**S.A.R.L. PROPULSE**
au Capital de 50.000 Frs
50, RUE BROSSOLETTE
78200 MANTES LA VILLE
R.C.S. Versailles B 339 878 282

## TABLE DES MATIERES

La documentation système comprend :

Introduction
Généralités sur les ordinateurs de la gamme Atari
Le contenu des disquettes de développement
Les shémas hardware
La documentation du hardware
Le manuel de référence du GEMDOS ATARI
Le manuel de référence du BIOS ATARI
La documentation de la ligne A
La documentation de l'interface clavier intelligent
Les spécifications MIDI

Les outils de développement comprennent :

Le manuel du compilateur C
Le manuel de l'assembleur
La documentation des utilitaires
La documentation du RCS

## INTRODUCTION

Le système de développement est celui qui a été utilisé pour la conception même du système d'exploitation de la machine.

Nous vous fournissons toutes les documentations utiles pour le développement dans de bonnes conditions.

Afin de pouvoir développer facilement, nous vous conseillons deux configurations de matériel :

_ un 1040 STFM et un lecteur double face supplémentaire
_ un 1040 STFM et un disque dur 20 Mo

N'oubliez pas de tester vos programmes avec le moniteur couleur pour assurer la compatibilité en mode couleur.

Si vous protégez votre logiciel nous vous rappelons que la norme des lecteurs de disquette est de 0 à 79 pistes, soit 80 pistes utilisables, et que toutes les protections en dessous et au dessus de ces pistes ne fonctionneront pas sur toutes les machines.

Pour obtenir des informations supplémentaires et vous mettre en contact avec d'autres développeurs et le service Technique d'ATARI FRANCE, vous devez vous connecter au réseau CalvaCom RCI en demandant l'accès à la cité ATARI-FRANCE.





## GENERALITES SUR LA MACHINE

Les ordinateurs ATARI 520 STF et 1040 STF bénéficient d'une architecture commune et utilisent le même jeu de composants LSI. Sur le plan technique on peut distinguer deux parties principales.

- Le système central
- Plusieurs systèmes d'entrées/sorties

Le système central est constitué des étages suivants :

- un microprocesseur MC 68000 à 8 MHz
- 192 octets de mémoire ROM
- 1 M octets de mémoire RAM (512 K pour les 520 STF)
- Le système DMA
- Le circuit d'horloge du système
- Le système de gestion des interruptions

Les systèmes d'entrées/sorties sont très complets et permettent de relier le système central à l'environnement externe.
Ils sont au nombre de huit :

_ Le système vidéo :

* procédé Bit map(1), occupant une zone mémoire RAM relogeable de 32000 octets.
* trois résolutions d'affichage
  . basse résolution 320x200 pixels avec 16 couleurs sur une palette de 512 couleurs,
  . moyenne résolution 640x200 pixels avec 4 couleurs sur une palette de 512,
  . haute résolution 640x400 pixels, monochrome.
* interface RVB analogique péritélévision ou monochrome numérique

_ Le système audio :

* 3 convertisseurs N/A programmables
* générateur de bruits

_ Le clavier intelligent :

* entièrement programmable
* géré par un micro-contrôleur 6301
* interfaces souris et manette de jeu intégrées

(1) Bit map : mémoire d'écran telle que chaque pixel correspond à 1 bit en mémoire en mode monochrome ou à plusieurs bits en mode couleur.

_ L'interface parallèle :

* compatible Centronics en sortie
* programmable en entrée
* débit de 4000 octets/s

_ L'interface série RS232C :

* programmable entre 50 et 19200 bauds
* supporte les signaux de contrôle RTS, CTS, DCD, DTR et RI (indicateur de sonnerie).

_ L'interface lecteur de disquette :

* lecteur interne de 720 Koctets (360 K pour le 520 STF)
* possibilité de branchement d'un deuxième lecteur externe
* gérée par un contrôleur WD 1772

_ L'interface disque dur :

* géré par un composant spécifique ATARI
* possibilité des signaux SASI avec une carte d'adaptation
* grand débit de 10 Mbits/s

_ L'interface MIDI

**LE SYSTEME CENTRAL**

Comprend le processeur central, la mémoire centrale (ROM et RAM), le gestionnaire des interruptions et le système DMA:

1. Le processeur central est formé de deux composants clés :

* Le microprocesseur qui est un MC 68000 disposant d'un bus externe 16 bits, d'un bus interne 32 bits et d'un bus d'adresse de 24 bits. Il est cadencé à 8 MHz.

* Un composant spécifique dénomé GLUE qui est, en quelque sorte, le chef d'orchestre du système. Il gère la cohésion de l'ensemble :

  _ en générant les signaux de timing vidéo Blank, DE (Display Enable), Vsync et Hsync,
  _ en générant la priorité des interruptions,
  _ en générant les signaux d'attribution du bus et de sélection des boitiers (chip select),
  _ en recevant les signaux du MFP 68901, du contrôleur de DMA et du MMU afin de synchroniser les transferts de données,





  _ en arbitrant le bus pendant les opérations d'accès mémoire (DMA),
  _ en détectant et signalant les accès illégaux,
  _ en jouant le rôle de chien de garde (absence de DTACK pendant une certaine durée lors de la lecture).

2. La mémoire centrale

Elle consiste en 192 Koctets de mémoire ROM et en un ou deux bancs de 512 Koctets de mémoire RAM dynamiques. Une zone de 128 Koctets de ROM supplémentaires est accessible par l'adjonction d'une cartouche.
Toute la mémoire système est directement adressable.

La mémoire RAM est organisée en mots de 16 bits et les accès peuvent se faire en mode octet (8 bits) ou en mode mot (16 bits)

Les mémoires utilisées sont logées dans des boitiers de 256 Kbits. Dans un 1040 STF il existe 32 boitiers de ce type pour constituer le million d'octets (1 Méga) de la mémoire centrale.
Le plan de la mémoire RAM système est le suivant :

```
000008-000800  Mémoire système (2 Koctets)
000800-07FFFF  Banc numéro 1 (512 Koctets)
080000-0FFFFF  Banc numéro 2 (1040 STF seulement)
```

Le système d'exploitation TOS (The Operating System) se trouve dans les six boitiers de ROM 32 Koctets formant 192 Koctets, ce qui est impressionnant par rapport aux systèmes d'exploitation bas de gammme. Il est cependant possible de charger tout autre système d'exploitation à partir d'une disquette.

Le plan de la ROM est :

```
FA0000-FAFFFF  banc 1 de 64 Koctets de cartouche
FB0000-FBFFFF  banc 2 de 64 Koctets de cartouche
FC0000-FEFFFF  192 Koctets de ROM (TOS)
```

La gestion de la mémoire RAM dynamique du système est effectuée par un composant spécifique Atari dénommé MMU (Memory Map Unit) :

Le MMU prélève sur le bus l'adresse désirée et produit les signaux RAS (Row Adress Strobe) et CAS (Column Address Strobe) destinés aux boitiers RAM dynamiques. Tous les accès à la mémoire RAM système passent par ce composant. Le système peut théoriquement gérer un maximum de 4 Méga octets de RAM. Le rafraichissement des mémoires dynamiques est également effectué par le MMU.

Le chargement du registre de décalage vidéo par les données de la RAM vidéo est également l'une des tâches effectuées par le MMU. Les opérations d'accès direct mémoire (DMA) passent également par ce composant pour accéder à grande vitesse à la mémoire RAM système.

Enfin il faut préciser le rôle du GLUE (voir plus haut) dans l'adressage de la mémoire système. En effet tous les décodages d'adresse sont effectués par son intermédiaire et de ce fait, c'est lui qui adresse à l'attention des boitiers RAM et ROM, les signaux de sélection appropriés.

3. accès direct mémoire

L'accès direct mémoire peut se faire en mode basse vitesse (250 à 500 Kbits/s) ou en mode haute vitesse (jusqu'à 8 Mbits/s). Les données sont échangées sur un bus 8 bits. Le lecteur de disquette transmet les données par l'intermédiaire du contrôleur DMA, placé en mode basse vitesse, alors que le disque dur et les autres périphériques connectés au bus DMA travaillent en mode haute vitesse. Les données échangées passent par le MMU pour arriver à la RAM.

A titre d'exemple voici le protocole d'échange des informations entre le sytème central et le lecteur de disquette. Lorsque le lecteur veut envoyer un octet, il prévient le contrôleur de DMA qui lit cette donnée sur le bus du contrôleur du lecteur (WD 1772) et prévient le composant GLUE. Ce dernier signale la présence de la donnée au composant MMU qui va la placer dans la mémoire RAM du système central, et ainsi de suite pour les autres données.

Lors d'un transfert à haute vitesse, le protocole d'échange est différent. Dans ce cas le contrôleur du DMA dispose d'une mémoire interne de 32 octets qui lui permet de sauvegarder les données échangées tant que le 68000 utilise le bus système. Ainsi on évite les pertes de données et le ralentissement des échanges.

