graphique 2. Dans ce mode, la table des couleurs ne peut être située qu'à deux emplacements dans la VRAM, soit en 0000h ou en 2000H. Si vous souhaitez que l'adresse de la table des couleurs soit 2000H, vous devrez mettre le MSB du registre 3 à 1. Dans tous les cas, les autres bits du registre 3 doivent être à 1. Ainsi, en mode graphique 2, les 2 seules valeurs possibles pour le registre 3 sont FFH ou 7FH.

5.1.5 Registre 4

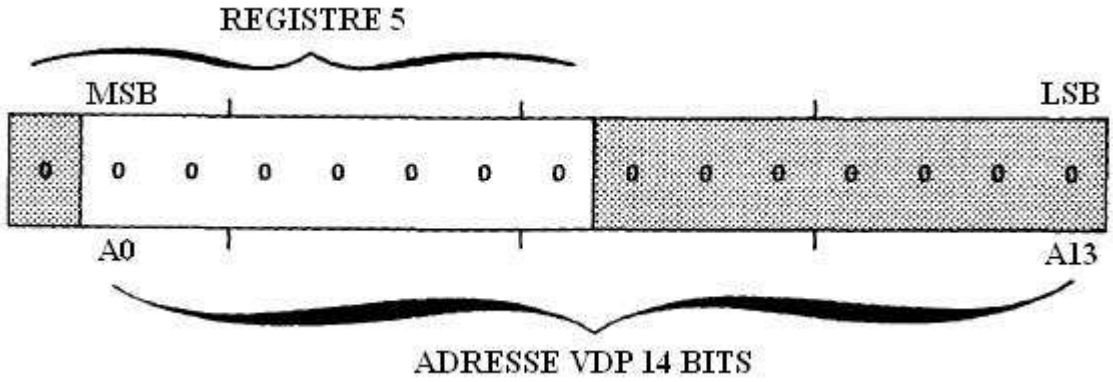


Le registre 4 indique au VDP à quel emplacement mémoire est située la table des patrons. Son contenu peut varier de 0 à 7. Le contenu de ce registre forme les 3 premiers bits de l'adresse VDP 14 bits, définissant ainsi l'emplacement de la table des patrons dans la VRAM égal à (registre 4)* 800H.

NOTE

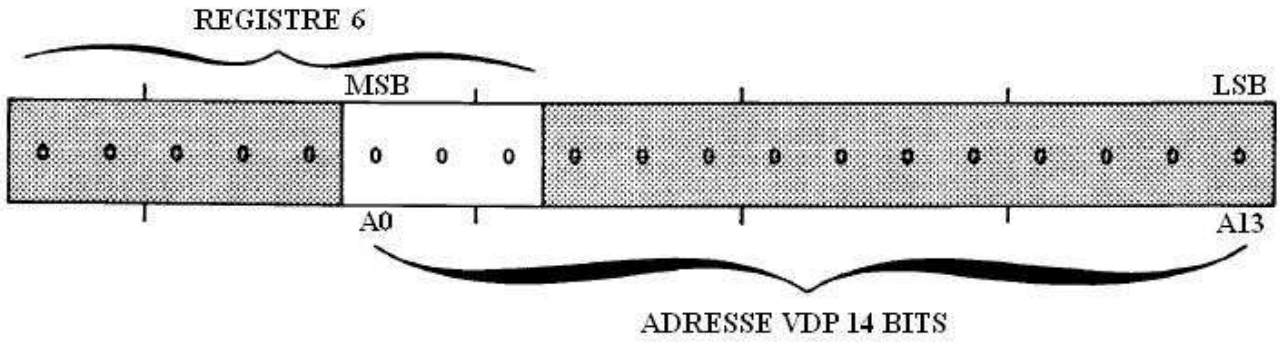
Le registre 4 fonctionne différemment quand le VDP est en mode graphique 2. Dans ce mode, la table des couleurs ne peut être située qu'à deux emplacements dans la VRAM, soit en 0000h ou en 2000H. Si vous souhaitez que l'adresse de la table des couleurs soit 2000H, vous devrez mettre le MSB du registre 4 à 1. Dans tous les cas, les autres bits du registre 4 doivent être à 1. Ainsi, en mode graphique 2, les 2 seules valeurs possibles pour le registre 4 sont 7 ou 3.

5.1.6 Registre 5



Le registre 5 indique au VDP à quel emplacement mémoire est située la table des attributs de sprite. Son contenu peut varier de 0 à 7FH. Le contenu de ce registre forme les 7 premiers bits de l'adresse VDP 14 bits, définissant ainsi l'emplacement de la table des attributs de sprites dans la VRAM égal à (registre 5)* 80H.

5.1.7 Registre 6



Le registre 5 indique au VDP à quel emplacement mémoire est située la table des patrons de sprite. Son contenu peut varier de 0 à 7. Le contenu de ce registre forme les 3 premiers bits de l'adresse VDP 14 bits, définissant ainsi l'emplacement de la table des patrons de sprites dans la VRAM égal à (registre 6)* 800H.

5.1.8 Registre 7

Registre 7

MSB D0							LSB D7
CLR 1	CLR 1	CLR 1	CLR 1	CLR 0	CLR 0	CLR 0	CLR 0

Les 4 premiers bits du registre 7 contiennent la couleur d'écriture (bits d'affichage à 1) en mode texte. Les 4 autres bits contiennent la couleur du plan de fond dans tous les modes et la couleur des bits d'affichage à 0 en mode texte.

5.2 Registre de statut en lecture seule

Registre de statut

MSB D0							LSB D7
FRAME FLAG	5ème SPR	C	NUMERO DU 5ème SPRITE				

Le registre de statut du VDP contient l'indicateur d'interruption, l'indicateur de coïncidence, l'indicateur de 5ème sprite et le numéro de 5ème sprite (s'il y en a un). Chacun d'entre eux est expliqué dans les paragraphes suivants.

5.2.1 Indicateur d'interruption (F)

L'indicateur F du registre de statut est mis à 1 à la fin du parcours de la dernière ligne de l'écran actif, juste avant que ne commence la couleur de fond au bas de l'écran. Il est remis à 0 si le registre de statut est lu ou quand le VDP est remis à 0 de façon externe (reset matériel). Si le bit d'interruption du registre 1 est actif (mis à 1), alors la ligne

d'interruption du VDP (INT) restera active (0) même quand l'indicateur F est à 1.

NOTE

Le registre de statut doit être lu image par image afin de « nettoyer » l'interruption et recevoir la nouvelle interruption de l'image suivante.

5.2.2 Indicateur de coïncidence (C)

L'indicateur C de coïncidence sera mis à 1 si un ou plusieurs sprites entrent en collision. Une collision arrive si au moins 2 sprites à l'écran ont au moins un pixel de chevauchement. Les sprites de couleur transparente ou ceux qui sont partiellement ou totalement en dehors de l'écran sont également pris en compte. L'indicateur C est remis à 0 si le registre de statut est lu ou quand le VDP est remis à 0 de façon externe (reset matériel).

5.2.3 Indicateur de 5ème sprite et numéro

L'indicateur de 5ème sprite est mis à 1 si 5 sprites actifs ou plus se trouvent sur la même ligne horizontale. Il est remis à 0 si le registre de statut est lu ou quand le VDP est remis à 0 de façon externe (reset matériel). Le numéro de sprite de plus basse priorité sur la ligne horizontale est chargé dans les 5 bits les moins significatifs du registre de statut chaque fois que l'indicateur de 5ème sprite est mis à 1. Cette mise à 1 ne génère pas d'interruption.

6 INITIALISER LE VDP

Après avoir allumé notre système VDP, la première chose à faire est d'initialiser les registres. Afin de bien le faire, il nous faut savoir plusieurs choses, comme quel mode d'affichage utiliser et où implanter les différentes tables nécessaires dans la VRAM. La figure 6-1 montre une procédure d'initialisation des 8 registres. La présente section est un bref descriptif des usages populaires de chaque mode.

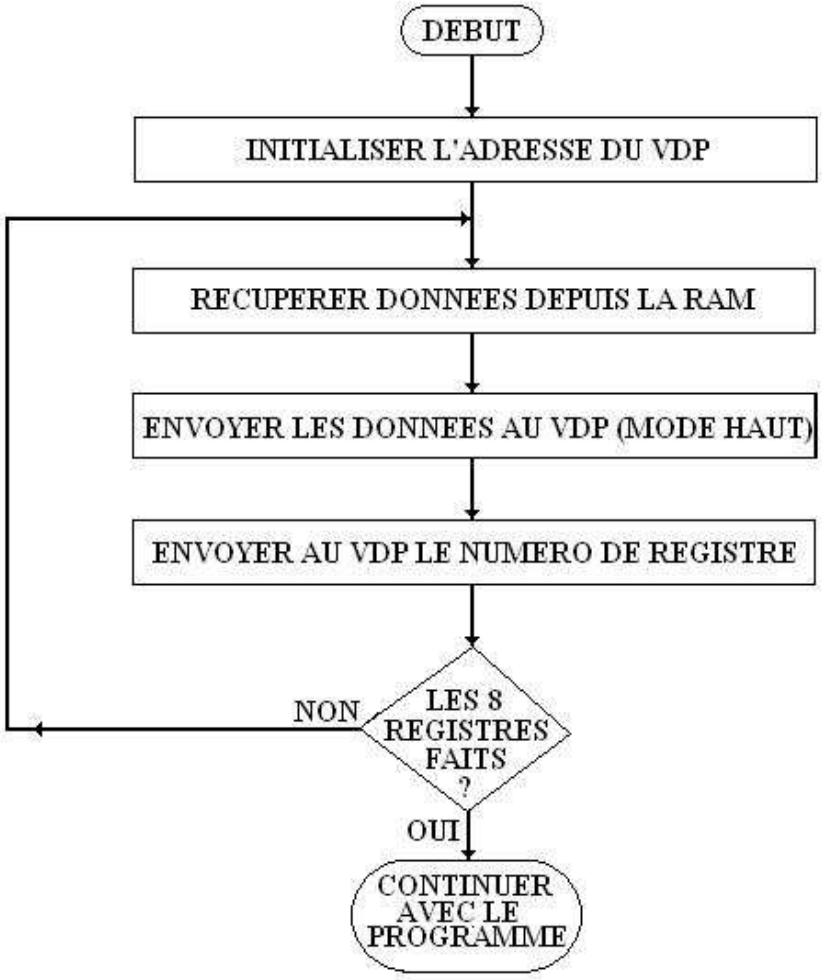


FIGURE 6-1 - INITIALISATION DES REGISTRES

6.1 Choisir le bon mode

La plupart des applications qui affichent du texte utilisent les modes texte ou graphique 1. Les jeux vidéo qui nécessitent des graphismes haute résolution utilisent les modes graphiques 1 ou 2. Le mode graphique 1 est un peu plus usité pour les jeux, car une image colorée, détaillée et de haute résolution peut être créée en utilisant un minimum de données. Le mode graphique 2 est utilisé quand on a besoin d'un grand niveau de détail et de couleurs pour une image haute résolution, ou quand on souhaite organiser la mémoire au niveau bitmap pour le calcul de points, de lignes, de cercles, etc. Le mode graphique 2 en bitmap est aussi très usité dans les cas d'applications graphiques professionnelles. Le mode multicolore est usité dans les jeux qui ne requièrent qu'un affichage basse résolution. Les sprites sont disponibles dans tous les modes excepté le mode texte et sont utilisés en priorité pour les objets qui bougent et changent de forme (animation).

Les descriptions détaillées des modes graphiques 1, 2, texte et multicolore sont au chapitre 8. Référez vous à ce chapitre pour décider quel type d'affichage choisir en fonction de votre application.

Quelques valeurs usuelles pour les tables pour l'initialisation des registres seront abordées dans les illustrations suivantes. Le plan de la VRAM résultante sera montré après les valeurs des tables.

Ces valeurs usuelles ne sont là qu'à titre d'exemple. Si vous préférez un autre plan pour la VRAM, vous pourrez calculer vos emplacements en vous basant sur ce qui a été dit au chapitre 5.

6.1.1 Initialisation du mode graphique 1

TABLEAU 6-1 – Initialisation du mode graphique 1

REGISTRE	MSB LSB	HEX	DESCRIPTION
R0	00000000	00	Mode graphique 1, pas de vidéo externe
R1	11000000	C0	16 K, active affichage, désactive interruption, sprites en 8x8, pas d'agrandissement
R2	00000101	05	Adresse de la table des noms = 1400H
R3	10000000	80	Adresse de la table des couleurs = 2000H
R4	00000001	01	Adresse de la table des patrons = 0800H
R5	00100000	20	Adresse de la table des attributs de sprite = 1000H
R6	00000000	00	Adresse de la table des patrons de sprite = 0000H
R7	00000001	01	Couleur de fond = noir

FIGURE 6-2 - PLAN DE LA VRAM EN MODE GRAPHIQUE 1

6.1.2 Initialisation du mode graphique 2

TABLEAU 6-2 – Initialisation du mode graphique 2

REGISTRE	MSB LSB	HEX	DESCRIPTION
R0	00000010	02	Mode graphique 2, pas de vidéo externe
R1	11000010	C2	16 K, active affichage, désactive interruption, sprites en 16x16, pas d'agrandissement
R2	00001110	0E	Adresse de la table des noms = 3800H
R3	11111111	FF	Adresse de la table des couleurs = 2000H
R4	00000011	03	Adresse de la table des patrons = 0000H
R5	01110110	76	Adresse de la table des attributs de sprite = 3B00H
R6	00000011	03	Adresse de la table des patrons de sprite = 1800H
R7	00001111	0F	Couleur de fond = blanc

FIGURE 6-3 - PLAN DE LA VRAM EN MODE GRAPHIQUE 2

6.1.3 Initialisation du mode multicolore

TABLEAU 6-3 – Initialisation du mode multicolore

REGISTRE	MSB LSB	HEX	DESCRIPTION
R0	00000000	00	Mode multicolore, pas de vidéo externe
R1	11001011	CB	16 K, active affichage, désactive interruption, sprites en 16x16, agrandissement
R2	00000101	05	Adresse de la table des noms = 1400H
R3	XXXXXXXX	XX	Table des couleurs non utilisée, bits ignorés
R4	00000001	01	Adresse de la table des patrons = 0800H
R5	00100000	20	Adresse de la table des attributs de sprite = 1000H
R6	00000000	00	Adresse de la table des patrons de sprite = 0000H
R7	00000100	04	Couleur de fond = bleu foncé

FIGURE 6-4 - PLAN DE LA VRAM EN MODE MULTICOLORE

6.1.4 Initialisation du mode texte

TABLEAU 6-4 – Initialisation du mode texte

REGISTRE	MSB LSB	HEX	DESCRIPTION
R0	00000000	00	Mode texte, pas de vidéo externe
R1	11010000	D0	16 K, active affichage, désactive interruption
R2	00000010	02	Adresse de la table des noms = 0800H
R3	XXXXXXXX	XX	Table des couleurs non utilisée, couleurs définies en R7
R4	00000000	00	Adresse de la table des patrons = 0000H
R5	XXXXXXXX	20	
R6	XXXXXXXX	00	
R7	11110101	F5	Texte blanc sur fond bleu

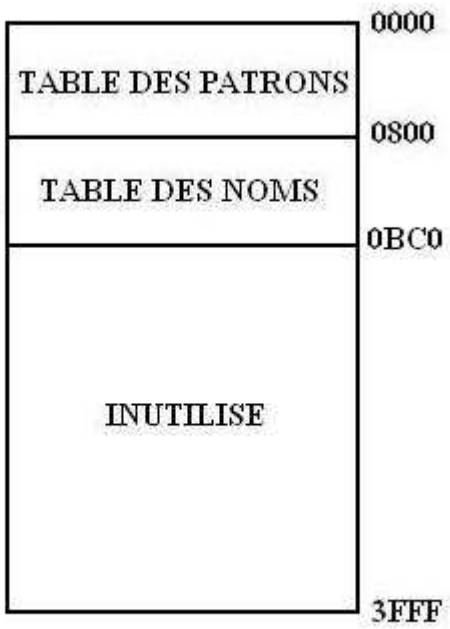


FIGURE 6-5 - PLAN DE LA VRAM EN MODE TEXTE

7 CREATION DE PATRONS

7.1 Tous les patrons sont créés de la même façon

Si vous savez créer des patrons de 8x8 pixels, alors vous savez créer des polices d'écriture ou des motifs pour le mode texte, le mode graphique 1, le mode graphique 2, et également des sprites. Dans les pages suivantes, nous définirons quelques patrons et montrerons comment ils doivent être placés en VRAM pour réaliser leur affichage.