4. Contrôle des interruptions

Le composant MFP 68901 se charge du traitement primaire des interruptions (15 niveaux sur les 16 disponibles sont utilisés). Les interruptions sont masquables au niveau du MFP. Lorsque le 68000 accepte le traitement d'une interruption, le MFP place sur le bus de données un vecteur d'interruption permettant au système central d'entreprendre un traitement approprié. Dans ce cas là le 68000 transforme ce vecteur en une adresse où se trouve la routine d'interruption.
Parmi les traitements utilisant les interruptions nous pouvons citer (par ordre de niveau de priorité décroissant) :





La détection de l'écran monochrome (afin de ne pas endommager un éventuel écran couleur branché en cours de fonctionnement du système), la gestion des signaux de l'interface RS-232C (CTS, DCD et RI), des signaux d'interruption des disquettes et disques durs (FDINT et HDINT), signal BUSY de l'interface parallèle, signal DE (Display enable annonçant le départ de l'affichage d'une ligne écran), les interruptions clavier et données MIDI et les interruptions horloges internes du MFP.

Une telle gestion des événements permet d'utiliser la technologie avancée du processeur central au maximum de sa puissance.

## LE SYSTEME VIDEO

Le systéme vidéo est constitué de la mémoire vidéo, d'un circuit (nommé SHIFTER) comprenant des registres à décalage et des registres de palettes de couleurs, du GLUE et du MMU.

1. la mémoire vidéo

Placée dans la zone mémoire adressable du système, la mémoire vidéo occupe une zone de 32000 octets. Cette zone peut être configurée en 1, 2 ou 4 plans correspondant aux résolutions graphiques haute, moyenne et basse. Les données vidéo sont sous forme de mots de 16 bits. Les informations vidéo sont transmises de la RAM au circuit SHIFTER par le circuit MMU. En mode monochrome chaque bit des mots de 16 bits transmis au SHIFTER correspond à un point ou pixel (picture element) allumé ou éteint; dans ce cas les 32000 octets correspondent à une résolution de 640 x 400 pixels (640 x 400 = 32000 x 8 bits = 256000 points). En mode couleur moyenne résolution la mémoire vidéo est constituée de deux plans. Voici comment le système restitue la bonne couleur. Les données vidéo sont chargées dans le SHIFTER sous forme de 2 mots contigus de 16 bits nommés MOT1 et MOT2. Le bit 1 du MOT1 et le bit 1 du MOT2 donnent la valeur de la couleur du premier point. Le bit 2 du MOT1 et le bit 2 du MOT2 donnent la couleur du deuxième point et ainsi de suite...

Exemple :

bit 1 du MOT1 = 0 et bit 1 du MOT2 = 0
signifient que le premier point a la couleur 0 de la palette.

bit 1 du MOT1 = 0 et bit 1 du MOT2 = 1
signifient que le premier point a la couleur 1 de la palette.

bit 1 du MOT1 = 1 et bit 1 du MOT2 = 0
signifient que le premier point a la couleur 2 de la palette.

bit 1 du MOT1 = 1 et bit 1 du MOT2 = 1
signifient que le premier point a la couleur 3 de la palette.

On constate ainsi la possibilité d'affichage de 4 couleurs (couleur 0 à couleur 3) différentes à l'écran.

Donc dans le cas de la moyenne résolution 1 point nécessite 2 bits d'information pour être codé. Or la zone mémoire vidéo mesure toujours 32000 octets. C'est pourquoi la résolution n'est plus de 640 x 400 mais 640 x 400/2 = 640 x 200.

En mode basse résolution on utilise 4 bits d'information pour pour chaque point affiché à l'écran. Ce qui nous donne la possibilité d'avoir 16 couleurs (2 à la puissance 4) mais une résolution de 640/2 x 400/2 = 320 x 200.

2. Le circuit SHIFTER (registre à décalage)

Il contient 16 registres de palette de couleurs. Les 16 registres sont utilisés en mode basse résolution. En mode moyenne résolution seul 4 registres sont utilisés alors qu'en haute résolution seul le premier bit du premier registre est utilisé pour passer en mode vidéo inversé ou en mode vidéo normal.

Chaque palette est programmée pour 8 niveaux de rouge, 8 niveaux de vert et 8 niveaux de bleu ce qui donne un total de 8 x 8 x 8 = 512 couleurs possibles sans toutefois pouvoir dépasser 16 couleurs affichables. Des programmeurs astucieux peuvent cependant utiliser l'interruption en fin de chaque ligne écran pour changer les 16 couleurs des registres de palette, ainsi il est possible d'afficher une nouvelle ligne avec 16 couleurs différentes. Il est donc possible d'avoir les 512 couleurs sur l'écran mais moyennant quelques astuces (voir l'exemple du logiciel Néochrome qui peut afficher plus de 256 couleurs simultanément).

La sortie couleur du circuit SHIFTER se fait sous forme numérique, et un convertisseur numérique/analogique permet de sortir le signal analogique correspondant pour attaquer l'étage de la sortie vidéo (RVB analogique compatible péritélévision).

En mode monochrome, le signal vidéo numérique est disponible sur une autre sortie du composant SHIFTER. Dans ce cas il est impossible de disposer du signal couleur. En effet le circuit SHIFTER fonctionne à haute fréquence (32 MHz) et produit un signal vidéo haute vitesse (fréquence de ligne de 35.7 KHz et fréquence de trame de 72 Hz). En conséquence, l'utilisateur bénéficie d'une image haute résolution extrêmement stable lui permettant un travail prolongé sans aucune fatigue visuelle.





3. LE GLUE

Est chargé de synchroniser les transferts vidéo entre la mémoire vidéo et le circuit SHIFTER. Il génère par ailleurs les signaux de synchronisation horizontale et verticale destinés à l'écran. Le signal Blanking éliminant l'affichage en dehors de la zone d'affichage d'écran provient également du circuit GLUE.
Enfin le signal DE (Display Enable) annonce au MMU et au SHIFTER qu'une ligne de données vidéo doit être chargée dans le SHIFTER par le MMU pour être visualisée.

4. LE MMU

Deux signaux permettent d'une part, de signaler au SHIFTER que des données vidéo sont prêtes à être lues par ce dernier (signal DCYC) et d'autre part, de valider le mode modification de la palette de couleurs du SHIFTER (signal CMPCS).

## LE SYSTEME AUDIO

Est contitué d'un synthétiseur sonore YM 2149 (compatible AY-3-8910) disposant de trois canaux indépendants mixés en sortie. Il permet de produire des effets sonores intéressants et même de produire des sons de très grande qualité. Ce circuit de constitution assez classique bénéficie d'une très grande maîtrise auprès de certains programmeurs qui savent en tirer un maximum de performances. Ce circuit est cadencé à 2 MHz dans les ordinateurs STF et peut dès lors disposer d'une bande passante en sortie de 30 Hz à 125 KHz. Le son reproduit est disponible sur le haut parleur du moniteur ou du téléviseur.

## LE CLAVIER INTELLIGENT

Pour chaque touche, le processeur du clavier transmet au système central le code correspondant à sa position sur la matrice des touches et son code ASCII. L'autorépétition est gérée par le système du fait de la différence des codes émis à l'appui et au relâchement de chaque touche. Le processeur transmet également les informations émises par la souris et la manette de commande. La communication entre le micro-contrôleur 6301 du clavier et le système central est assurée par un circuit ACIA 6850. Le contrôleur 6301 est cadencé à 1 MHz et dispose d'un programme placé dans une ROM interne et d'une zone RAM. La mémoire ROM interne contient également un programme d'auto diagnostic exécuté au démarrage du système ou lors de l'envoi d'un signal RESET. Les données séries de l'ACIA sont traitées par interruption demandée auprès du MFP et transmises au 68000.

La souris est fournie en standard avec les ordinateurs STF. Il s'agit d'une souris opto-mécanique ayant les caractéristiques suivantes : résolution de 100 points par pouce et vitesse de 25 cm par seconde.

Les deux connecteurs disponibles permettent de brancher une souris ou une manette (port numéro 1) et une manette (port numéro 2). Le port numéro 1 dispose des entrées pour les directions haute, basse, droite et gauche et pour les boutons droit et gauche. Le port numéro 2 dispose des entrées pour les quatres directions précitées et une entrée pour le bouton de la manette.

## INTERFACE PARALLELE

La sortie parallèle Centronics est assurée par le composant YM 2149 qui dispose de deux ports parallèles. La gestion de signaux de contrôle STROBE et BUSY de la norme Centronics est assurée par le MFP et le YM 2149. Le port parallèle peut être configuré en entrée comme en sortie et permet alors d'assurer une liaison bidirectionnelle avec les équipements externes aux ordinateurs de la gamme STF d'ATARI.

Les registres internes sont directement lus par l'unité centrale dés lors que le YM 2149 a été sélectionné par le GLUE.

## INTERFACE SERIE RS-232C

Elle autorise des échanges en mode série asynchrone avec des équipements externes (ordinateur central, autres micro-ordinateurs reliés par modem ou en liaison directe, réseau...) à des vitesses allant de 50 à 19200 bauds. Les signaux de contrôle classiques sont disponibles et sont gérés par le système. Les communications sont gérées par le composant MFP qui outre son rôle de gestionnaire des interruptions, dispose d'un USART (Universal Synchronous /Asynchronous Receiver/Transmitter) permettant d'établir des liaisons en mode série. Il dispose également de quatre timers dont un sert de générateur de bauds. Des tampons de ligne 1488 et 1489 assurent l'adaptation des niveaux TTL aux niveaux requis par la norme RS-232C.

## INTERFACE LECTEUR DE DISQUETTE

Un contrôleur WD 1772 se charge de la gestion de cette interface. Deux lecteurs de disquettes 3"1/2 peuvent être gérés par le système. Les données transmises par le contrôleur passent obligatoirement par le processeur DMA qui en informe le processeur central. Les transferts entre le contrôleur et le processeur central se font par interruption. Le contrôleur WD 1772 dispose de commandes de haut niveau parmi lesquelles nous pouvons citer le formatage complet d'une piste, la lecture et l'écriture d'un secteur, etc...





## INTERFACE DISQUE DUR

L'interface disque dur consiste en un processeur spécifique ATARI qui permet d'assurer des échanges à grande vitesse avec des périphériques externes. Le contrôleur du disque dur est constitué d'une carte externe se trouvant dans l'unité de disque dur elle-même permettant le chaînage de 8 unités. Les commandes transmises au disque dur sont dans un format bloc du type SCSI (Small Computer System Interface). Les transferts DMA sont contrôlés par le périphérique externe qui informe de sa disponibilité par un signal spécial DMA. Les données sont ensuite acheminées vers la RAM par le contrôleur de mémoire (MMU). Le débit de transfert DMA peut atteindre la valeur de 1 Moctets/s.

## INTERFACE MIDI (MUSICAL INSTRUMENT DIGITAL INTERFACE)

Elle offre aux ordinateurs ATARI STF la possibilité de contrôler des synthétiseurs, des séquenceurs, des boites à rythmes. D'autres types d'utilisation sont envisageables, ainsi il est possible de constituer un réseau local d'ordinateurs STF chaînés par l'intermédiaire des entrées/sorties MIDI. L'information MIDI est véhiculée en mode série asynchrone à 31250 bauds sur une boucle de courant. Deux connecteurs DIN 5 broches permettent d'avoir les informations MIDI IN et MIDI OUT (les informations MIDI THRU se trouvent sur les broches non utilisées du MIDI OUT).

Les données en provenance du processeur central sont sérialisées par un ACIA 6850. La réception des données MIDI se fait par interruption.

# CHAPITRE I

# SPECIFICATIONS HARDWARE

Ce chapitre se veut une description globale des caractéristiques techniques de l'ATARI ST, une vue d'ensemble de l'architecture système. Pour une étude détaillée des différents composants, référez vous aux chapitres traitant spécifiquement des différents circuits.

## 1. L'architecture système

L'architecture matérielle de l'ATARI ST (Sixteen/Thirty-two) consiste en un système principal, un sous-système graphique, un sous-système musical et plusieurs sous-systèmes périphériques.

Le ST est structuré autour du microprocesseur MOTOROLA MC68000 (bus de donnée de 16 bits, bus d'adresse de 24 bits), capable d'adresser directement 16 Méga-Octets de mémoire. Les caractéristiques matérielles de l'ordinateur ST se composent du:

Système Principal

* MC68000 cadencé à 8 MHz, 16 bits de données/ 24 bits d'adresse
* 192 Ko de mémoire morte (ROM) extensible par cartouche à 320 Ko
* 512 ou 1024 Ko de mémoire vive (RAM)
* le système DMA (Direct Memory Access)

Sous-système Graphique

* 32 Ko de mémoire vidéo utilisant le procédé Bit-Map
* résolution de 320x200 en 16 couleurs parmi 512
* résolution de 640x200 en 4 couleurs parmi 512
* résolution de 640x400 en monochrome

Sous-système Musical

* générateur de sons programmable
* interface de communication musicale

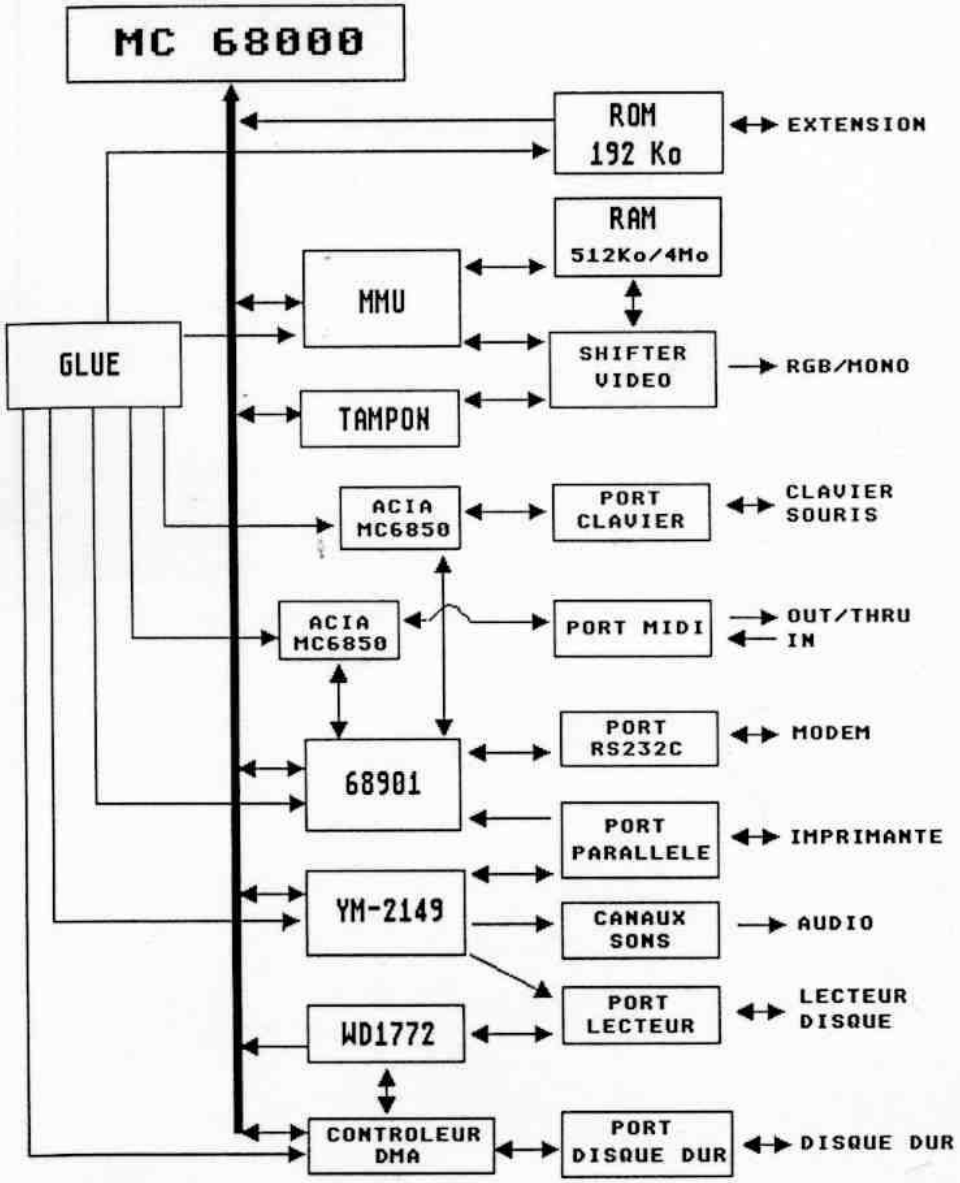
Sous-systèmes Périphériques

* clavier intelligent
* deux boutons de souris
* interface monochrome ou couleur RVB
* interface parallèle
* interface série RS-232
* interface MIDI
* contrôleur de lecteur de disquette et interface DMA
* interface DMA pour disque dur

Le schéma de la page suivante représente la structure matérielle simplifiée de l'ATARI ST.







## 2. Le système central

Le système central comprend le processeur, la mémoire centrale ainsi que le contrôleur DMA. Il doit être considéré comme un système fermé, bien que la vitesse de transfert élevée de ses interfaces périphériques lui assure une ouverture vers l'extérieur.

### 2.1. Le processeur

L'ATARI ST est architecturé autour du micoprocesseur MC68000 de MOTOROLA:
* 16 bits de données,
* 24 bits d'adresse,
* fréquence d'horloge 8 MHz.

Les caractéristiques principales de ce microprocesseur sont les suivantes:
* 8 registres de données de 32 bits,
* 9 regitres d'adresse (dont 2 pointeurs de pile),
* 16 Méga-Octets de mémoire adressable,
* 14 modes d'adressage,
* 5 types de données,
* jeu de 56 instructions.

Le processeur est directement épaulé par le circuit MFP 68901 qui gère, entre autres choses, les interruptions.

### 2.2. La configuration mémoire

La mémoire centrale se compose de cinq bancs de 64 Ko de ROM (trois bancs en configuration standard, deux bancs pour une cartouche éventuelle) et de un ou deux bancs de mémoire RAM dynamique, chaque banc pouvant être constitué de 128 Ko, 512 Ko ou 2 Mo de mémoire.

La reconnaissance de la ROM est assurée par une identification logicielle, celle de la RAM par la programmation du Registre de Configuration Mémoire (lecture/écriture, zéro au reset). La configuration RAM doit être reconnue à l'initialisation de la machine et peut être déterminée en utilisant l'algorithme suivant:

DEBUT — Ecrire 0x000A (2 bancs de 2 Mo) dans le Registre de Configuration Mémoire.