1) La figure 7-1 est une grille qui sera utilisée pour créer des patrons de 8x8 pixels. Chaque petit carré à l'intérieur de la grille représente un pixel à l'écran.

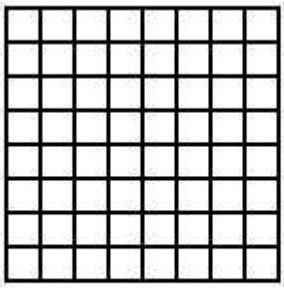


FIGURE 7-1 - GRILLE DE PATRON 8x8 PIXELS

2) Remplissez les carrés de la grille afin de créer votre patron de texte, motif graphique ou sprite. Les exemples pour la lettre A, une flèche et une étoile sont montrés dans la figure 7-2.

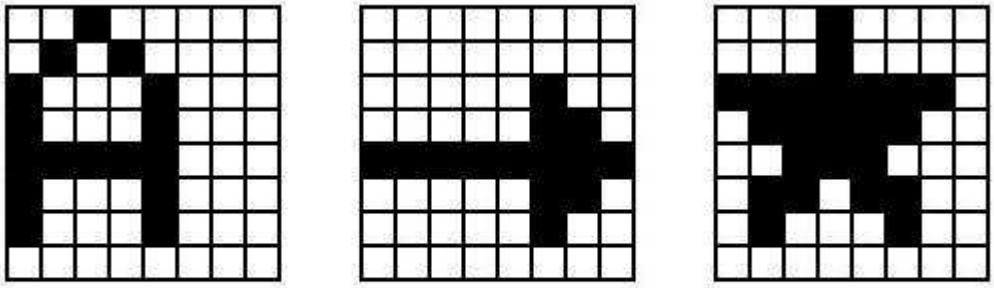


FIGURE 7-2 - EXEMPLES DE CREATION DE PATRONS

NOTE

Si vous définissez un patron destiné à être utilisé en mode texte, celui-ci devra être justifié à gauche dans un bloc de 6x8 pixels, comme le A de la figure 7-2. Voyez le chapitre 8.5 pour plus de précisions.

3) Maintenant vient le travail de conversion numérique : d'abord, il faut assigner 1 aux carrés remplis et 0 aux vides. Convertissez alors les 1 et 0 en leur équivalent hexadécimal, comme sur la figure 7-3.

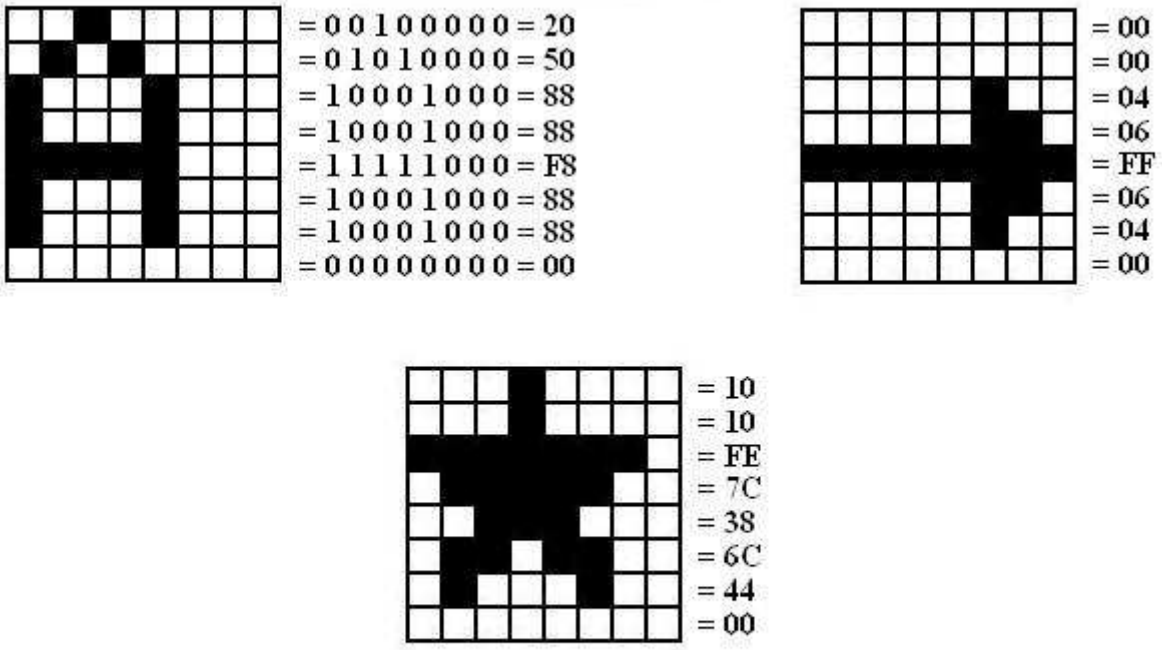


FIGURE 7-3 - CONVERSION HEXADECIMALE

4) Placez maintenant les 8 octets qui définissent le patron dans la table des patrons. On prend comme hypothèse de travail que la table des patrons commence en VRAM à l'adresse 800H, et que la flèche sera le patron numéro 0. Placez ensuite les 8 octets, comme indiqué sur la figure 7-4.

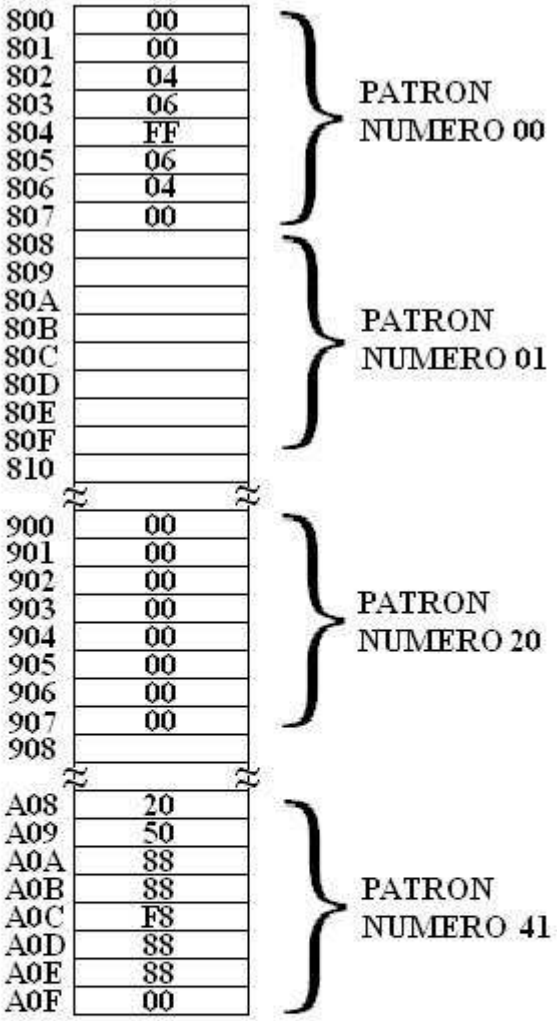


FIGURE 7-4 - TABLE DES PATRONS

7.1.1 Définir des patrons pour le texte

Lorsque vous utilisez du texte dans vos applications, il peut être pratique de place les 8 octets définissant le caractère dans l'emplacement de son code ASCII courant. Cela simplifiera l'écriture du texte à l'écran. Ecrire la valeur du code ASCII dans la table des noms implique l'apparition du caractère concerné à l'écran. Comme c'est montré dans l'exemple 7-1, le caractère « espace » est en position 20H dans la table des patrons, et la lettre A en position 41H.

EXEMPLE 7-1

ASCII	Espace	= 20H
	?	= 3FH
	A	= 41H
	B	= 42H
	C	= 43H
	Etc.	

NOTE

Quand on définit des patrons pour le mode texte, le patron doit être contenu dans une grille de 6x8 pixels, comme la figure 7-5 l'illustre. Les 2 LSB sont inutilisés et donc non affichés par le VDP.

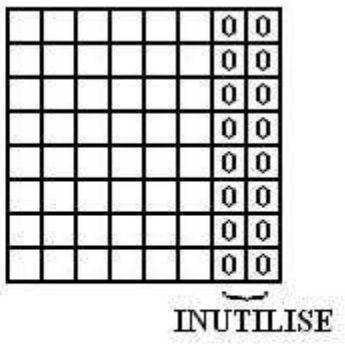


FIGURE 7-5 - GRILLE DE PATRON 6x8 PIXELS POUR LE MODE TEXTE

7.1.2 Définir des patrons de sprite

1) Pour utiliser des sprites de taille 0 (8x8 pixels), les patrons sont définis exactement comme la flèche et l'étoile réalisés précédemment, avec un changement. Au lieu d'entrer le code dans la table des patrons, on l'entre dans la table des patrons de sprites (ou table génératrice de sprites). La figure 7-6 illustre une grille de sprite et la table génératrice de sprites pour un patron de 8x8 pixels.

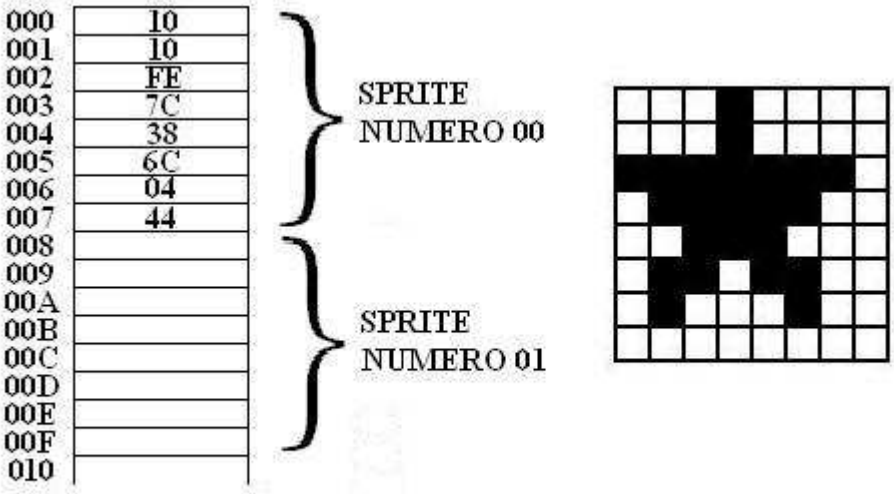


FIGURE 7-6 - GRILLE DE SPRITE 8x8 ET TABLE GENERATRICE DE SPRITES

2) Si vous utilisez les sprites de taille 1 (16x16 pixels), alors les patrons seront toujours définis en motifs de 8x8 pixels. Il faut 4 motifs 8x8 pixels pour obtenir une grille de 16x16 pixels comme le montre la figure 7-7.

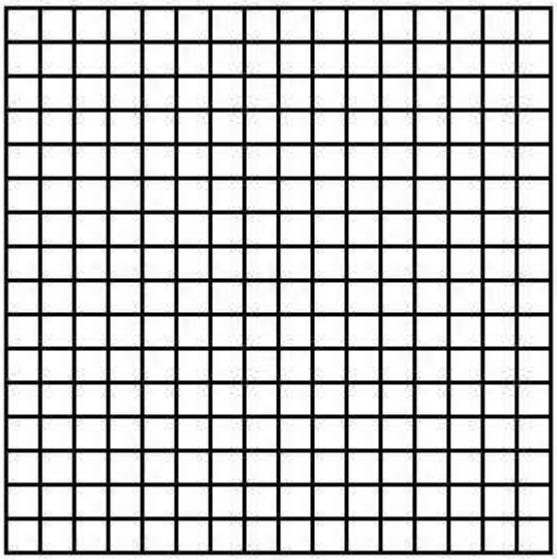


FIGURE 7-7 - GRILLE DE SPRITE 16x16 PIXELS

3) Remplissez les carrés pour créer votre patron de sprite de taille 1. Un exemple vous est montré en figure 7-8.

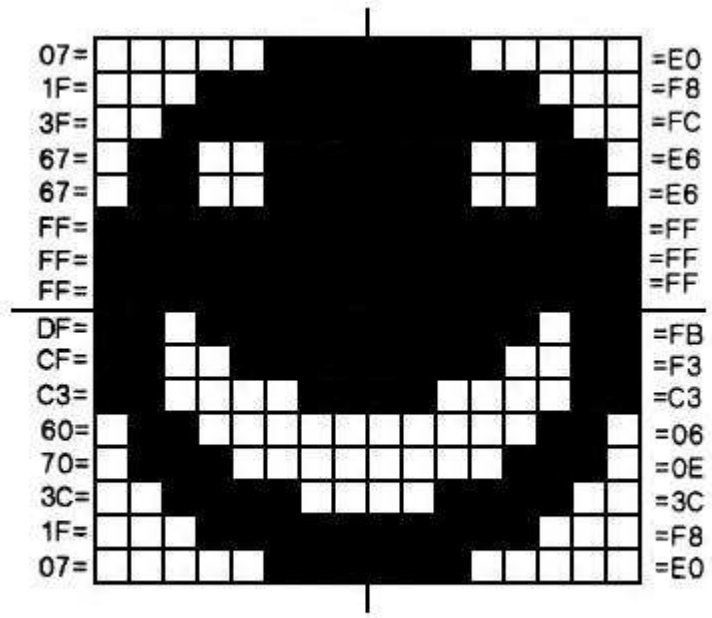


FIGURE 7-8 - PATRON DE SPRITE TAILLE 1

4) Encodez ensuite le patron de sprite. Cela se fait en divisant le sprite en 4 morceaux comme le montre la figure 7-9. Les 4 motifs de 8x8 pixels seront encodés dans l'ordre suivant :

   Motif 1 : en haut à gauche

   Motif 2 : en bas à gauche

   Motif 3 : en haut à droite

   Motif 4 : en bas à droite

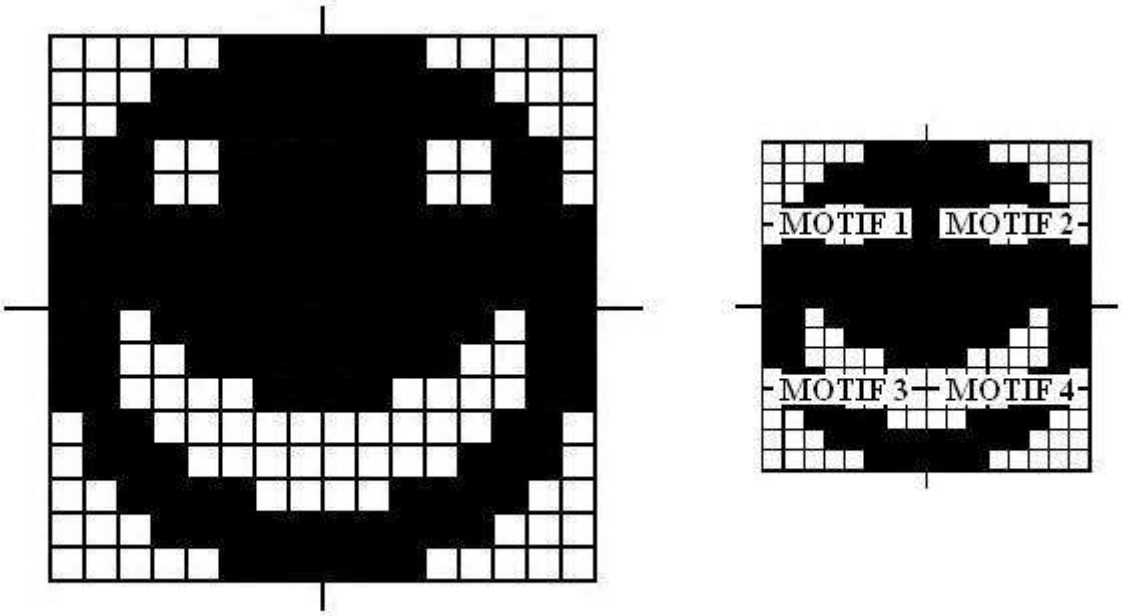


FIGURE 7-9 - ORGANISATION DES SPRITES EN TAILLE 1

5) Placez les 32 octets d'informations dans la table génératrice de sprites, en acceptant qu'ici la table est située à l'adresse 0000H de la VRAM. La figure 7-10 illustre comment se présente la table génératrice de sprites pour un sprite de 16x16 pixels.