BANC 0 — Ecrire un échantillon de 0x000000 à 0x00001FF
Lire un échantillon de 0x000200 à 0x00003FF
Si égal, alors BANC 0 = 128 Ko; aller à BANC 1.
Lire un échantillon de 0x000400 à 0x0005FF
Si égal, alors BANC 0 = 512 Ko; aller à BANC 1.
Lire un échantillon de 0x000000 à 0x0001FF





Si égal, alors BANC 0 = 2 Mo; aller à BANC 1.
Erreur RAM au BANC 0.

BANC 1 — Ecrire un échantillon de 0x200000 à 0x2001FF
Lire un échantillon de 0x200200 à 0x2003FF
Si égal, alors BANC 1 = 128 Ko; aller à FIN.
Lire un échantillon de 0x200400 à 0x2005FF
Si égal, alors BANC 1 = 512 Ko; aller à FIN.
Lire un échantillon de 0x200000 à 0x2001FF
Si égal, alors BANC 1 = 2 Mo; aller à FIN.
Le BANC 1 n'existe pas.

FIN — Ecrire la configuration dans le Registre de Configuration Mémoire.
Inscrire la taille totale de la mémoire (haut de la RAM) pour un usage futur.

La mémoire RAM est utilisée aussi bien par le microprocesseur que par le contrôleur vidéo. La mémoire d'écran fait donc partie intégrante de la mémoire principale.

### 2.3. Accès direct à la mémoire (DMA)

Le circuit DMA peut travailler en mode basse vitesse (250 à 500 Kbits/s) ou en mode haute vitesse (plus de 12 Mbits/s). Les données transitent par un registre 8 bits.

L'adresse de l'opération lecture/écriture DMA est chargée dans l'Adresse de Base DMA ainsi que dans le Registre Compteur (Lecture/Ecriture, zéro au RESET). Dans la mesure où il n'existe qu'un seul canal DMA, une seule opération est exécutable en même temps.

Une opération DMA utilise un tampon de 32 octets (processus FIFO: First In-First Out) programmable par le Registre de Contrôle de Mode (écriture seulement, non-affecté au RESET) et le Registre de Compte de Secteurs (écriture seulement, zéro au RESET). Le suivi de l'opération (succés, échec) est assuré par le Registre d'Etat DMA (lecture seulement, 1 au RESET). Ce dernier est remis à zéro lors d'une sélection Lecture/Ecriture.

Le micoprocesseur 68000 et le contrôleur DMA ont accès au BUS d'une manière égalitaire, le premier demandeur étant le premier servi. L'accès est garanti pendant la durée complète de l'opération si le demandeur n'en abandonne pas le contrôle.

## 3. Sous-système graphique

Les bases du sous-système graphique se composent du contrôleur vidéo et de la mémoire d'écran. Il ne sera pas traité ici des notions de BitMap, BitBlk, BitBlt, ni des coordonnées système ou des couleurs. Ces notions sont présentées dans la partie logicielle de cette documentation.

## 3.1. La mémoire vidéo

Celle-ci est configurée en plans logiques entrelacés par mots de 16 bits et forme une zone mémoire contiguë de 32 Ko. L'adresse de base de la mémoire vidéo (qui doit obligatoirement être un multiple de 256) est placée dans le Registre d'Adresse Vidéo de Base (lecture/écriture, zéro au reset) ainsi que dans le Registre Compteur d'Adresse Vidéo (lecture seule, zéro au reset). Ce dernier est ensuite incrémenté. Le schéma qui suit montre le diagramme des différentes configurations possibles de la mémoire vidéo:

```
                  16 bits
                 |
4 plans          | plan 0 | plan 1 | plan 2 | plan 3 | plan 0 |...
                 ------------------------------------------------
2 plans          | plan 0 | plan 1 | plan 0 | plan 1 | plan 0 |...
                 ------------------------------------------------
1 plan           | plan 0 | plan 0 | plan 0 | plan 0 | plan 0 |...
                 ------------------------------------------------
```

La mémoire d'écran est considérée comme une partie de la mémoire centrale et, quelque soit le nombre de plans qui la constituent, conserve la même structure: le bit d'ordre 15 du premier mot représente toujours le coin supérieur gauche de l'écran.

```
                  octet fort      | octet faible
                 ----------------------------------
MOT 0            |F|E|D|C|B|A|9|8|7|6|5|4|3|2|1|0|
                 ----------------------------------
```

## 3.2 Configuration Vidéo

Le ST possède trois modes de configuration vidéo:
* résolution 320x200 avec 4 plans (basse résolution)
* résolution 640x200 avec 2 plans (moyenne résolution)
* résolution 640x400 en mono-plan (haute résolution)

Le choix du mode est spécifié au Registre de Décalage du contrôleur vidéo. Une palette de 16 mots, dont seuls les 3 bits faibles des 3 quartets faibles sont significatifs, gère la couleur. Chacun des 16 mots de la palette contient 3 bits pour la couleur rouge (bits A,9,8), 3 bits pour la couleur verte (bits 6,5,4), 3 bits pour la couleur bleu (bits 2,1,0). Huit niveaux d'intensité de rouge, de vert, de bleu produisent 512 couleurs possibles.

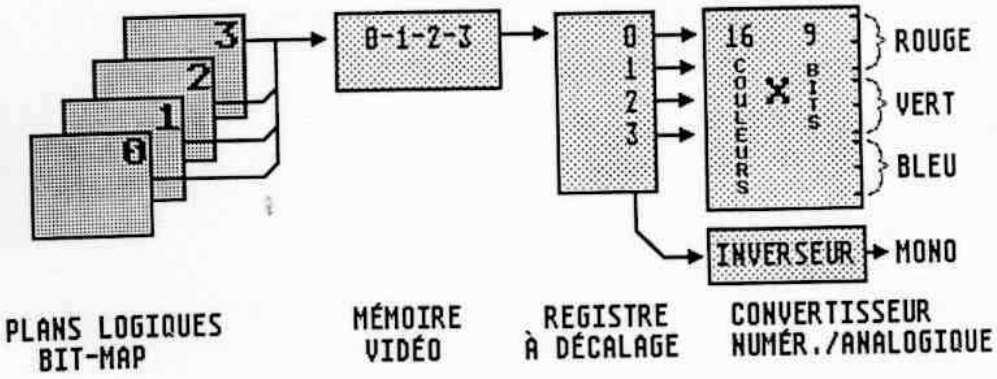
En 320x200 et 4 plans, la palette des 16 couleurs peut être utilisée, tandis qu'en 640x200 et 2 plans, seuls les 4 premiers mots de la palette sont applicables. En 640x400 monochrome, la palette des couleurs n'est pas prise en compte: le signal est envoyé à un inverseur qui ne prend en compte que l'inversion vidéo en testant le bit 0 de la couleur 0 de la palette. Cette couleur est également utilisée en mode multi-plans pour définir la couleur





de la bordure de l'écran. En mode monochrome, cette bordure est toujours noire.

Le processus que suit le contrôleur vidéo est le suivant: en fonction de la résolution, 2 ou 4 mots (chaque mot représentant un plan) sont lus dans la mémoire vidéo et placés dans un registre du contrôleur. Les bits de même ordre (ceux du plan 0 correspondent aux bits faibles) sont ensuite collectés et servent d'index à la palette de couleurs. Les 3 bits de rouge, 3 bits de vert, 3 bits de bleu de la couleur reconnue sont ensuite orientés vers un convertisseur numérique/analogique pour produire les signaux RVB. Le schéma qui suit explique ce processus.




## 4. Sous-système musical

Le sous-système musical est composé d'un générateur de sons programmable et d'une interface série pour instruments de musique (MIDI). Cette interface permet des communications en série à très haute vitesse avec la plupart des périphériques musicaux sophistiqués.

## 4.1. Le générateur de sons

Le générateur de sons YM-2149 peut produire de la musique synthétisée, des effets sonores et des réactions d'écho (écho lors de l'appui d'une touche, écho d'alarme lors d'un cliquage externe à un bouton de sortie). Avec une horloge à 2 MHz, il est capable d'offrir des sons s'étalant sur une fréquence de 30 Hz (audible) à 125 KHz (ultra-sons). Le générateur décharge au maximum le système central et permet de gérer trois canaux indépendants. A la sortie, ces canaux sont mixés, entre eux ainsi qu'avec le signal audio d'entrée, puis envoyés vers le haut-parleur d'une télévision ou d'un moniteur.

Les registres internes des générateurs de sons sont accessibles via le 'Registre Multi-Fonctions' (R7) (écriture seulement, tous les registres à zéro au RESET). Les registres des générateurs sonores contrôlent un signal carré tandis que le registre du générateur de bruit contrôle un signal de fréquence modulable, avec une largeur d'impulsion pseudo-aléatoire.

Les sons et les bruits peuvent être mixés grâce au 'Registre de Contrôle de Mixage', et orientés vers un canal particulier.

Les registres d'amplitude autorisent le choix entre une amplitude fixe ou une amplitude variable. Dans ce dernier cas, un registre d'amplitude est combiné à un registre d'enveloppe.

Les registres de générateur d'enveloppe permettent de gérer les paramètres sonores comme l'attaque, la retenue, l'alternance, la continuïté et offrent ainsi un éventail complet de la gamme chromatique.

## 4.2. Interface de communication musicale

L'Interface Digitale pour Instruments de Musique (MIDI) autorise la connexion du ST avec des synthétiseurs, des séquenceurs, des boîtes à rythmes ou tout autre périphérique possédant l'interface MIDI. Cette ligne de communication série à très grande vitesse (31.25 Kbaud) fournit ou reçoit des informations par l'intermédiaire de deux ports: MIDI OUT (pour les sorties) et MIDI IN (pour les entrées). Le port optionnel MIDI THRU est également présent sur les broches inutilisées de MIDI OUT.

Le Bus MIDI permet de gérer 16 canaux dans l'un des trois modes d'adressage suivants:

* OMNI: les données sont adressées simultanément à tous les récepteurs.
* POLY: chaque récepteur est adressé séparément.
* MONO: chaque voix d'un récepteur est adressée séparément.

L'information est communiquée à travers cinq types de format de données dont la liste est fournie ici dans l'ordre décroissant de priorité:

* RESET système: retour à la configuration par défaut (à utiliser avec précautions afin d'éviter un blocages des recepteurs).
* messages EXCLUSIFS: adressage ciblé selon la marque d'un constructeur (Sequential Circuits, Kawai, Roland, Korg, Yamaha)
* messages HORLOGE: synchronisation
* messages COMMUNS: diffusion d'informations à tout récepteur
* messages CANAUX: sélection de note, données diverses,...

NOTE: Les octets d'état ont leur bit fort mis à un, les octets de données ont leur bit fort mis à 0.





# 5. Les sous-systèmes périphériques

Ce chapitre décrit les sept sous-systèmes périphériques que comprend le ST, c'est à dire:

* un clavier intelligent,
* une interface vidéo,
* une interface parallèle,
* une interface série RS232,
* une interface MIDI,
* une interface lecteur de disquette,
* une interface disque dur.

Chaque description de ces sous-systèmes est suivie d'un schéma de brochage du connecteur. Les broches non-utilisées ne sont pas indiquées.

## 5.1. Le clavier intelligent

Le clavier intelligent d'ATARI transmet les codes d'appui et de relâchement des touches, les mouvements de la souris, les informations des manettes de jeu, ainsi que les signaux d'horloge gérant l'heure et la date.

Le clavier communique avec la processeur maître par l'intermédiaire d'un circuit de communications asynchrones, l'ACIA MC6850, cadencée à 500 Khz. La vitesse de transfert est de 7812.5 bits/s obtenue après division par 64 du compteur de cycles de l'ACIA.

Toutes les fonctions clavier (interrogation des touches, déplacements de la souris, etc...) sont contrôlées par le microprocesseur 8 bits HD6301.

Le clavier intelligent est équipé d'un port mixte souris/manette et d'un port exclusif manette de jeu. La souris à deux boutons d'ATARI est un mécanisme opto-mécanique aux capacités minimales suivantes:

* précision de 100 points par pouce (4 points par mm)
* rapidité de déplacement de 10 pouces par seconde (250mm/s)
* taux d'erreur maximale par impulsion: 50%

La manette de jeu est de type quatre directions plus un bouton de feu.

Le clavier peut traiter les différents signaux des deux ports souris/manette selon trois modes distincts:

* souris ET manette,
* souris désactivée ET manette,
* manette ET manette.

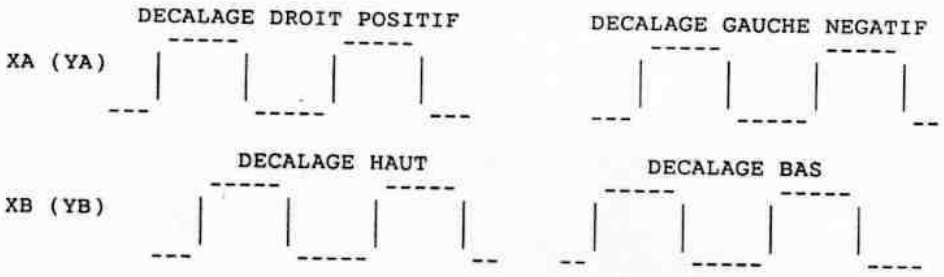
BROCHAGE DU PORT 0: SOURIS ET MANETTE
==> CONNECTEUR MALE 9 BROCHES

| | | |
|---|---|---|
| | ST | |
| MATRICE CLAVIER | 1 | <--- SOURIS XB / MANETTE HAUT |
| MATRICE CLAVIER | 2 | <--- SOURIS XA / MANETTE BAS |
| MATRICE CLAVIER | 3 | <--- SOURIS YA / MANETTE GAUCHE |
| MATRICE CLAVIER | 4 | <--- SOURIS YB / MANETTE DROITE |
| PROCESSEUR CLAVIER | 5 | <--- BOUTON GAUCHE / BOUTON FEU |
| | 7 | ---- ALIMENTATION ----------> |
| | 8 | ---- MASSE |
| PROCESSEUR CLAVIER | 9 | <--- BOUTON DROIT / FEU MANETTE 1 |

CARACTERISTIQUES DES SIGNAUX

| | | |
|---|---|---|
| SOURIS : | broches 1 à 4 | Niveaux TTL |
| MANETTE: | broches 1 à 4 | Niveaux TTL |
| | broche 6 | Niveaux TTL, ramenés à la masse |
| | broche 7 | + 5 volts |
| | broche 9 | Niveaux TTL, ramenés à la masse |

PHASES DE DIRECTION SOURIS




BROCHAGE DU PORT 1: MANETTE
==> CONNECTEUR MALE 9 BROCHES

| | | |
|---|---|---|
| | ST | |
| MATRICE CLAVIER | 1 | <---- MANETTE HAUT |
| MATRICE CLAVIER | 2 | <---- MANETTE BAS |
| MATRICE CLAVIER | 3 | <---- MANETTE GAUCHE |
| MATRICE CLAVIER | 4 | <---- MANETTE DROITE |
| PROCESSEUR CLAVIER | 5 | <---- BOUTON FEU |
| | 7 | ----- ALIMENTATION ----------> |
| | 8 | ----- MASSE |





CARACTERISTIQUES DES SIGNAUX

| | |
|---|---|
| broches 1 à 4, 6 | Niveaux TTL |
| broche 7 | + 5 volts |

## 5.2. L'interface vidéo

L'interface vidéo du ST permet un affichage en basse et moyenne résolution (320x200 et 640x200) sur les récepteurs de télévision, les moniteurs composites ou RVB. Elle permet la haute résolution sur un moniteur monochrome spécifique.

Deux interfaces vidéo sont disponibles: l'une pour le standard PAL (Europe), l'autre pour le standard NTSC (U.S.A.).

Pour les récepteurs américains, le signal du modulateur RF se cale sur deux canaux de diffusion: le canal 2 (55.25 Mhz) et le canal 3 (61.25 Mhz).

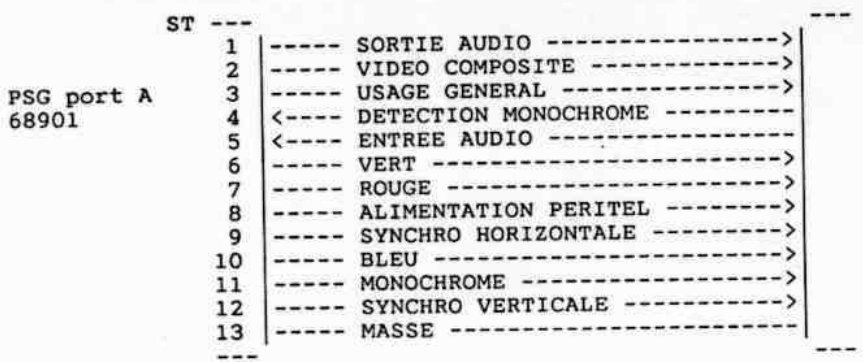
Pour ne pas endommager l'écran par des signaux relatifs à la haute résolution, une ligne de détection de moniteur monochrome est testée en permanence: le circuit MK68901 génère une interruption si la ligne change d'état.

Le choix de la synchronisation (externe ou interne, à 50 ou 60 Hz) est établi par le registre de Mode de Synchronisation (lecture/écriture, zéro au RESET). Le port A du Générateur de Sons Programmable YM-2149 est relié à une broche d'Usage Général qui peut servir par exemple à la mise sous tension, par télécommande, d'un moniteur externe ou encore à une synchronisation interne.

Deux vecteurs d'interruption sont générés pour autoriser la synchronisation logicielle de la zone d'affichage horizontale et verticale (c'est à dire les signaux BLANKING interdisant l'affichage en dehors d'une zone délimitée de l'écran). Le Timer B du 68901 (actif à l'état haut, et en Mode Comptage d'Evénements) gère le signal BLANKING horizontal en provoquant une interruption lorsque son compteur passe à 1. Le compteur du Timer B utilise le signal de la ligne DE (Display Enable ou Autorisation d'Affichage), qui survient à la fin de la première ligne affichée.