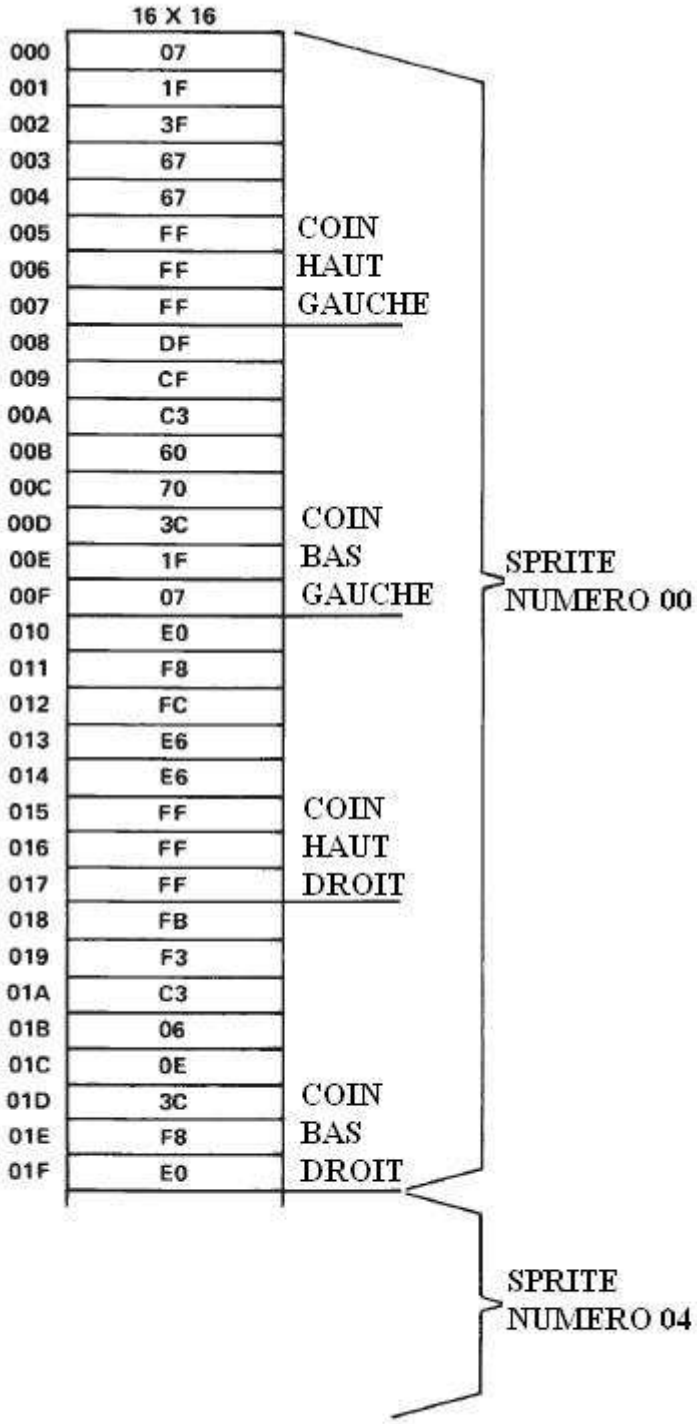


FIGURE 7-10 - TABLE GENERATRICE DE SPRITES

8 Les modes d'affichage

8.1 Mode graphique 1

Le VDP est en mode graphique 1 lorsque les 3 bits de mode (M1, M2 et M3) des registres R0 et R1 sont à 0. Dans ce mode, le plan a une résolution de 256 pixels horizontaux sur 192 pixels verticaux. L'écran est divisé en blocs contenant chacun un patron de 8x8 pixels. Il y a 32 blocs horizontaux sur 24 blocs verticaux. La figure 8-1 illustre la position de ces 768 blocs à l'écran.

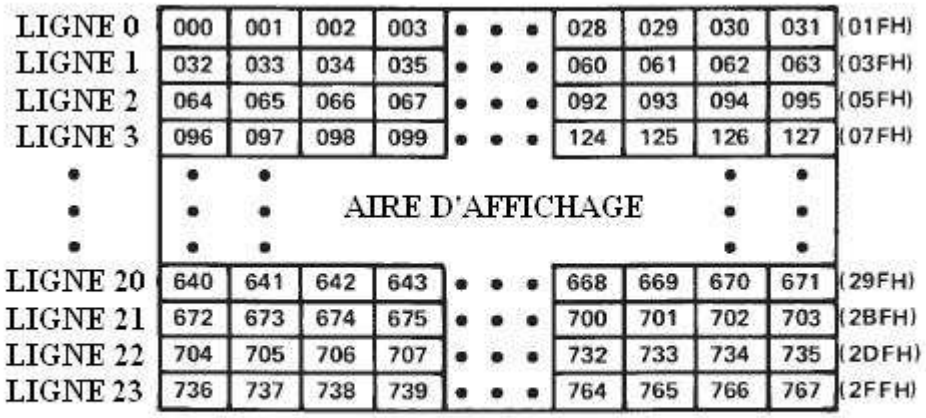


FIGURE 8-1 - PLAN DE LA TABLE DES NOMS EN MODE GRAPHIQUE 1

Trois tables sont requises en VRAM pour créer une image en mode graphique 1, ces tables sont la table des noms, la table des patrons et la table des couleurs. Si tous les bits possibles pour les patrons et couleurs sont définis, une image en mode graphique 1 prendra au maximum 2848 octets (B20H).

8.1.1 Table des patrons

La table des patrons contient une bibliothèque de patrons définis par l'utilisateur qui peuvent être affichés à chacune des 768 positions de l'écran. Elle fait 2048 octets de long, et est constituée de 256 patrons de 8 octets. Chacun de ces patrons de 8 octets définit une surface de 8x8 pixels. Tous les 1 à l'intérieur d'un patron correspondent à une couleur (qu'on appellera couleur 1) et tous les 0 à une autre couleur (couleur 0).

Une caractéristique du mode graphique 1, par opposition au graphiques bitmap, est le fait qu'une fois qu'un caractère a été défini et enregistré dans la table des patrons, il peut être utilisé de nombreuses fois à l'écran sans avoir à être redéfini.

Exemple 8-1

Si seul le premier patron de 8 octets est défini (patron numéro 0), vous pouvez placer ce patron à chacune des 768 places possibles en écrivant 00H pour chaque octet de la table des noms.

8.1.2 Table des noms

Comme illustré dans la figure 8-1, il y a 768 emplacements pour un écran. Chacun de ces emplacements est représenté par un octet de mémoire dans la table des noms. Le premier octet de la table des noms spécifie quel patron sera placé dans le coin supérieur

gauche de l'écran. Le dernier octet de la table des noms spécifie quel patron sera placé dans le coin inférieur droit de l'écran. Chaque entrée d'octet dans la table des noms désigne un des 256 (FFH) patrons. L'emplacement cette table des noms de 768 octets en VRAM est défini par l'adresse de base donnée dans le registre 2 du VDP.

8.1.3 Table des couleurs

La table des couleurs fait 32 octets de long en mode graphique 1. Son emplacement dans la VRAM est déterminé par les 8 bits d'adresse de base du registre 3 du VDP.
La couleur des 1 et des 0 à l'intérieur d'un patron est donné par la table des couleurs. Chaque entrée d'octet dans la table des couleurs définit 2 couleurs. Les 4 MSB donnent la couleur des 1, les 4 LSB celle des 0. Ainsi nous pouvons créer 256 uniques patrons de 8x8 pixels, mais avec seulement 32 entrées possibles, chaque entrée devant alors définir les couleurs pour plus d'un patron. En fait, le premier octet de la table des couleurs définit la couleur pour les 8 premiers patrons, le deuxième octet définit la couleur pour les 8 suivants et ainsi de suite.
Le tableau 8-1 illustre la table des couleurs en mode graphique 1.
la figure 8-2 illustre comment les différentes tables interagissent avec l'écran.

TABLEAU 8-1 – Table des couleurs en mode graphique 1

Octet n°	Patron N°	Octet n°	Patron N°
0	0..7	16	128..135
1	8..15	17	136..143
2	16..23	18	144..151
3	24..31	19	152..159
4	32..39	20	160..167
5	40..47	21	168..175
6	48..55	22	176..183
7	56..63	23	184..191
8	64..71	24	192..199
9	72..79	25	200..207
10	80..87	26	208..215
11	88..95	27	216..223
12	96..103	28	224..231
13	104..111	29	232..239
14	112..119	30	240..247
15	120..127	31	248..255

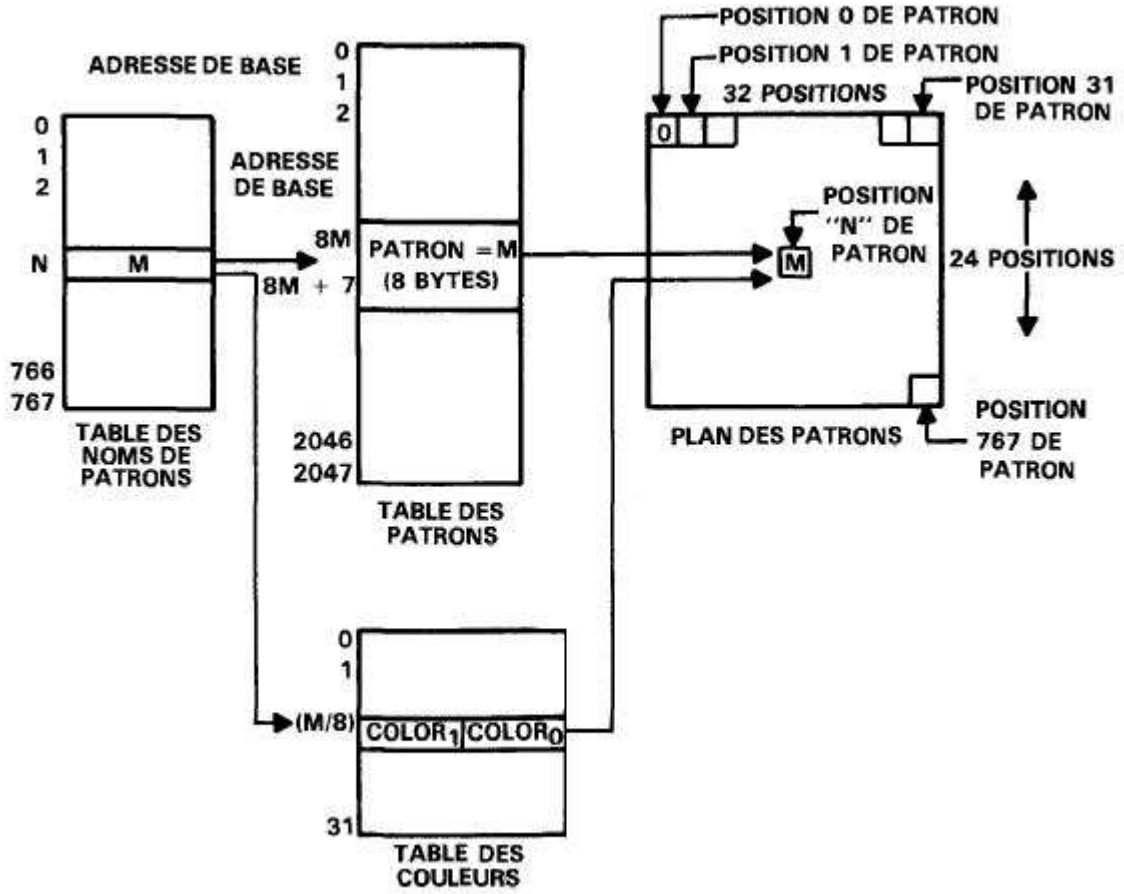


FIGURE 8-2 - PLAN D'ECRAN EN MODE GRAPHIQUE 1

8.2 Mode graphique 2

Le mode graphique 2 est très similaire au mode graphique 1 dans la façon dont l'écran est organisé. La résolution est toujours de 256 pixels horizontaux sur 192 pixels verticaux. Trois tables sont toujours nécessaires en VRAM pour générer un affichage : la table des noms, la table des patrons et la table des couleurs. La table des noms fait toujours 768 octets de long, mais la longueur des tables des patrons et des couleurs est plus grande. Au lieu d'avoir à choisir dans une bibliothèque de 256 motifs de 8x8 pixels pour l'affichage dans les 768 emplacements (ce qui implique que les patrons doivent être ré-employés), on peut définir 768 patrons de 8x8 pixels en mode graphique 2. Ceci permet la création d'un motif unique pour chaque emplacement à l'écran. Au lieu d'un octet de couleur pour 8 patrons, il y a maintenant 8 octets d'information par patron, rendant ainsi les tables des patrons et de couleur de même longueur.

Comme il y a 8 octets d'information de couleur par patron, 2 couleurs uniques peuvent être être spécifiées pour chaque ligne du patron de 8x8 pixels. Cela permet d'avoir jusqu'à 16 couleurs dans un seul patron.

8.3 Table des patrons

La table des patrons fait 6144 (1800H) octets de long, partant du principe que tous les patrons sont définis, et est considérée comme 3 blocs égaux de 2048 octets d'informations de patrons. Chacun de ces 3 blocs est divisé en 256 blocs de 8x8 pixels.

Les 256 premiers patrons ne peuvent être affichés que dans le tiers supérieur de l'écran, les 256 suivants dans le tiers central et les 256 derniers dans le tiers inférieur de l'écran.

8.3.1 Table des couleurs

La table des couleurs fait 6144 (1800H) octets de long, partant du principe que toutes les couleurs sont définies, et est partagée en 3 blocs exactement comme la table des patrons. Chaque bloc de 2048 octets est divisé en 256 définitions de couleur, chacune faisant 8 octets de long. Les 256 premières couleurs correspondent directement aux 256 premiers patrons définis, et ainsi de suite.
Il faut 8 octets pour définir la forme d'un patron et huit octets pour en définir la couleur. Chaque octet dans une définition de couleur définit la couleur des bits allumés ou non sur la ligne correspondante du patron. Les 4 bits les plus hauts définissent la couleur des bits allumés (1), les 4 plus bas la couleur des bits éteints (0). Un exemple de la manière dont est représentée la couleur est montré dans la figure 8-3.

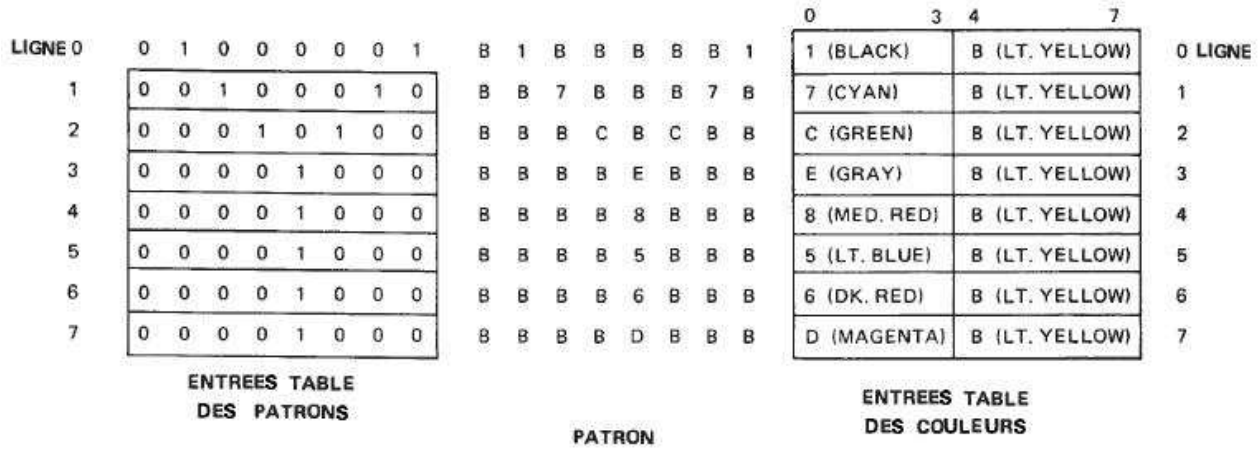


FIGURE 8-3 - REPRESENTATION DU MAPPING COULEUR/PATRON

8.4 Table des noms

Comme en mode graphique 1, la table des noms comprend 768 entrées qui correspondent aux 768 positions de patrons à l'écran. Comme chaque entrée de la table des noms fait au maximum un octet, on ne peut spécifier qu'un parmi 256 (FFH) patrons. Afin de pouvoir définir un patron unique pour chacune des 768 positions, l'écran est divisé en trois parties (voir figure 8-4). Chacune de ces sections fait 256 octets de long et ainsi, comme un octet peut prendre 256 valeurs différentes, un patron unique peut être spécifié pour chaque emplacement à l'écran. Un exemple de plan d'écran en mode graphique 2 est consultable en figure 8-5.

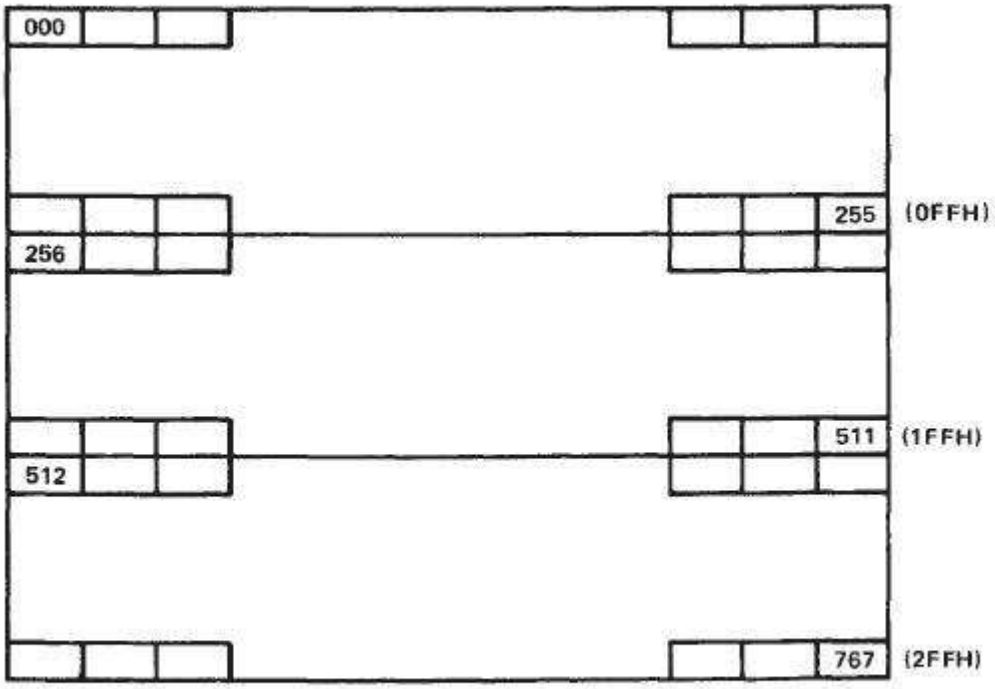


FIGURE 8-4 - TABLE DES NOMS DIVISEE EN 3 BLOCS EN MODE GRAPHIQUE 2

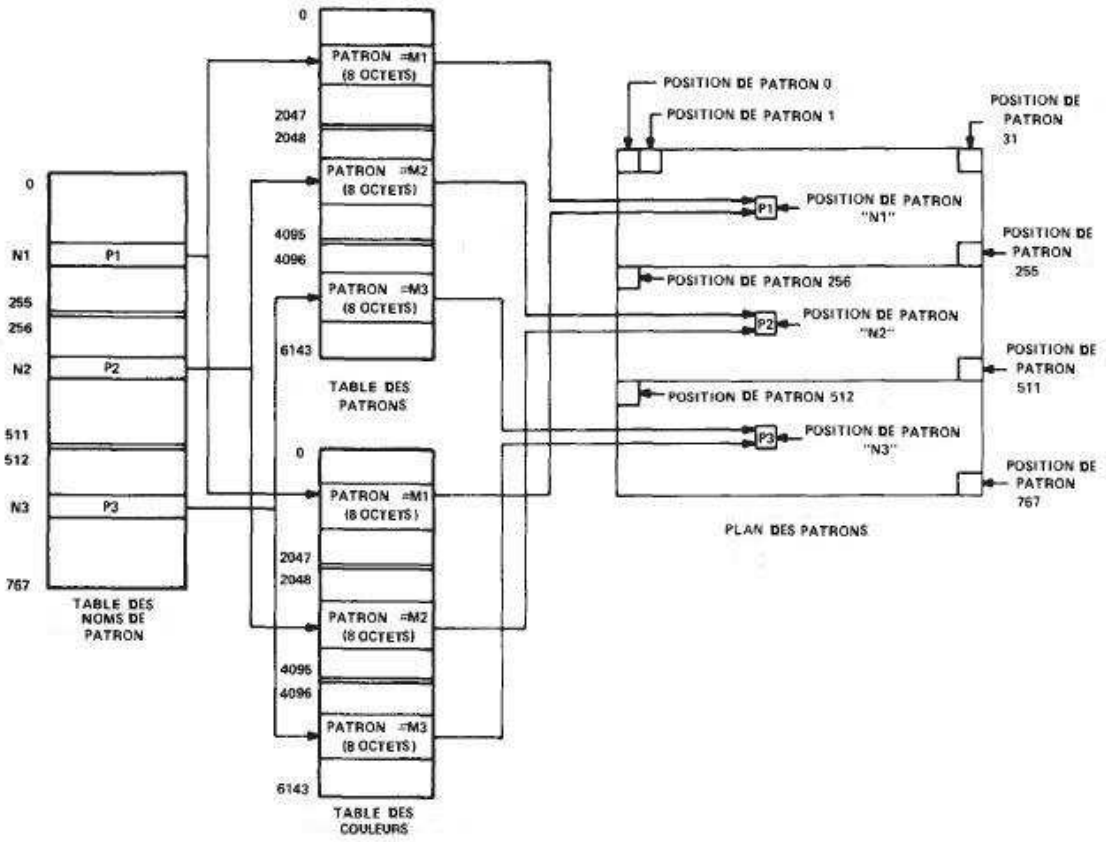


FIGURE 8-5 - PLAN D'ECRAN EN MODE GRAPHIQUE 2

8.4.1 Mode graphique 2 en affichage bitmap

Une fonctionnalité ingénieuse du mode graphique 2 est qu'il peut être arrangé par le programmeur pour être utilisé en affichage bitmap. C'est très utile, surtout lorsque votre application permet l'utilisation d'un algorithme permettant le calcul des positions de pixels plutôt que de les dessiner « à la main », en codant chaque pixel à la fois. Le mode bitmap vous permet d'adresser chaque pixel à l'écran, afin d'allumer des points, dessiner des lignes, des cercles, etc. Le seul inconvénient étant que bien que le plan des patrons soit entièrement en bitmap, l'assignation des couleurs ne l'est pas. Ainsi une couleur unique par pixel ne peut pas être spécifiée, deux possibilités s'offrent alors : utiliser plus de deux couleurs, mais attention aux endroits où elles sont appliquées, ou utiliser 2 couleurs (pixels allumés d'une couleur, pixels éteints d'une autre), sans problème d'application.

La technique pour passer en mode bitmap pour le mode graphique 2 est d'écrire une valeur différente pour chacune des 768 entrées de la table des noms. Ce qui signifie que le VDP appliquera une unique entrée de la table des patrons pour chaque position à l'écran. En écrivant un octet dans une entrée à 8 octets de la table des patrons, chaque pixel à l'écran peut être allumé ou éteint.

La manière la plus simple pour illustrer ce point est d'écrire la même valeur à chaque entrée de la table des couleurs. Une valeur de couleur de 4FH à chaque entrée de la table des couleurs rendra une couleur d'écriture bleue sur fond blanc (les pixels allumés en blanc, les pixels éteints en bleu).

Comme indiqué plus haut, pour spécifier le bitmap, un unique patron est défini pour chaque entrée de la table des noms. Une manière organisée de faire cela est d'écrire 00H à la première entrée de la table des noms, 01H pour la seconde, 02H pour la troisième et ainsi de suite. Après avoir atteint FFH, le processus est répété deux fois afin que la table des noms contienne bien 3 plages de valeurs de 00H à FFH.

A ce point, nous pouvons oublier la table des noms et la table des couleurs pour nous concentrer sur la table des patrons. Chaque bit d'un octet d'une entrée de la table des patrons représente maintenant un unique point à l'écran. La figure 8-6 illustre comment la table des patrons est représentée à l'écran. On partira du principe que le début de la table des patrons est à l'adresse 0000H de la VRAM.

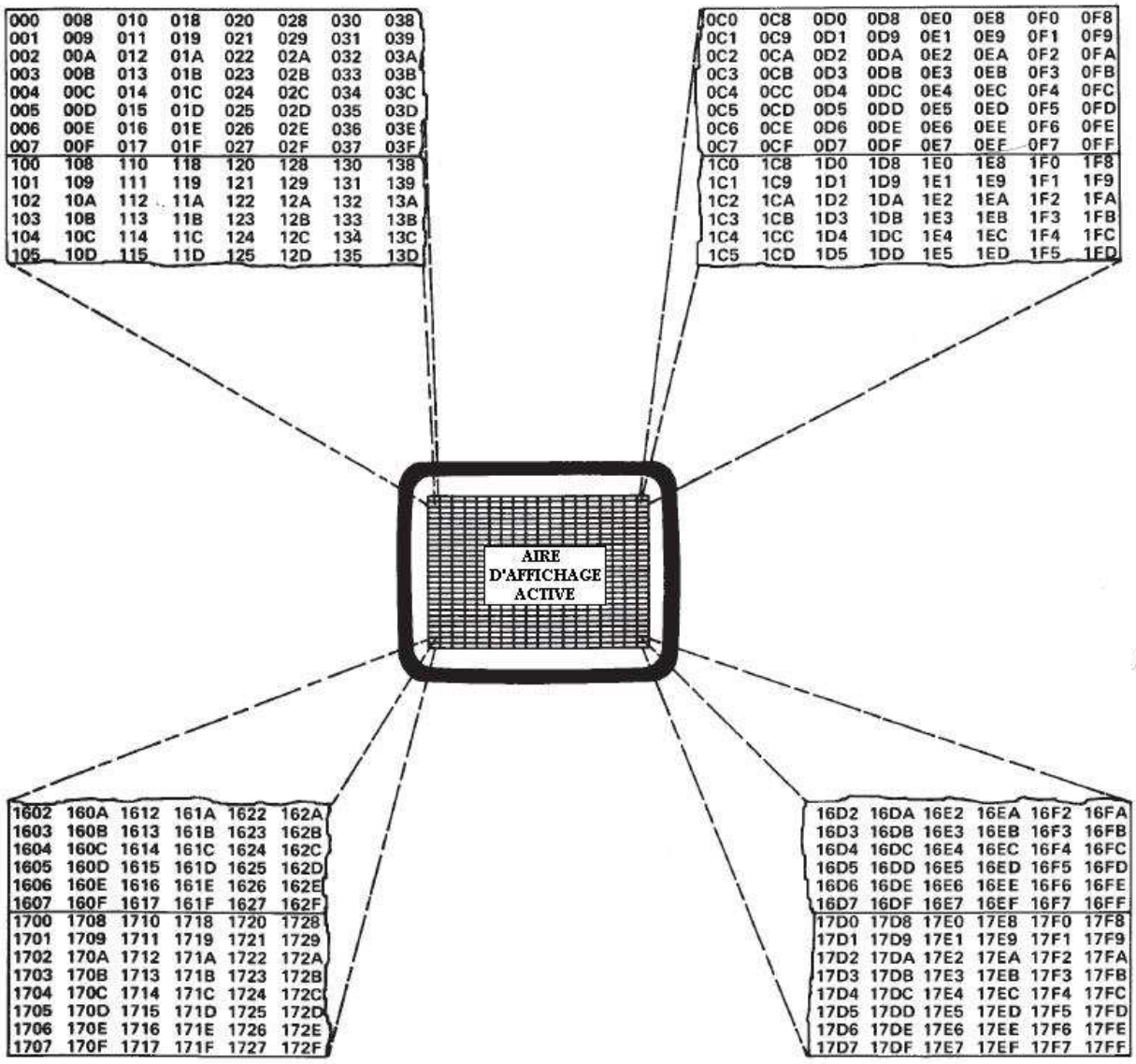


FIGURE 8-6 - TABLE DES PATRONS EN MODE GRAPHIQUE 2 ARRANGEE POUR GRAPHIQUES EN BITMAP

En regardant cette illustration, nous pouvons voir qu'afin d'allumer le pixel qui serait dans le coin supérieur gauche, nous devrions écrire 80H comme premier octet de la table des patrons (situé en 0000H). De même, pour allumer le pixel du coin inférieur droit, il faudrait mettre la valeur 01H à l'adresse 17FFH.