Le mode de décalage de l'affichage vidéo doit être sélectionné pendant l'intervalle du BLANKING vertical.

BROCHAGE DU PORT VIDEO ==> PRISE DIN FEMELLE 13 BROCHES




CARACTERISTIQUES DES SIGNAUX

| | |
|---|---|
| broche 1 | 1 Volt de crête à crête, 10 Kohms |
| broche 2 | 1 Volt de crête à crête, 75 ohms |
| broche 3 | Niveaux TTL, (non-assignée) |
| broche 4 | Niveaux TTL, actif à l'état bas, résistance de rappel de 1 Kohm pour +5 Volts |
| broche 5 | 1 Volt de crête à crête, 10 Kohms |
| broches 6-7 | 1 Volt de crête à crête, 75 ohms |
| broche 8 | +12 Volts |
| broche 9 | 5 Volts, actif à l'état bas, 3.3 Kohms |
| broche 10 | 1 Volt de crête à crête, 75 ohms |
| broche 11 | 1 Volt de crête à crête, 75 ohms |
| broche 12 | 5 Volts, actif à l'état bas, 3.3 Kohms |

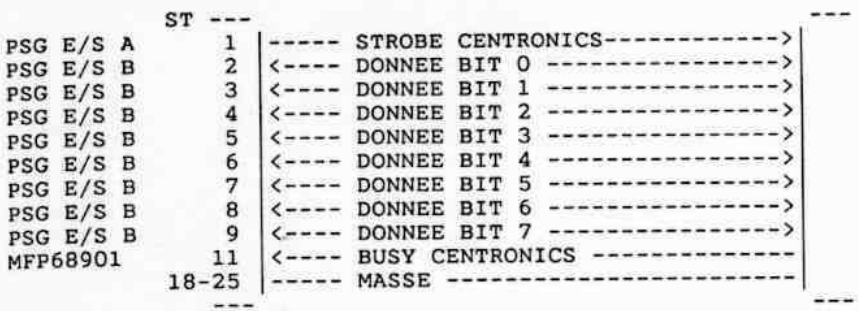
## 5.3. L'interface parallèle

L'interface parallèle de l'ATARI ST gère le signal STROBE ("données valides") à partir du Générateur de Sons YM-2149 (PSG) et le signal Centronics BUSY ("récepteur occupé") à partir du MPF 68901. Le signal ACKNLG ("acquiescement") n'est pas utilisé. Les signaux STROBE et BUSY participent tous deux au protocole dit de "la poignée de main" (handshaking). Les données 8 bits en lecture/écriture sont transmises à travers le port B du PSG à un débit de 4000 octets/seconde.





BROCHAGE DU PORT PARALLELE ==> CONNECTEUR FEMELLE 25 BROCHES




CARACTERISTIQUES DES SIGNAUX

| | |
|---|---|
| broche 1 | Niveaux TTL, actif à l'état bas |
| broches 2-9 | Niveaux TTL |
| broche 11 | Niveaux TTL, actif à l'état haut, résistance de rappel de 1 Kohm pour 5 Volts |

## 5.4. L'interface RS232

L'interface RS232 d'ATARI permet les communications série synchrones et asynchrones. Cinq lignes de contrôle d'asservissement sont traitées:

* Demande d'Emettre (Request To Send)
* Terminal de Données Prêt (Data Terminal Ready)

Ces deux lignes transitent par le port A du PSG.

* Prêt à Emettre (Clear To Send)
* Détection de Porteuse (Data Carrier Detect)
* Indicateur de Sonnerie (Ring Indicator)

Ces trois signaux sont réceptionnés par le circuit MPF 68901.

L'émission/réception des entrées d'horloge de l'USART (un composant du 68901, émetteur/récepteur synchrone-asynchrone) est contrôlée par le Générateur de Bauds du timer D (l'une des quatre horloges internes du 68901). Ce timer est cadencé à 2.4576 Mhz et permet des vitesses de transferts asynchrones comprises entre 50 à 19200 bauds.

L'USART gère un tampon d'un octet et contrôle les erreurs de communications.

BROCHAGE DU PORT RS232 ==> CONNECTEUR MALE 25 BROCHES

```
          ST ---                                        ---
              1 |----- MASSE DE SECURITE ------------|
MFP           2 |----- DONNEE EMISE ---------------->|
MFP           3 |<---- DONNEE RECUE -----------------|
PSG E/S A     4 |----- DEMANDE D'EMETTRE ----------->|
MFP           5 |<---- PRET A EMETTRE ---------------|
              7 |----- MASSE SIGNAL -----------------|
MFP           8 |<---- DETECTION DE PORTEUSE --------|
PSG E/S A    20 |----- TERMINAL DE DONNEES PRET ---->|
MFP          22 |<---- INDICATEUR DE SONNERIE -------|
             ---                                      ---
```

CARACTERISTIQUES DES SIGNAUX

| | |
|---|---|
| broches 2-5 | Niveaux RS232C |
| broches 8,20,22 | Niveaux RS232C |

## 5.5. L'interface MIDI

L'interface MIDI de l'ATARI ST est constituée par un composant ACIA MC6850 chargé de contrôler les communications série asynchrones. L'horloge EMISSION et l'horloge RECEPTION sont toutes deux reliées à un quartz de 500 Khz. La vitesse de transmission de 31.25 Kbauds est générée en définissant le Taux de Division d'Horloge à 16 (via le Registre de Contrôle de l'ACIA).

Les spécifications du protocole MIDI consistent en une donnée de 8 bits précédée d'un bit de Start et suivie d'un bit de Stop.

BROCHAGE DU PORT MIDI OUT/THRU
==> CONNECTEUR DIN FEMELLE 5 BROCHES

```
          ST ---                                        ---
MIDI IN       1 |----- DONNEE TRANSMISE THRU ------->|
              2 |----- MASSE DE SECURITE ------------|
              3 |<---- BOUCLE DE RETOUR THRU --------|
MIDI ACIA     4 |----- DONNEE EMISE OUT ------------->|
              5 |<---- BOUCLE DE RETOUR OUT ---------|
             ---                                      ---
```

BROCHAGE DU PORT MIDI IN
==> CONNECTEUR DIN FEMELLE 5 BROCHES

```
          ST ---                                        ---
MIDI ACIA     4 |<---- DONNEE RECUE IN --------------|
              5 |----- BOUCLE DE RETOUR IN --------->|
             ---                                      ---
```





CARACTERISTIQUES DES SIGNAUX

Boucle de courant 5 mA, actif à l'état bas.

## 5.6. L'interface disque

L'interface lecteur d'ATARI est pilotée, à travers le contrôleur DMA, par le circuit WD1772. Deux lecteurs chaînés en série (disque 0 et 1) peuvent être gérés. Les commandes sont envoyées au Registre de Commandes du WD1772 en orientant au préalable, via le Registre de Contrôle de Mode du DMA, les commandes vers le lecteur de disque. Une séquence de lecture/écriture sur disquette obéit à l'organigramme suivant:

* Sélection du disque 0 ou 1 (A ou B) via le port A du générateur de sons (PSG YM-2149).
* Sélection de la face du disque (0 ou 1) via le port A du PSG.
* Chargement du registre d'Adresse de Base du DMA et du registre Compte de Secteurs du DMA.
* Inversion lecture/écriture pour mettre à zéro le Registre d'Etat (Registre de Contrôle de Mode du DMA).
* Sélection du mode lecture/écriture (Registre de Contrôle de Mode du DMA).
* Sélection du Registre de Compte de Secteurs (Registre de Contrôle de Mode du DMA).
* Chargement du Registre de Compte de Secteurs (déclenchement de l'opération DMA).
* Sélection du Registre de Commandes du circuit disque (Registre de Contrôle de Mode du DMA).

* Emission d'une commande de lecture/écriture (Registre de Contrôle du WD 1772).
* DMA actif jusqu'à ce que le compte de secteurs soit nul (Registre d'Etat du DMA, à ne pas scruter lorsque DMA actif).
* Emission d'une commande d'Interruption Forcée lors d'un transfert de secteurs multiples (sauf en cas de limite de piste atteinte).
* Test des éventuelles erreurs DMA (via le Registre d'Etat DMA).

La détection matérielle d'un changement de disque n'est pas assurée.

L'interface disque dur est également gérée par le contrôleur DMA à travers le contrôleur de disque dur ATARI. Ce dernier utilise le protocole de descripteur de commandes SCSI (Small Computer System Interface).