Arrivés à ce point, nous pouvons écrire une routine qui, avec des coordonnées X et Y données, nous donnera l'adresse de l'octet à écrire et la valeur de la donnée à écrire. Ce qui suit est une procédure pas à pas sur la manière de calculer l'adresse et la donnée.

Exemple 8-2

DONNEES :
X = 00H – FFH ou 0 – 255
Y = 00H – C0 ou 0 – 192

1) Prendre la valeur entière de (X/8) et la multiplier par 8. La résultante est la partie horizontale de l'octet. Ce que nous devons tracer est déterminé par le reste, s'il y en a un, de la division précédente.
2) Prendre la valeur entière de (Y/8) et la multiplier par 100H. La résultante est la partie verticale de l'octet. S'il y a un reste, il faut l'ajouter à la composante verticale de l'octet. On a ainsi l'adresse verticale de départ.
3) Additionner la composante horizontale de l'octet et l'adresse verticale de départ. On obtient l'adresse de l'octet dans lequel nous devrons écrire une donnée afin de tracer notre point.
4) Utiliser le reste de (X/8) pour regarder dans la table suivante quelle donnée afficher. Les valeurs suivant le reste sont les suivantes :

Reste (X/8)	**Valeur à écrire**
0	80H
1	40H
2	20H
3	10H
4	08H
5	04H
6	02H
7	01H

L'équation décrite dans le paragraphe précédent peut être écrite ainsi :

ADRESSE DE L'OCTET = 8(INT(X/8)) + 256(INT(Y/8)) + R(Y/8)

avec R(Y/8) correspondant au reste de la division de Y/8.

La donnée à écrire est à trouver grâce au reste de la division (X/8) en prenant la valeur adéquate du tableau.

8.4.2 Jouer avec l'adressage de la VRAM

Nous avons vu dans les chapitres précédents comment utiliser le mode graphique 2 normalement ou en mode bitmap. Nous allons maintenant expliquer d'autres astuces à tenter sur le VDP. En bricolant les valeurs des registres R2 à R6 (en entrant des valeurs non standards), on peut obtenir des effets intéressants.
Vous êtes averti que jouer avec les adresses de la VRAM peut donner des effets intéressants, mais aussi presque toujours causer des effets indésirables comme perdre la possibilité d'utiliser les sprites, ou de ne pouvoir en utiliser qu'un petit nombre. Plutôt qu'en dire trop sur ce sujet, nous allons décrire une nouvelle configuration intéressante qui peut être obtenue et vous laisser le reste.
Le tableau 8-2 montre les initialisations de registres pour ce mode particulier. Notez que

les seuls registres à comporter des valeurs inhabituelles sont les registres 3 et 4 qui déterminent la table des couleurs et la table des patrons.

TABLEAU 8-2 – Valeurs d'initialisation du nouveau mode

Registre	MSB LSB	HEX	Description
R0	00000010	02	Mode graphique 2, pas de vidéo externe
R1	11000010	C2	16 K, autorise affichage, interdit interruptions, sprites 16x16, pas d'agrandissement
R2	00001110	0E	Adresse table des noms = 3800H
R3	10011111	9F	Adresse table des couleurs = 2000H à 2800H
R4	00000000	00	Adresse table des patrons = 0000H à 0800H
R5	01110110	76	Adresse table attributs sprite = 3B00H
R6	00000011	03	Adresse table génératrice des sprites = 1800H
R7	00001111	0F	Couleur de fond = blanc

Ce que fait ce mode est de réduire la taille des tables des couleurs et des patrons de 1800H à 800H octets. Ce qui nous permet de définir jusqu'à 256 patrons de 8x8 pixels et leur patron de couleur de 8 octets correspondant. L'application des couleurs est celle du mode graphique 2 habituel.

La table des noms de 768 octets n'est plus divisée en 3 sections mais fonctionne comme en mode graphique 1. Un octet d'information écrit n'importe où dans la table des noms sélectionnera le patron approprié et la couleur 8 octets associée et le placera à l'écran.

Ce mode est utile car il permet les économies de mémoire du mode graphique 1 tout en bénéficiant des qualités d'affichage du mode 2. Cependant, un seul patron pour chaque emplacement d'écran ne peut plus être défini, ce qui est nécessaire pour les images haute définition ou l'affichage en bitmap. Dans ce mode, on ne peut plus non plus afficher 32 sprites. Si vous essayer d'afficher plus de 8 sprites à la fois, ils commenceront à se dupliquer à l'écran !

8.5 Mode texte

Le VDP est en mode texte quand les bits M1=1, M2=0 et M3=0. Dans ce mode l'écran est divisé en 40 blocs horizontaux par 24 blocs verticaux, chacun pouvant contenir une forme de caractère (voir figure 8-7). Chacune de ces formes fait 6 pixels de large sur 8 pixels de haut. Il n'y a que 2 tables requises en VRAM afin de pouvoir réaliser un affichage en mode texte : la table des noms et la table des patrons. La table des couleurs n'est pas requise ici car les informations de couleur sont données par le contenu du registre 7. Les 4 MSB définissent la couleur d'écriture, le s4 LSB la couleur du fond. Ainsi, si le registre R7 contient la valeur F1H, la couleur d'écriture sera le blanc (F) et la couleur de fond le noir (1).

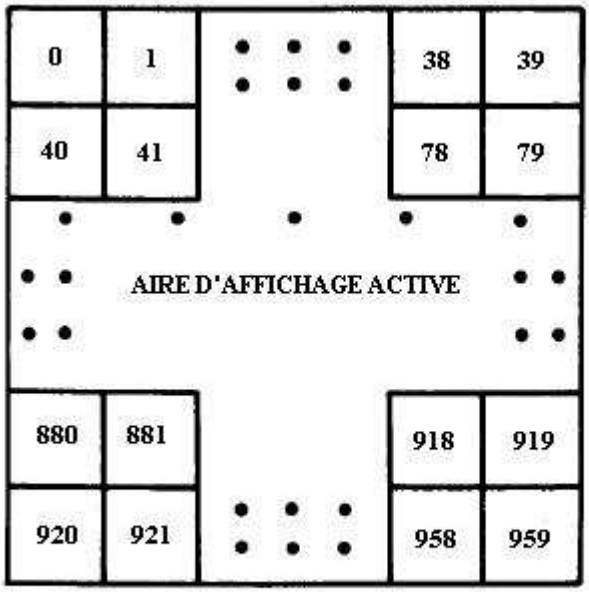


FIGURE 8-7 - POSITIONS DE LA TABLE DES NOMS EN MODE TEXTE

8.5.1 Table des noms

La table des noms en mode texte est très similaire à celle du mode graphique 1, à ceci près qu'elle fait non plus 32x24 mais 40x24 blocs de 8x8 pixels. Ce qui nous donne 960 emplacements d'écran, et donc 960 entrées dans la table des noms.

Chaque entrée dans la table des noms fait un octet et on peut ainsi spécifier un patron parmi 255 (FFH). Si la première entrée de la table des noms est 00H alors le premier patron défini dans la table des patrons sera affiché dans le coin supérieur gauche de l'écran. Si cette entrée contient FFH, ce sera le dernier patron de la table des patrons qui sera affiché dans le coin supérieur gauche.

8.5.2 Table des patrons

La table des patrons fait 2048 (800H) octets de long et est composée de 256 patrons de 8 octets, chacun pouvant représenter un caractère alphanumérique. Comme chaque position d'écran est seulement de six pixels par huit au lieu de huit par huit dans les modes graphiques, le VDP ignore les deux LSB de chaque patron. Ainsi, en mode texte, un patron est défini comme le montre la figure 8-8 , laissant les deux LSB à 0 et définissant notre caractère sur les 6x8 pixels restants.
Afin de laisser un espace entre les caractères à l'écran, la plupart des patrons définis en mode texte n'utiliseront qu'une grille de 5x7. D'autres caractères spéciaux pourront cependant être définis sur toute la grille 6x8.

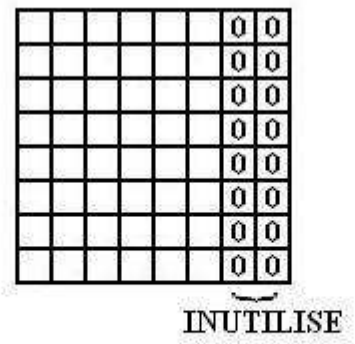


FIGURE 8-8 - GRILLE DE PATRON 6x8 PIXELS POUR LE MODE TEXTE

Jusqu'à 256 patrons peuvent être définis dans la table des patrons, bien que moins d'espace soit requis si tous les 256 patrons ne sont pas utilisés. Par exemple, si votre application ne nécessite que les nombres de 0 à 9 et les majuscules de A à Z, alors seuls les 36 premiers patrons (288 octets) seront nécessaires. Ces 36 patrons seront alors sélectionnés en écrivant dans la table des noms des nombres allant de 0 à 36. Si par exemple la lettre « A » était le premier patron défini, il pourrait être placé dans toutes les positions de l'écran en écrivant 00H dans les 960 entrées de la table des noms.

La figure 8-9 montre le plan d'écran en mode texte.

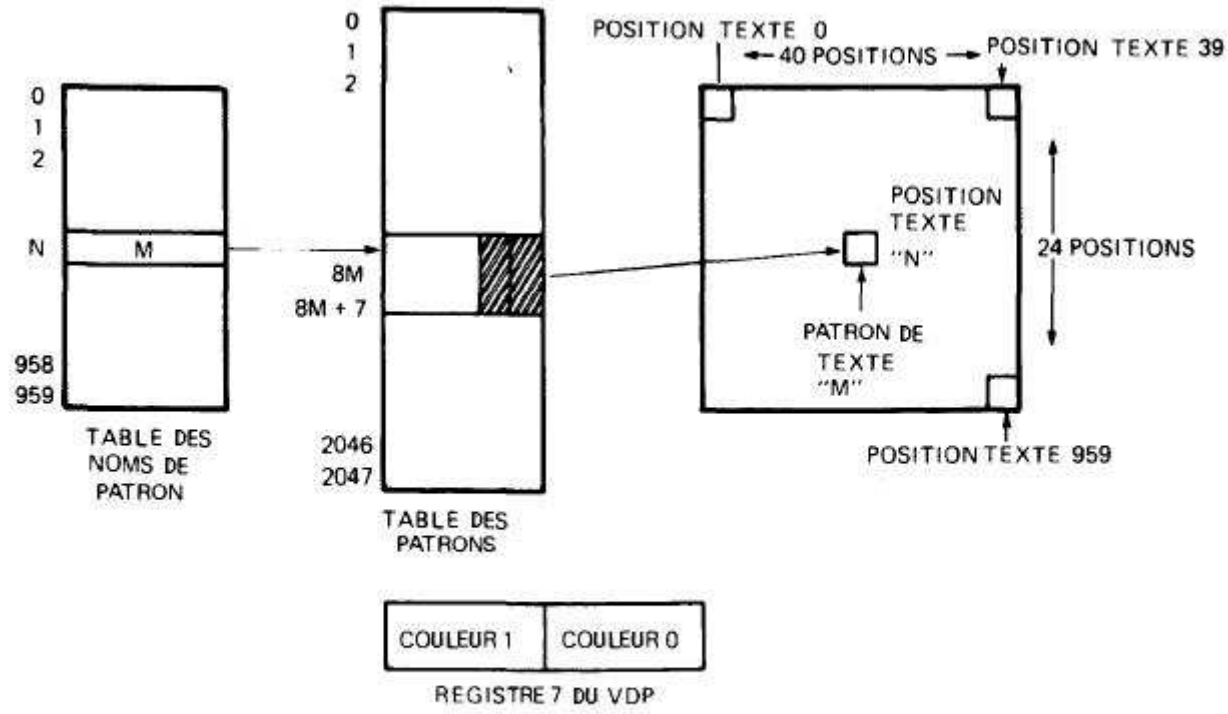


FIGURE 8-9 - PLAN D'ECRAN EN MODE TEXTE

8.6 Mode multicolore

Le VDP est en mode multicolore lorsque les bits de mode M1, M2 et M3 prennent les valeurs suivantes :

M1 = 0

M2 = 1

M3 = 0

Le mode multicolore offre un affichage basse résolution de 64x48 blocs de couleur. Chaque bloc de couleur est équivalent à un groupe de 4x4 pixels qui peut être de l'une des 16 couleurs offertes par le VDP, transparent compris. La couleur de fond et les plans de sprites sont aussi actifs dans ce mode.

NOTE

Le mode multicolore n'est pas supporté par
le VDP avancé de Texas Instruments.

Seules 2 tables sont nécessaires en VRAM pour produire une image en mode multicolore, la table des noms et la table des patrons. La table des noms consiste en 768 entrées, comme dans les autres modes graphiques, bien que la table des noms ne pointe plus vers une liste de couleurs, puisque dans ce mode la couleur est elle-même dérivée de la table des patrons. Le nom pointe vers un segment de 8 octets de VRAM dans la table des patrons.

Seuls 2 octets de ce segment sont utilisés pour définir l'image à l'écran. Ces 2 octets spécifient 4 couleurs, occupant chacune une aire de 4x4 pixels. Les 4 MSB du premier octet définissent la couleur du quartier supérieur gauche, les 4 LSB du premier octet la couleur du quartier supérieur droit, les 4 MSB du deuxième octet la couleur du quartier inférieur gauche, et les 4 LSB du deuxième octet la couleur du quartier inférieur droit. Les deux octets se combinent ainsi pour créer un patron multicolore de 8x8 pixels, comme illustré en figure 8-10.

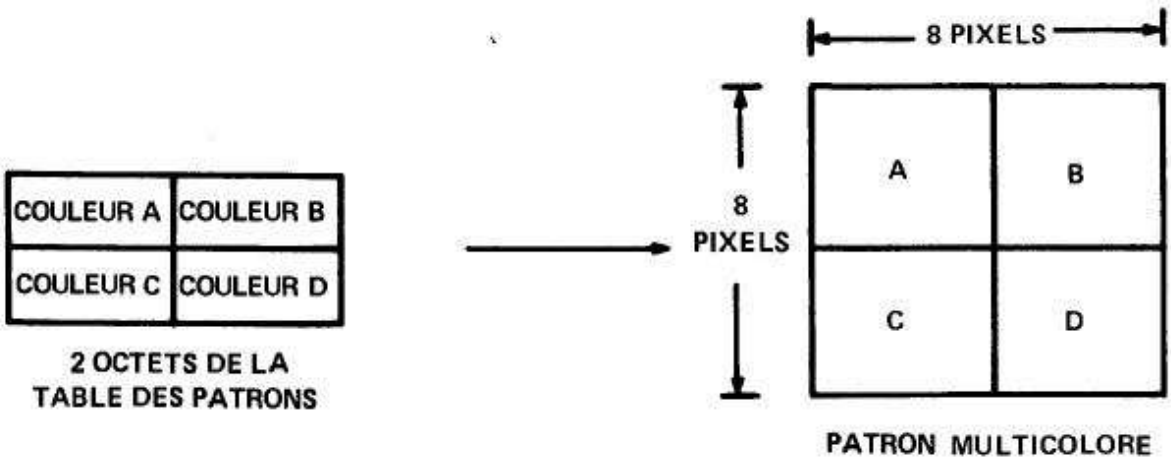


FIGURE 8-10 - PATRON 8x8 MULTICOLORE

L'emplacement des deux octets pointés par la table des noms au sein du segment de 8 octets dépend de la position à l'écran où le nom est implanté. Pour les noms de la ligne supérieure (0-31), les 2 octets sont les deux premiers dans les segments pointés par les noms. La ligne suivante (32-63) utilise les octets 3 et 4, la suivante les 5 et 6 et la suivante les 7 et 8... Pour le reste de l'écran, l'opération est la même.

Voyons à présent un exemple pas à pas pour lever les incertitudes sur le mode de fonctionnement du mode multicolore. La figure 8-11 est composée de la table des noms et de la table des patrons en mode multicolore, ainsi que de la représentation à l'écran. Une autre image d'écran est ajoutée pour dépeindre comment les 767 positions, chacune composée de 4 blocs de 4x4 pixels, remplissent l'écran.

Dans notre exemple (voir fig. 8-11) une entrée de la table des noms pointe sur les emplacements 08H et 09H. Le premier demi-octet en 08H contient la couleur rouge (06H) et le deuxième la couleur bleue (04H). Pour le deuxième octet, c'est respectivement le bleu et le rouge. Ainsi, la position 0 de l'écran contient les 4 couleurs spécifiées. Les calculs pour cet exemple et ceux de la figure 8-11 sont les suivants :

EQUATION POUR TOUVER LES EMPLACEMENTS DANS LA TABLE DES PATRONS

PREMIER OCTET = 2*LIGNE + NOM*8

DEUXIEME OCTET = PREMIER OCTET + 1

LIGNE = MOD4(TRONCATURE(POSITION DU PATRON/32))

CALCULS POUR LES EXEMPLES DE LA FIGURE 8-11

POSITION DU NOM	NUMERO DE PATRON	LIGNE DE LA TABLE DES PATRONS
0	02	2*0 + 02*8 = 10 (octet 1) 10 + 1 = 11 (octet 2)
1	00	2*0 + 00*8 = 00 (octet 1) 0 + 1 = 01 (octet 2)
31	01	2*0 + 01*8 = 08 (octet 1) 08 + 1 = 09 (octet 2)
32	02	2*1 + 02*8 = 12 (octet 1) 12 + 1 = 13 (octet 2)
767	FF	2*3 + FF*8 = 7FE (octet 1) 7FE + 1 = 7FF (octet 2)

COULEUR	CODE HEXA	SYMBOLE
BLEU	04	B
ROUGE	06	R
VERT	0C	G
BLANC	0F	W

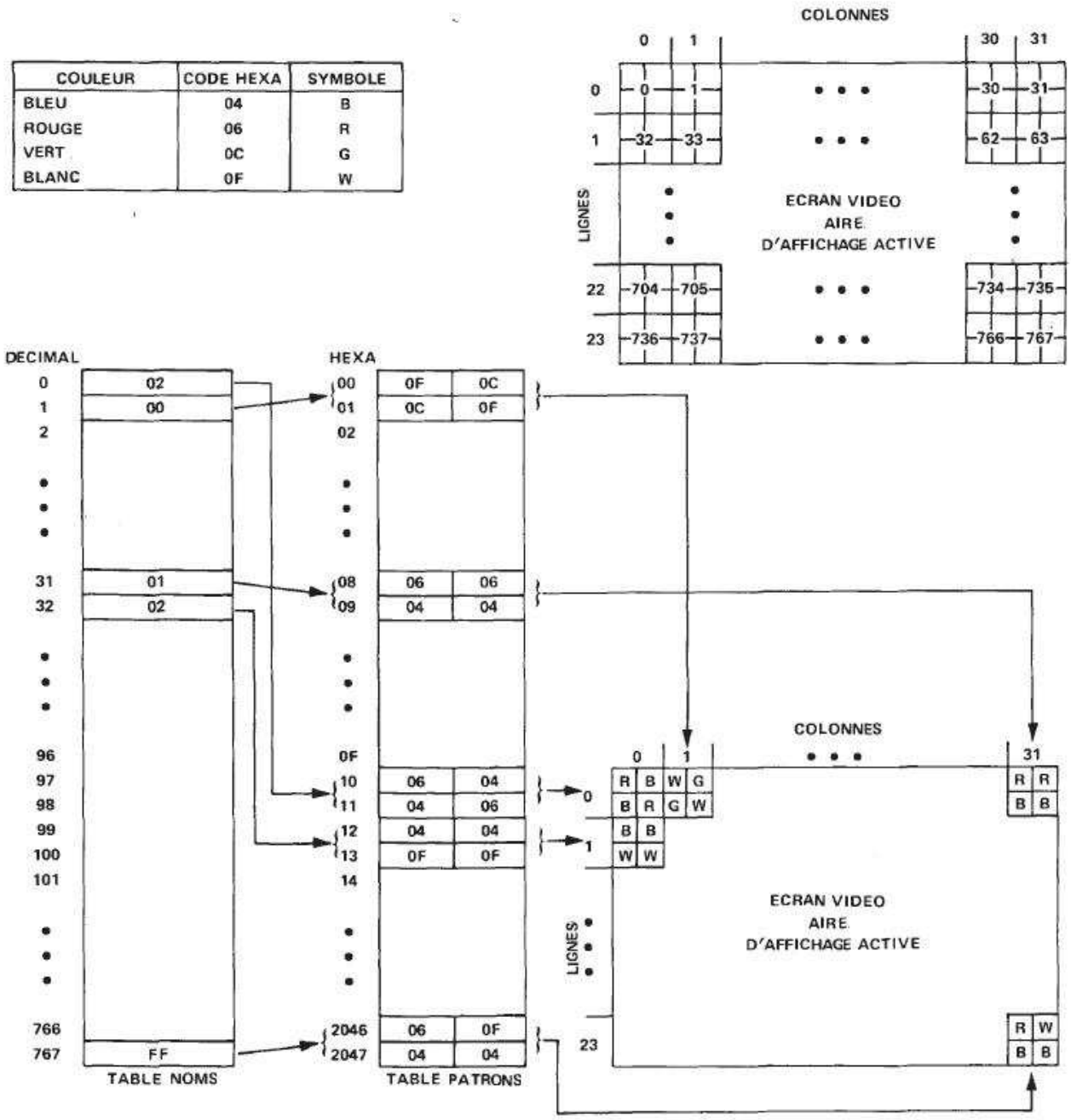


FIGURE 8-11 - SCHEMA D'IMPLANTATION EN MULTICOLORE

L'implantation du contenu de la VRAM à l'image est simplifiée en utilisant des noms dupliqués dans la table des noms, ainsi les séries d'octets utilisés à l'intérieur des segments de huit octets spécifient un patron carré de couleurs de 2x8 pixels à l'écran comme une simple traduction du segment de huit octets en VRAM pointé par le nom commun.
De cette façon, 768 octets sont toujours utilisés pour la table des noms et 1536 octets sont utilisés dans la table des patrons (24 lignes x 32 colonnes x 2 octets/position de patron). Ainsi, un total de 1728 octets sont requis en VRAM. On note que les tables commencent en 1 et 2Ko et ne sont de fait pas contigües.

9 Sprites

Les sprites sont des patrons spéciaux dédiés à l'animation, qui peuvent être déplacés rapidement à l'écran et dont la forme peut changer avec peu d'efforts de programmation. L'affichage vidéo possède 32 plans de sprites Table des patrons de sprite qui contiennent chacun un unique sprite. Ces plans sont numérotés de 0 à 31 (voir chapitre 2-1), le plan 0 étant le plan prioritaire sur les autres plans, le plan 31 étant le plan de plus basse priorité. Quand plus d'un sprite est à la même coordonnée à l'écran, c'est le sprite de plus haute priorité qui est affiché. Il faut aussi noter que les plans de sprites sont également prioritaires vis-à-vis des plans de patron et de couleur de fond.

Ils existent en 2 tailles : 8x8 ou 16x16 pixels. La taille des sprites est déterminée par le bit de taille du registre 1 du VDP. Ce registre contient également le bit d'agrandissement, qui lorsqu'il est mis à 1, permet l'agrandissement au double de sa taille d'un sprite. Ainsi, un sprite 8x8 devient un sprite 16x16 et un sprite 16x16 devient un sprite 32x32. Malheureusement, quand un sprite est agrandi, sa résolution en pâtit, car chaque pixel d'origine est alors modifié en une aire de 2x2 pixels.

Les patrons de sprites sont définis comme des blocs individuels de 8x8 pixels exactement de la même manière que les patrons du mode texte ou des modes graphiques. Un sprite de taille 0 n'a besoin que d'un seul patron en 8x8 pixels. Un sprite de taille 1 est fait de 4 blocs de 8x8 pixels. Les bits éteints dans un sprite sont automatiquement mis en couleur transparente par le VDP, ce qui permet de voir le plan de patrons ou le plan de couleur de fond à travers. Un bon moyen d'imaginer un plan de sprite est de voir une plaque de verre sur laquelle un motif de 8x8 ou 16x16 pixels est collé.

Deux tables sont requises en VRAM pour produire un affichage de sprite : la table des attributs de sprite nous indique les caractéristiques de chaque sprite, comme son emplacement à l'écran, sa couleur et quelle patron lui correspond. La table génératrice des sprites contient quant à elle une bibliothèque de formes à choisir.

Les 32 sprites du VDP peuvent être visualisés simultanément, cependant, un maximum de 4 sprites peut être en même temps sur la même ligne. Si cette règle est outrepassée, les 4 sprites de plus haute priorité seront affichés alors que le cinquième et les suivants seront automatiquement mis en couleur transparente. De plus, l'indicateur de cinquième sprite dans le registre de statut sera mis à 1, et le numéro du sprite fautif est mis dans le registre de statut (voir chapitre 5-2).

Le VDP offre aussi une possibilité limitée de vérification de collision. Si 2 sprites actifs ont des bits se superposant, alors l'indicateur de coïncidence du registre de statut sera mis à 1. Il est à noter que seul le fait de l'existence d'une collision est enregistré, et non les numéros des sprites incriminés. La plupart des applications qui nécessitent de savoir quels sprites entrent en collision scannent perpétuellement la table des attributs de sprite.

9.1 Table génératrice des sprites

La table génératrice des sprites a une taille maximale de 2048 (800H) octets est est localisée dans la VRAM dans des pas de 2 Ko. Son emplacement est donné par l'adresse de base contenue dans le registre 6 du VDP.

Il faut 8 octets pour définir un patron de sprite en 8x8 pixels, et 32 (4x8) octets pour définir un sprite en taille 1. Ainsi, 256 patrons peuvent être définis pour des sprites de taille 0, et 64 pour des sprites de taille 1.

La table génératrice des sprites peut être réduite à 8 octets pour des sprites de taille 0 ou 32 octets pour des sprites de taille 2, car la même forme de sprite peut être réutilisée pour autant de sprites que souhaité. Il n'y a alors qu'à répéter le même numéro de sprite dans la table d'attribut des sprites.

9.2 Table des attributs de sprite

La table des attributs de sprite contient 4 octets d'informations pour chaque sprite affiché. Si les 32 sprites sont affichés, cette table sera donc de 128 octets. L'emplacement de cette table en VRAM est donné par l'adresse de base contenue dans le registre 5 du VDP. Les 4 premiers octets de la table concernent le sprite du plan 0, qui est le sprite de plus haute priorité, les 4 derniers, le sprite du plan 31. La figure 9-1 illustre la relation entre la table des attributs de sprite et les plans de sprite.

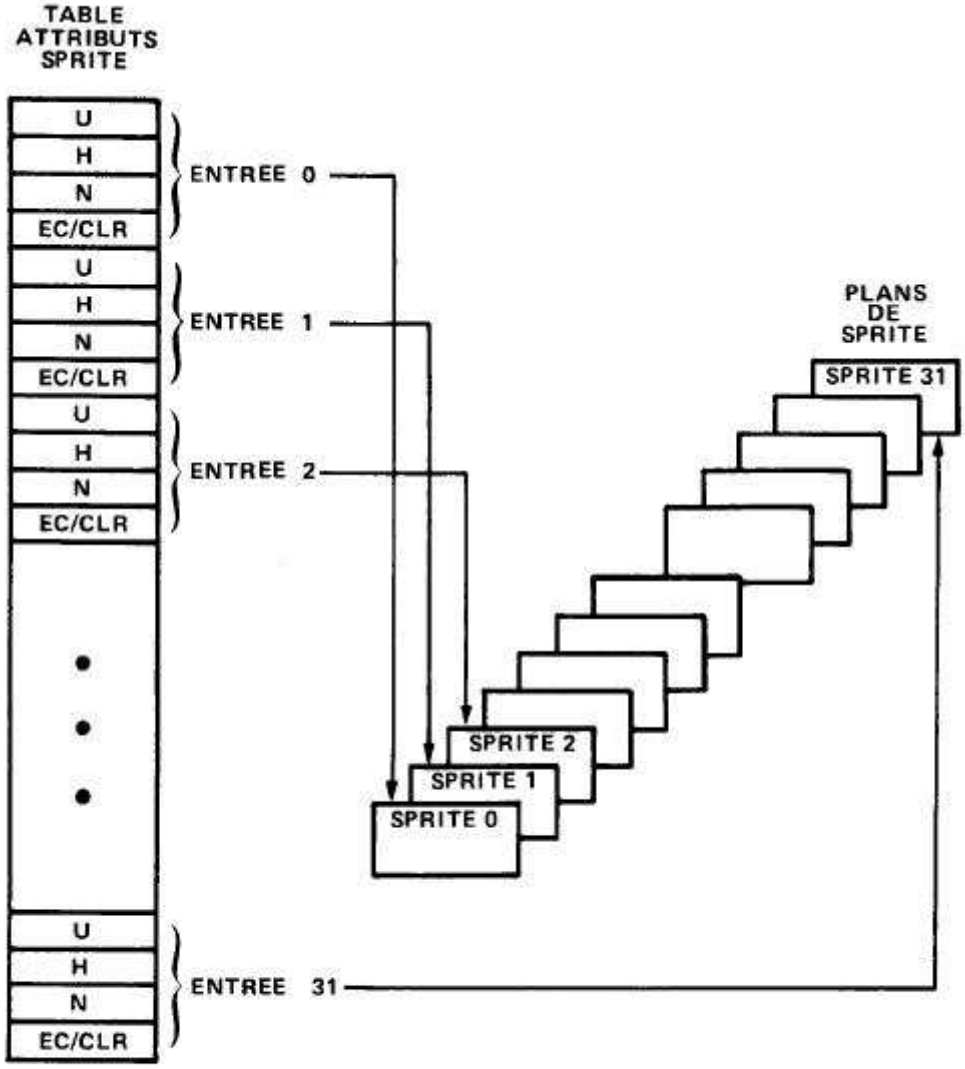


FIGURE 9-1 - RELATION ENTRE TABLE D'ATTRIBUT DE SPRITE ET PLANS DE SPRITE

En se référant à la figure 9-2, examinons à quoi correspondent les quatre octets. Les deux premiers octets déterminent les coordonnées du sprite à l'écran. Le premier octet est la position verticale, et le second la position horizontale. Le troisième octet est le numéro du sprite qui spécifie quel patron de la table génératrice des sprites sera utilisé pour définir sa forme. Le quatrième octet a deux fonctions : les 4 LSB déterminent la couleur du sprite et le bit « early-clock » (MSB) permet de décaler la position horizontale du sprite de 32 pixels. L'utilisation de ce bit permet aux sprites d'apparaître correctement depuis le côté gauche de l'écran. Les trois autres bits sont inutilisés et donc mis à 0.