L'interface disque dur ATARI (ATARI Hard Disk Interface: AHDI) gère partiellement les commandes SCSI (codes opérations de la classe 0), ces dernières étant retranscrites au format ACSI (ATARI Computer System Interface). Le format ACSI correspond à un bloc de commande de 6 octets:

DESCRIPTION D'UN BLOC ACSI

```
OCTET 0   |xxxxxxxx|
           ||||||||
           |||  ---------------- Code opération
            --------------------Numéro du Contrôleur

OCTET 1   |xxxxxxxx|
           ||||||||
           |||  ---------------- Haut adresse du bloc
            ------------------- Numéro de périphérique

OCTET 2   |xxxxxxxx|
           ||||||||
            ------------------- Milieu adresse du bloc

OCTET 3   |xxxxxxxx|
           ||||||||
            ------------------- Bas adresse du bloc

OCTET 4   |xxxxxxxx|
           ||||||||
            ------------------- Nombre de blocs

OCTET 5   |xxxxxxxx|
           ||||||||
            ------------------- Octet de contrôle
```

Voici la liste des codes de commandes valides:

| Code de Commande | Description | |
|---|---|---|
| 0x00 | Test d'unité disponible | |
| 0x05 | Vérification de piste | * |
| 0x06 | Formatage de piste | * |
| 0x08 | Lecture | * |
| 0x0A | Ecriture | * |
| 0x0B | Positionnement de tête | |
| 0x0D | Correction de configuration | |
| 0x15 | Sélection de mode | |
| 0x1A | Mode de lecture | |

*: transfert multiple de secteurs avec recherche automatique

Le processus des commandes émises vers le contrôleur de disque dur s'apparente beaucoup au processus des commandes émises vers le contrôleur de disque. La différence essentielle tient dans la forme du descripteur de commandes. Une séquence de lecture/écriture sur disque dur obéit à l'organigramme suivant:

* Chargement du registre d'Adresse de Base du DMA et du registre Compte de Secteurs du DMA.





* Inversion lecture/écriture pour mettre à zéro le Registre d'Etat (Registre de Contrôle de Mode du DMA).
* Sélection du mode lecture/écriture (Registre de Contrôle de Mode du DMA).
* Sélection du Registre de Compte de Secteurs (Registre de Contrôle de Mode du DMA).
* Chargement du Registre de Compte de Secteurs (déclenchement de l'opération DMA).
* Sélection du registre de Commande Interne du contrôleur (registre de Contrôle de Mode du DMA).
* Termine la sélection du contrôleur en plaçant la ligne A0 à zéro
* Après le dernier octet de commande, sélection du contrôleur (registre de Contrôle de Mode du DMA).
* DMA actif jusqu'à ce que le compte de secteurs soit nul (Registre d'Etat du DMA, à ne pas scruter lorsque DMA actif).
* Test d'éventuelles erreurs DMA (registre d'Etat DMA).
* Test de l'octet d'Etat du contrôleur et, si nécessaire, exécution des corrections qui suivent une commande de Vérification de piste ou de Lecture secteurs.

La taille d'un secteur sur disque ou disque dur est de 512 octets.

BROCHAGE DU PORT LECTEUR DE DISQUE
==> CONNECTEUR DIN FEMELLE 14 BROCHES

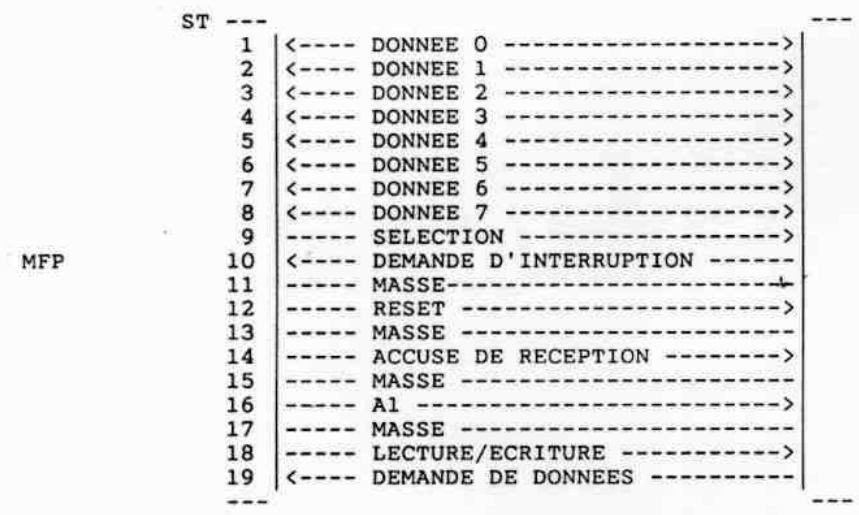
```
                ST ---                                        ---
                    1 |<---- DONNEE LUE --------------------|
PSG E/S A           2 |----- SELECTION FACE 1 ------------->|
                    3 |----- MASSE LOGIQUE -----------------|
                    4 |<---- IMPULSION D'INDEX ------------|
PSG E/S A           5 |----- SELECTION LECTEUR 0 ---------->|
PSG E/S A           6 |----- SELECTION LECTEUR 1 ---------->|
                    7 |----- MASSE LOGIQUE -----------------|
                    8 |----- MISE EN ROUTE MOTEUR --------->|
                    9 |----- DIRECTION --------------------->|
                   10 |----- PAS --------------------------->|
                   11 |----- DONNEE ECRITE ----------------->|
                   12 |----- SENS DU TRANSFERT ------------->|
                   13 |<---- PISTE 00 ----------------------|
                   14 |<---- PROTECTION EN ECRITURE --------|
                      ---                                     ---
```

NOTE: La masse de sécurité ne doit pas être connectée du côté ST; le câble associe par contre les broches DONNEE LUE et DONNEE ECRITE aux masses logiques.

CARACTERISTIQUES DES SIGNAUX

| | |
|---|---|
| broche 1 | Niveaux TTL, actifs bas, resistance de rappel de 1 Kohm pour +5 Volts |
| broche 2 | Niveaux TTL, actifs bas, haut au RESET |
| broche 4 | Niveaux TTL, actifs bas, resistance de rappel de 1 Kohm pour +5 Volts |
| broches 5-6 | Niveaux TTL, actifs bas, haut au RESET |
| broches 8-12 | Niveaux TTL, actifs bas (inversé) |
| broches 13-14 | Niveaux TTL, actifs bas, resistance de rappel de 1 Kohm pour +5 Volts |

BROCHAGE DU PORT DISQUE DUR
==> CONNECTEUR FEMELLE 19 BROCHES




CARACTERISTIQUES DES SIGNAUX

| | |
|---|---|
| broches 1-8 | Niveaux TTL |
| broche 9 | Niveaux TTL, actifs bas |
| broche 10 | Niveaux TTL, actifs bas, resistance de rappel de 1 Kohm pour +5 Volts |
| broche 12 | Niveaux TTL, actifs bas, RESET système |
| broche 14 | Niveaux TTL, actifs bas |
| broches 16,18 | Niveaux TTL |
| broche 19 | Niveaux TTL, actifs bas, resistance de rappel de 1 Kohm pour +5 Volts |





## 6. Composants standards

La configuration standard de l'ATARI ST est constituée des différents éléments matériels suivants:

| | |
|---|---|
| SYSTEME CENTRAL | Microprocesseur MC68000 à 8 Mhz.<br>Périphérique Multi-Fontions MK68901.<br>192 Koctets de ROM.<br>512 ou 1 Moctets de RAM.<br>Circuit de contrôle de mémoire.<br>Circuit de contrôle logique.<br>Circuit de contrôle DMA. |
| SYSTEME GRAPHIQUE | 32 Koctets de mémoire vidéo.<br>Circuit vidéo de registre à décalage. |
| SYSTEME MUSICAL | Générateur de Sons Programmable YM-2149. |
| PERIPHERIQUE | Clavier Intelligent ATARI contrôlé par le microprocesseur HD6301 à 1 Mhz.<br>Deux boutons de souris.<br>Deux ACIA MC-6850.<br>Contrôleur de disque WD1772. |

## 7. Cartouche d'extension

L'ATARI ST permet l'adjonction d'une ROM additionnelle qui peut contenir un maximum de 128 Koctets. La présence de la cartouche, et à fortiori sa taille, ne peut être détectée matériellement. Le brochage de la cartouche ROM est décrit ci-dessous (seules 15 lignes faibles d'adresses sont disponibles).

BROCHAGE DU PORT ROM ==> CONNECTEUR FEMELLE 40 BROCHES

```
ST ---                                              ---
     1 |----- ALIMENTATION +5 V ----------->|
     2 |----- ALIMENTATION +5 V ----------->|
     3 |<---- DONNEE 14 --------------------|
     4 |<---- DONNEE 15 --------------------|
     5 |<---- DONNEE 12 --------------------|
     6 |<---- DONNEE 13 --------------------|
     7 |<---- DONNEE 10 --------------------|
     8 |<---- DONNEE 11 --------------------|
     9 |<---- DONNEE 8 ---------------------|
    10 |<---- DONNEE 9 ---------------------|
    11 |<---- DONNEE 6 ---------------------|
    12 |<---- DONNEE 7 ---------------------|
    13 |<---- DONNEE 4 ---------------------|
    14 |<---- DONNEE 5 ---------------------|
    15 |<---- DONNEE 2 ---------------------|
    16 |<---- DONNEE 3 ---------------------|
    17 |<---- DONNEE 0 ---------------------|
    18 |<---- DONNEE 1 ---------------------|
    19 |----- ADRESSE 13 -------------------->|
    20 |----- ADRESSE 15 -------------------->|
    21 |----- ADRESSE 8 --------------------->|
    22 |----- ADRESSE 14 -------------------->|
    23 |----- ADRESSE 7 --------------------->|
    24 |----- ADRESSE 9 --------------------->|
    25 |----- ADRESSE 6 --------------------->|
    26 |----- ADRESSE 10 -------------------->|
    27 |----- ADRESSE 5 --------------------->|
    28 |----- ADRESSE 12 -------------------->|
    29 |----- ADRESSE 11 -------------------->|
    30 |----- ADRESSE 4 --------------------->|
    31 |----- SELECTION ROM 3 --------------->|
    32 |----- ADRESSE 3 --------------------->|
    33 |----- SELECTION ROM 4 --------------->|
    34 |----- ADRESSE 2 --------------------->|
    35 |----- STROBE DONNEE SUPERIEURE------->|
    36 |----- ADRESSE 1 --------------------->|
    37 |----- STROBE DONNEE INFERIEURE ----->|
 38-40 |----- MASSE ------------------------|
    ---                                             ---
```

## 8. Carte mémoire

Les deux premiers Kilo-octets de la mémoire vive du ST sont réservés à la table des vecteurs d'exceptions et à la pile superviseur. Cette zone ainsi que l'espace mémoire alloué à la carte des entrées/sorties n'est modifiable qu'en mode superviseur. Tenter d'y lire ou écrire en mode utilisateur provoquera une erreur Bus.