	BIT NUMERO								
	MSB							LSB	
	0	1	2	3	4	5	6	7	
COORDONNEE VERTICALE	POS. VERT.	POS. VERT.	POS. VERT.	POS. VERT.	POS. VERT.	POS. VERT.	POS. VERT.	POS. VERT.	OCTET 0
COORDONNEE HORIZONTALE	POS. HORIZ.	POS. HORIZ.	POS. HORIZ.	POS. HORIZ.	POS. HORIZ.	POS. HORIZ.	POS. HORIZ.	POS. HORIZ.	OCTET 1
POINTEUR NUMERO DE SPRITE	NUMERO	NUMERO	NUMERO	NUMERO	NUMERO	NUMERO	NUMERO	NUMERO	OCTET 2
COULEUR ET EARLY CLOCK BIT	EARLY CLOCK	0	0	0	COULEUR	COULEUR	COULEUR	COULEUR	OCTET 3

FIGURE 9-2 - ENTREES DE LA TABLE DES ATTRIBUTS DE SPRITE

9.2.1 Position verticale

Le premier octet d'information des attributs est la position verticale du sprite à l'écran. Cette coordonnée détermine la distance à laquelle le sprite va être du bord supérieur de l'écran en pixels. La position du sprite est mesurée relativement au bord supérieur gauche du sprite. Une valeur de -1 (FFH) positionnera le sprite contre le somment de l'écran, et une valeur de 191 (BFH) positionnera le sprite hors de l'écran, tout en bas, comme le montre la figure 9-3. Des valeurs négatives peuvent être utilisées pour permettre l'apparition depuis le sommet de l'écran. Les valeurs comprises entre -32 et -1 permettent l'apparition de tous les types de sprite, même les plus larges (32x32 pixels).

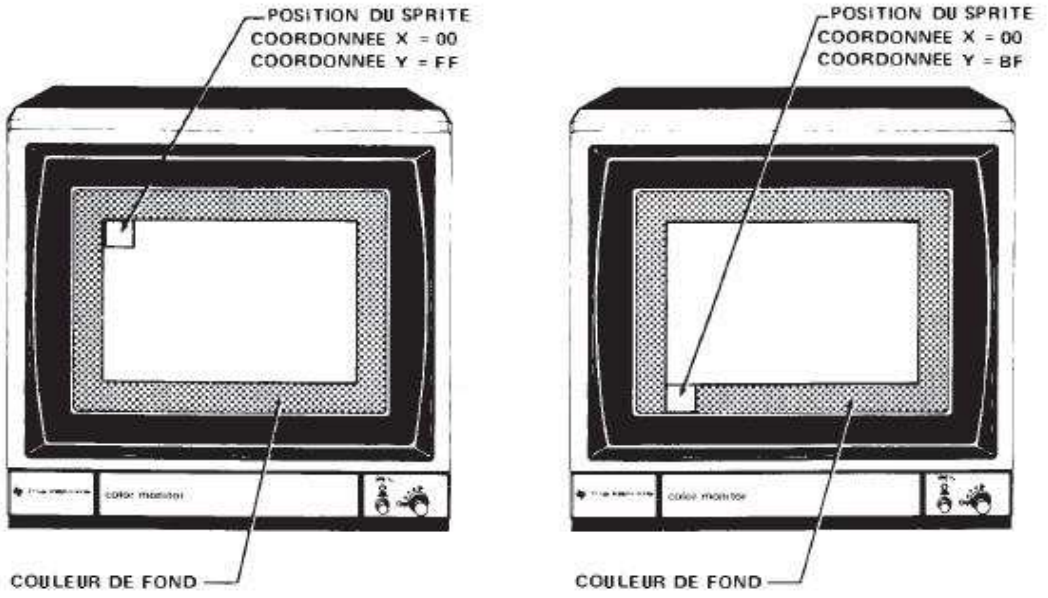


FIGURE 9-3 - POSITIONNEMENT VERTICAL DU SPRITE

Certaines applications ne requièrent aucun ou moins de 32 sprites affichés à l'écran. Une valeur de D0H dans le premier octet de la table des attributs de sprites permettra la fin de l'affichage des sprites. Si aucun sprite n'est utilisé, D0H devra être la première valeur rencontrée dans la table. Si seulement un sprite est utilisé, D0H sera la première valeur dans la deuxième entrée de la table des attributs de sprite. Une fois que le VDP trouve cette valeur de D0H dans une des entrées, tous les sprites de moindre priorité se trouveront désactivés.

9.2.2 Position horizontale

Le deuxième octet d'information dans la table d'attribut de sprite est la coordonnée horizontale. Cette valeur détermine la distance en pixels depuis le côté gauche de l'écran. Une valeur de 0 nous donnera un sprite collé au bord gauche, tandis qu'une valeur de 255 (FFH) fera disparaître totalement le sprite à droite de l'écran, comme illustré sur la figure 9-4. L'utilisation de valeurs avant 255 permettra la disparition vers la droite du sprite.

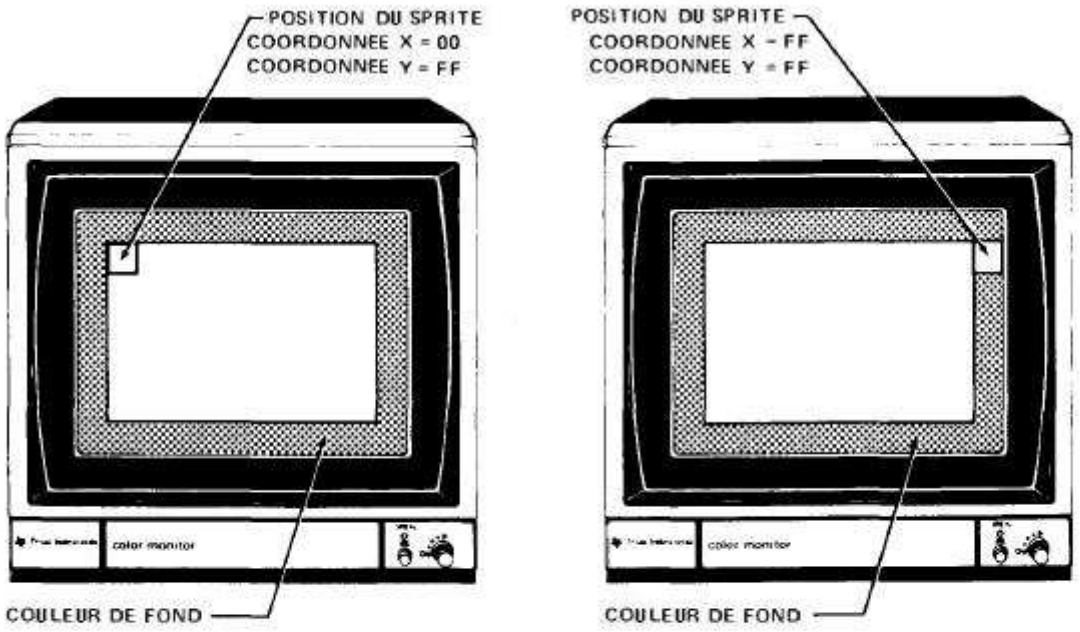


FIGURE 9-4 POSITIONNEMENT HORIZONTAL DU SPRITE

Afin de faire apparaître un sprite depuis le côté gauche de l'écran, un bit spécial appelé « early clock » est utilisé. Ce bit est le MSB du quatrième octet d'une entrée de la table des attributs et sera abordé plus tard.

9.2.3 Numéro de sprite

Le troisième octet d'information contenu dans une entrée de la table d'attributs de sprite est le numéro de sprite. La fonction de cet octet est très similaire à ce que l'on a pour une entrée de la table des noms en mode graphique. La valeur contenue dans cette octet détermine quel patron sera utilisé pour la forme du sprite. Il pointe vers un octet d'information de la table génératrice des sprites où est situé le début de la définition de la forme.

Exemple 9-1

Sprites 8x8 (taille 0)

Une valeur de 00H comme entrée vers la table génératrice des sprites signifie que c'est le premier patron de 8 octets de la table qui sera utilisé pour définir la forme du sprite. La valeur 01H nous donnerait la forme suivante, et ainsi de suite. En continuant jusqu'à 255, on peut ainsi obtenir jusqu'à 256 formes au choix.

Exemple 9-2

Sprites 16x16 (taille 1)

La valeur pointe vers une entrée de 8 octets de la table génératrice des sprites. Comme un sprite 16x16 est constitué de 4 sprites 8x8, les numéros de sprites devraient être 00H, 04H, 08H, 0CH etc. Quand le bit de taille de sprite du registre 1 est mis à 1, le VDP sélectionne non seulement le premier octet de la table génératrice des sprites, mais aussi les 3 suivants.

La possibilité d'avoir des patrons sélectionnables par numéros simplifie grandement la programmation d'animations. Par exemple, si les 4 premiers patrons représentes les différentes étapes d'un personnage qui marche, nous pouvons passer de l'une à l'autre des formes en modifiant simplement le numéro de sprite entre 0 et 3, et en répétant la séquence.

9.2.4 Couleur de sprite et bit « early clock »

Le dernier octet d'entrée de la table d'attributs de sprite sert à 2 fonctions : les 4 LSB contiennent le code couleur, qui peut être n'importe lequel des 16 codes proposés par le VDP. Le MSB est le bit « early clock » qui permet lorsqu'il est mis à 1 de décaler la position horizontale d'un sprite de 32 pixels vers la gauche. Les 3 autres bits ne sont pas utilisés et devront donc être mis à 0.

Le bit « early clock » est utilisé pour permettre des apparitions de sprites par la gauche. Quand ce bit est actif, la position horizontale d'un sprite est décalée de 32 pixels vers la gauche. Considérons un sprite en position 00H, qui est collé au bord gauche de l'écran. Si le bit « early clock » est à 1, même le plus large des sprites (32x32 pixels) serait alors totalement hors de l'écran. Cela permet des valeurs entre 0 et 31 pour faire apparaître le sprite par la gauche. Evidemment, le bit doit être remis à 0 pour pouvoir ensuite faire disparaître le sprite par le bord droit.

Maintenant que toutes les informations sur les sprites ont été vues, observons la figure 9-5 pour voir l'illustration de l'implantation des sprites à l'écran.

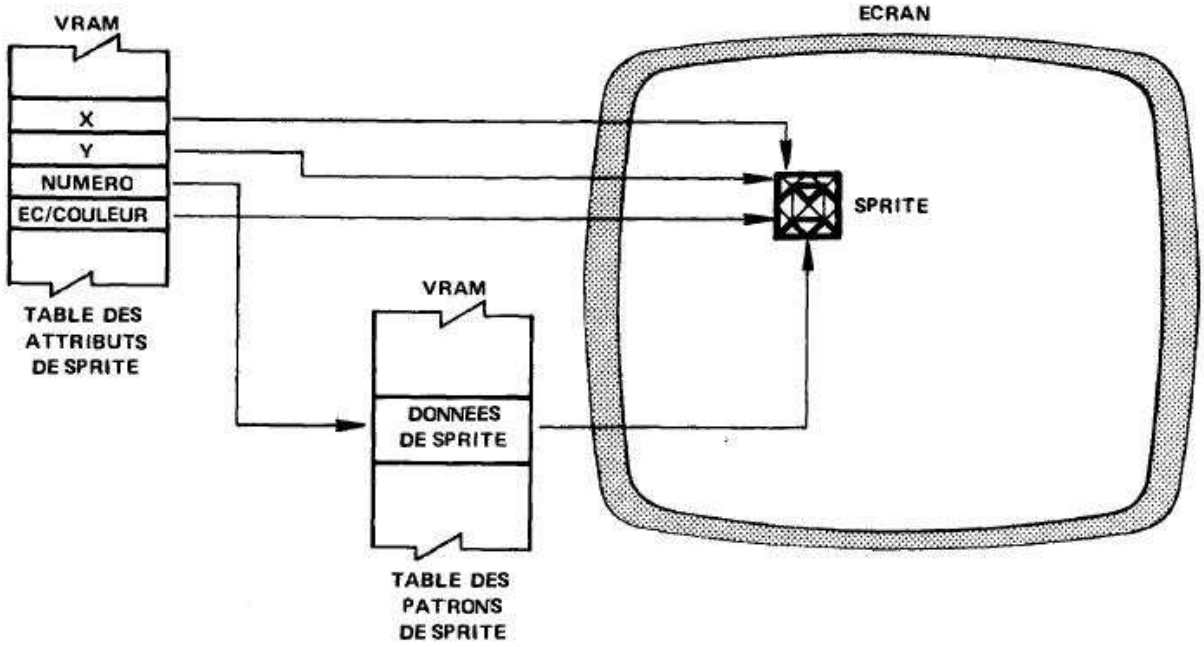


FIGURE 9-5 - IMPLANTATION DES SPRITES

10 Astuces de programmation

10.1 Scrollings horizontaux et verticaux

La façon la plus simple de faire défiler le plan de patrons est de manipuler les valeurs situées dans la table des noms. Le seul inconvénient à cette méthode est que l'écran bougera d'incréments plus grands qu'un pixel. En modes graphiques 1,2 et multicolore, le pas sera de 8 pixels. En mode texte, 6 pixels horizontalement et 8 verticalement. Le mouvement est déterminé par la taille d'un patron.
Un des avantages majeurs de cette méthode est que seul un petit nombre d'octets ont besoin d'être déplacés afin de faire défiler l'écran entier. Dans les modes graphiques 1,2 et multicolore, la table des noms fait 768 octets contre 960 en mode texte. La figure 10-1 montre la séquence pour un défilement d'écran à gauche avec enroulement. La table des noms dans cette illustration possède 768 entrées, et est conçue pour un défilement en modes graphiques 1 ou 2. En y regardant de plus près, on peut voir que les étapse à suivre sont les suivantes :

1) Lire les données situées en colonne 0 en VRAM et les stocker. Les données sont les entrées 0,32,64,96... 736.
2) Lire les données de la colonne 1 et les écrire en colonne 0. Lire les données de la colonne 2 et les écrire dans la colonne 1, et ainsi de suite jusqu'à la colonne 30.
3) Prendre les données stockées de la colonne 0 et les placer en colonne 31. L'écran a ainsi défilé vers la gauche d'une colonne (8 pixels) et s'est enroulé.
4) Répéter cette séquence pour faire défiler continuellement l'écran.

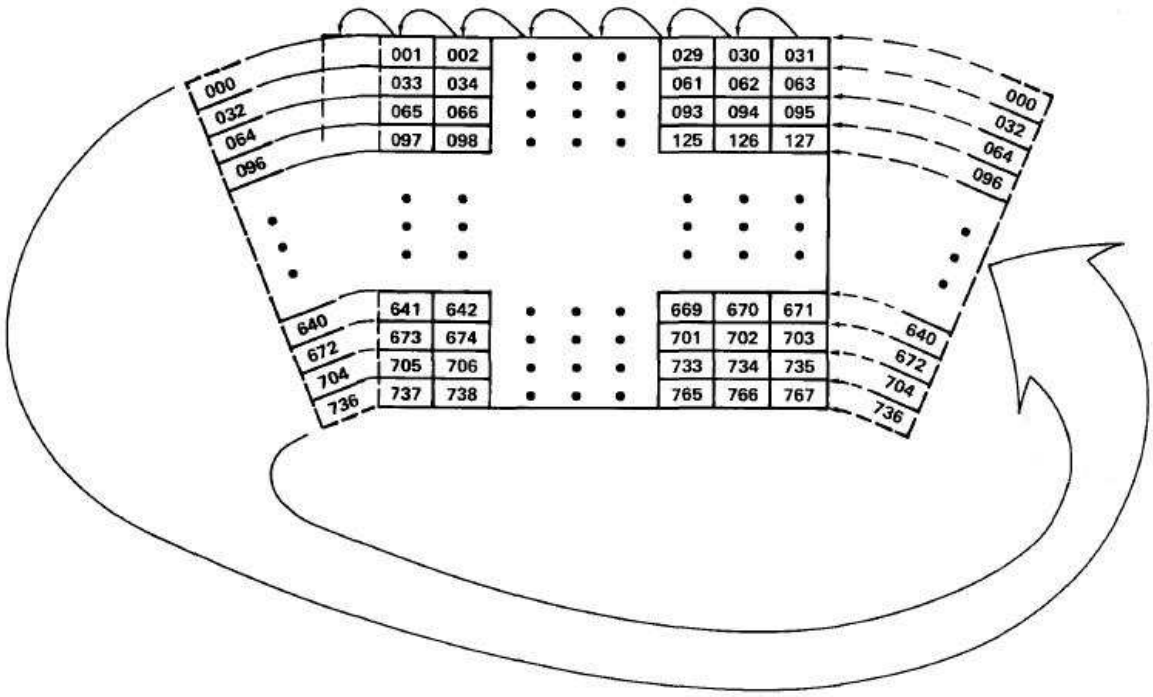


FIGURE 10-1 - DEFILEMENT DE LA TABLE DES NOMS

10.2 Animer des sprites

La procédure pour animer des sprites est assez simple. D'abord, il faut charger des patrons de sprite que vous souhaitez animer dans la table génératrice des sprites en VRAM. Ensuite, chargez les attributs dans la table des attributs de sprite en VRAM. Dans cet exemple, nous verrons comment animer 2 sprites, un d'un personnage qui marche, et l'autre étant une planète qui tourne. L'enchaînement des différentes formes utilisées sont dans les figures 10-2 et 10-3.

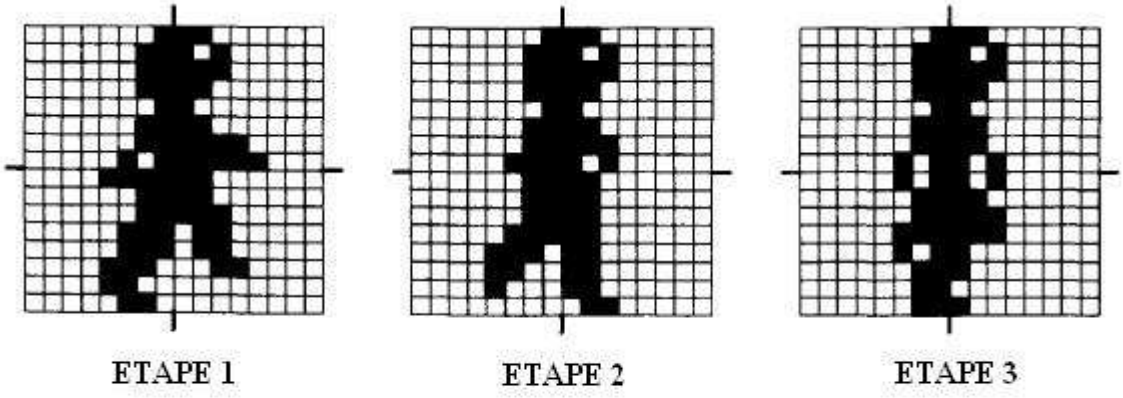


FIGURE 10-2 - PERSONNAGE QUI MARCHE

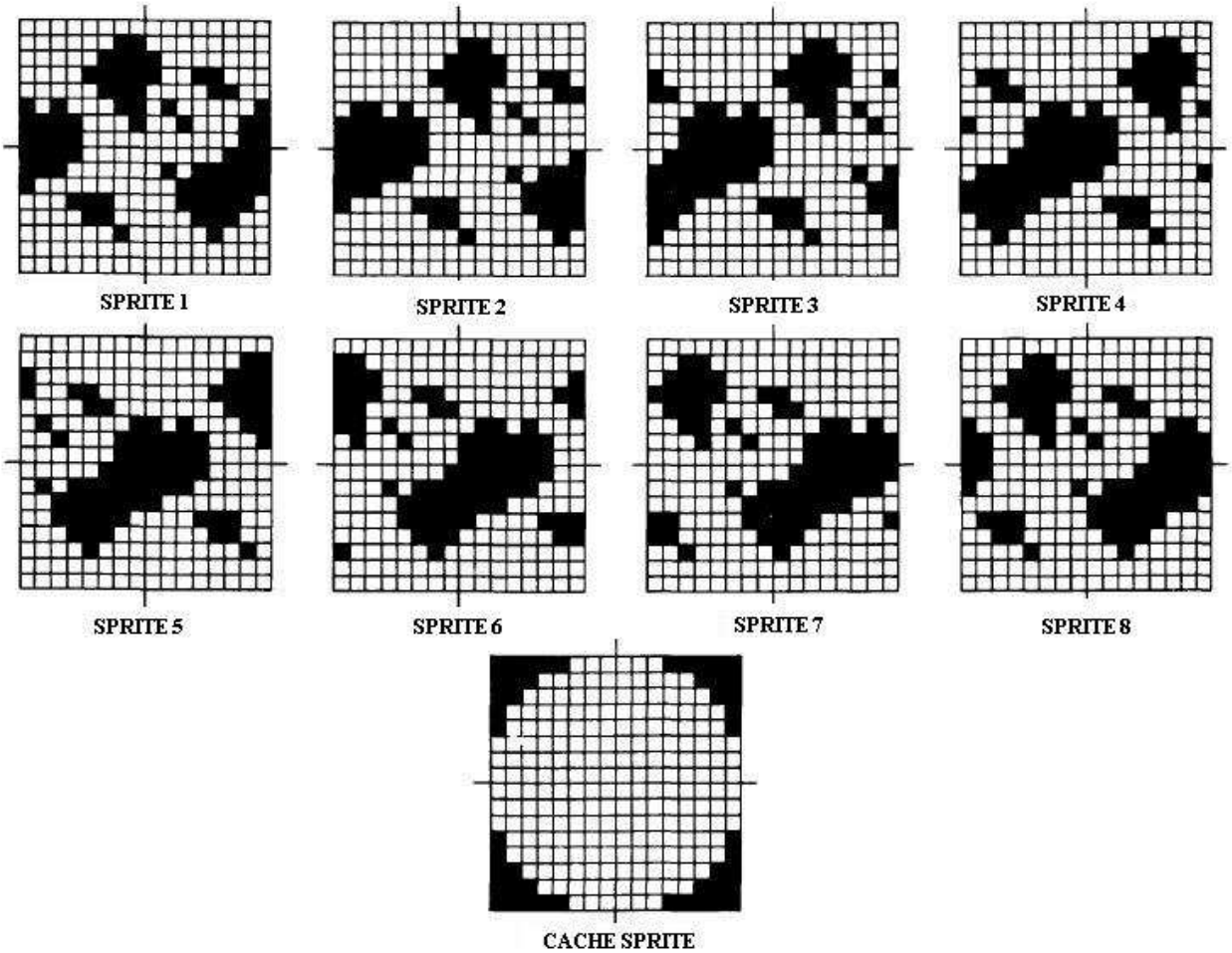


FIGURE 10-3 - PLANETE ANIMEE

En regardant la figure 10-2, on s'aperçoit que le sprite du personnage qui marche est de taille 1 qui passe par 3 étapes d'animation. La planète qui tourne est également un sprite de taille 1 (voir figure 10-3) et passe par 8 étapes d'animation.
Comme les formes de la planète ont été dessinées à plat, de forme carrées, un patron de sprite servant de cache est utilisé en figure 10-4 pour lui donner une forme ronde. La même figure donne le résultat pour les sprites avec ce cache posé par dessus.

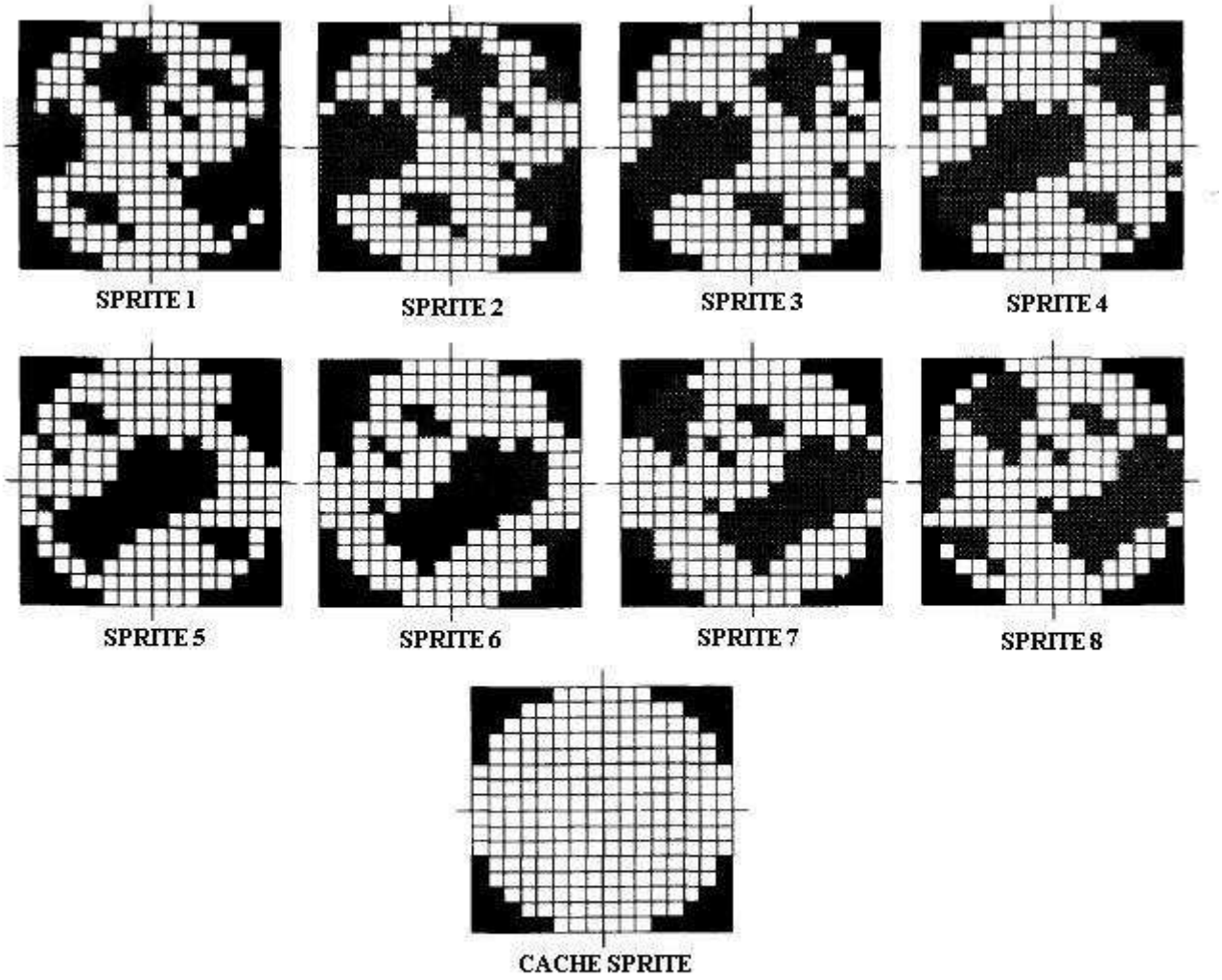


FIGURE 10-4 - PLANETE ANIMEE AVEC CACHE

A présent que toutes les données graphiques des sprites ont été créées, nous pouvons remplir la table des attributs de sprite pour pouvoir les afficher à l'écran. 3 entrées dans cette table seront créées. Les 4 premiers octets définissent la première entrée, celle du sprite de plus haute priorité (sprite 0), ici le personnage qui marche.

Un programme fait pour animer le personnage changerait le numéro d'octet de 00H à 04H à 08H. Ceci déplacerait le sprite à travers les formes définies précédemment. 00H est la valeur initiale car la première étape de la forme du personnage qui marche commence à l'octet 0 de la table génératrice de sprites.

Les deux entrées suivantes de la table des attributs de sprite (sprites 1 et 2) sont utilisés pour définir la planète qui tourne. Le cache est défini comme sprite prioritaire vis à vis de la planète. Cela nous permet de masquer les bits qui dépassent du cache, afin qu'elle paraisse bien ronde.

Le numéro d'octet du cache est 2CH car les données de son patron tombent à cette adresse dans la table génératrice des sprites. Cet octet devrait rester le même dans un programme qui animerait la planète. Si les coordonnées de la planète devaient changer en cours de programme, il en serait de même pour celles du cache.

La troisième entrée de la table des attributs de sprite (sprite 2) est pour les différentes étapes d'animation de la planète. L'affectation de l'adresse 0C pointe vers le patron de même adresse dans a table génératrice des sprites. C'est la première étape de la planète qui tourne. Au cours du programme, il faudrait changer le numéro d'octet pour déplacer dans les différentes étapes d'animation. Les valeurs seraient alors 0CH, 10H, 14H, 18H, 1CH, 20H, 24H et 28H. Après avoir passé ces étapes, il faudrait répéter la séquence depuis le début.

10.3 Collision de sprites

L'indicateur de collision de sprites , situé dans le registre de statut, est mis à 1 chaque fois que 2 sprites ont des points qui se chevauchent. La plupart des applications ont besoin de savoir non seulement s'il y a eu collision, mais quels sprites en particulier sont entrés en collision. Un bon exemple pour cela est la planète qui tourne décrite précédemment. Comme nous avons défini 2 sprites pour créer la forme de la planète, destinés à être l'un sur l'autre à l'écran, l'indicateur de collision serait toujours à 1. Si nous voulions vérifier la collision entre le personnage qui marche et la planète, il serait nécessaire de garder une trace de leurs positions à l'écran dans le programme. Ceci peut être fait en lisant les coordonnées X et Y de chaque entrée de la table des attributs de sprite, et en les comparant ensuite. Dans le cas du personnage et de la planète, si les 2 premiers octets de la première entrée d'attribut étaient les mêmes que les 2 premiers octets de la deuxième entrée, nous saurions que le personnage est situé exactement au sommet de la planète.

August 1984
A Version*
TBL2232-9701
Printed in U.S.A.

TEXAS INSTRUMENTS

SPPU004