Les quatres premiers mots de la RAM sont une copie de la ROM et stockent le pointeur de la pile superviseur ainsi que le





compteur de programme lors du RESET. La liste qui suit donne les adresses principales de la carte mémoire.

CARTE MEMOIRE DU ST

| | | |
|---|---|---|
| 00 0000 | ROM | RESET: Pointeur de Pile Superviseur |
| 00 0004 | ROM | RESET: Compteur de Programme |
| 00 0008 | RAM | Début de la RAM |
| | . | |
| 08 0000 | RAM | 512 Ko de RAM |
| | . | |
| 10 0000 | RAM | 1024 Ko de RAM |
| | . | |
| | . | |
| | . | |
| FA 0000 | ROM | Début ROM si cartouche (320 Ko de ROM) |
| | . | |
| FC 0000 | ROM | RESET: Pointeur de Pile Superviseur |
| FC 0004 | ROM | RESET: Compteur de Programme |
| FC 0008 | ROM | 192 Ko de ROM (débutant en FC0000) |
| FE FFFF | ROM | Fin de la ROM |
| | . | |
| | . | |
| FF 8000 | E/S | Registres de Configuration |
| FF 8200 | E/S | Registres du Contrôleur Vidéo |
| FF 8400 | E/S | Réservé |
| FF 8600 | E/S | Registres DMA et Contrôleur de Disque |
| FF 8800 | E/S | Registres Sons |
| FF 8A00 | E/S | Circuit d'accélération graphique ('Blitter') |
| | . | |
| FF FA00 | E/S | Registres du MPF 68901 |
| FF FC00 | E/S | Registres des ACIAs 6850 |

## 9. Carte mémoire des entrées/sorties

L'espace mémoire réservé pour la carte des entrées/sorties débute en $FF0000 et prend fin en $FFFFFF. Le stockage des registres du MFP 68901 et des ACIAs 6850 commence respectivement en $FFFA00 et $FFFC00. L'accès à ces variables réservées, sans passer par les circuits qui en ont la charge, provoquera une erreur BUS (2 bombes sur l'écran).

NOTES: Pour un bit particulier, la description fournie dans le tableau ci-dessous décrit d'abord la signification lorsque le bit est à 1, puis lorsque le bit est à 0. Exemple:
FF 820A: Si bit 1 mis à 1 = 50 Hz
Si bit 1 mis à 0 = 60 HZ
Les lettres 'L' et 'E' désignent respectivement la possibilité de Lecture et d'Ecriture.
La lettre 'x' indique que l'état du bit n'a pas d'importance.

## CARTE DES ENTREES/SORTIES

### CONFIGURATION

FF 8001 L/E |----xxxx| Configuration Mémoire

Les différentes valeurs des 4 bits faibles ont la signification suivante:

| | BANC 0 | BANC 1 (si utilisé) |
|---|---|---|
| 0000 | 128 Ko | 128 Ko |
| 0001 | 128 Ko | 512 Ko |
| 0010 | 128 Ko | 2 Mo |
| 0011 | Réservé | |
| 0100 | 512 Ko | 128 Ko |
| 0101 | 512 Ko | 512 Ko |
| 0110 | 512 Ko | 2 Mo |
| 0111 | Réservé | |
| 1000 | 2 Mo | 128 Ko |
| 1001 | 2 Mo | 512 Ko |
| 1010 | 2 Mo | 2 Mo |
| 1011 | Réservé | |
| 11xx | Réservés | |

### AFFICHAGE

FF 8201 L/E |xxxxxxxx| Base haute de mémoire vidéo (modulo 65536)

FF 8203 L/E |xxxxxxxx| Base basse de mémoire vidéo (modulo 256)

FF 8205 L/ |xxxxxxxx| Base haute adresse vidéo (modulo 65536)

FF 8207 L/ |xxxxxxxx| Base moyenne adresse vidéo (modulo 256)

FF 8209 L/ |xxxxxxxx| Base basse adresse vidéo (modulo 0)

FF 820A L/E |------xx| Mode de synchronisation

bit 0 Synchro Externe/Interne
bit 1 Balayage 50 Hz/60 Hz

FF 8240 L/E |-----xxx-xxx-xxx| Couleur 0/0 de la palette (bordure)

bit 0 Mono: vidéo inverse/normale
bits 0,1,2 Bleu
bits 4,5,6 Vert
bits 8,9,10 Rouge





| | | | |
|---|---|---|---|
| FF 8242 | L/E | \|-----xxx-xxx-xxx\| | Couleur 1/1 de la palette |
| FF 8244 | L/E | \|-----xxx-xxx-xxx\| | Couleur 2/2 de la palette |
| FF 8246 | L/E | \|-----xxx-xxx-xxx\| | Couleur 3/3 de la palette |
| FF 8248 | L/E | \|-----xxx-xxx-xxx\| | Couleur 4 de la palette |
| FF 824A | L/E | \|-----xxx-xxx-xxx\| | Couleur 5 de la palette |
| FF 824C | L/E | \|-----xxx-xxx-xxx\| | Couleur 6 de la palette |
| FF 824E | L/E | \|-----xxx-xxx-xxx\| | Couleur 7 de la palette |
| FF 8250 | L/E | \|-----xxx-xxx-xxx\| | Couleur 8 de la palette |
| FF 8252 | L/E | \|-----xxx-xxx-xxx\| | Couleur 9 de la palette |
| FF 8254 | L/E | \|-----xxx-xxx-xxx\| | Couleur 10 de la palette |
| FF 8256 | L/E | \|-----xxx-xxx-xxx\| | Couleur 11 de la palette |
| FF 8258 | L/E | \|-----xxx-xxx-xxx\| | Couleur 12 de la palette |
| FF 825A | L/E | \|-----xxx-xxx-xxx\| | Couleur 13 de la palette |
| FF 825C | L/E | \|-----xxx-xxx-xxx\| | Couleur 14 de la palette |
| FF 825E | L/E | \|-----xxx-xxx-xxx\| | Couleur 15 de la palette |

FF8260 L/E |------xx| Mode de décalage

| | |
|---|---|
| 00 | 320x200, 4 plans |
| 01 | 640x200, 2 plans |
| 10 | 640x400, 1 plan |
| 11 | Réservé |

### RESERVE

FF 8400 |----------------| Réservé

### DMA/DISQUE

FF 8600 |----------------| Réservé
FF 8602 |----------------| Réservé

FF 8604 L/E |--------xxxxxxxx| Contrôleur de disque (mot)

FF 8606 L/ |------------xxx| Registre d'Etat DMA (mot)

bit 0 Etat d'erreur.
bit 1 Etat du Compte de Secteurs.
bit 2 Etat Demande de Donnée inactif.

FF 8606 E/ |-------xxxxxxxx-| Contrôle de Mode DMA (mot)

bit 1 A0
bit 2 A1
bit 3 Sélection Disque Dur/Lecteur
bit 4 Sélection Compte de Secteurs
bit 5 Réservé (mis à zéro)
bit 6 DMA inactif/actif
bit 7 Commande Lecteur/Disque Dur
bit 8 Ecriture/Lecture

| | | | |
|---|---|---|---|
| FF 8609 | L/E | \|xxxxxxxx\| | Base haute DMA |
| FF 860B | L/E | \|xxxxxxxx\| | Base moyenne DMA |
| FF 860D | L/E | \|xxxxxxxx\| | Base basse DMA |

REGISTRES SONS

| | | | |
|---|---|---|---|
| FF 8800 | L/ | \|xxxxxxxx\| | Donnée lue du PSG (Port B)<br>Donnée Interface Parallèle |
| FF 8800 | E/ | \|xxxxxxxx\| | Sélection Registre PSG |

4 bits faibles = Numéro du registre

| | |
|---|---|
| 0000 | Registre Réglage fin Canal A |
| 0001 | Registre Réglage fort Canal A |
| 0010 | Registre Réglage fin Canal B |
| 0011 | Registre Réglage fort Canal B |
| 0100 | Registre Réglage fin Canal C |
| 0101 | Registre Réglage fort Canal C |
| 0110 | Contrôle Générateur de bruit |
| 0111 | Contrôle Mixage |
| 1000 | Amplitude Canal A |
| 1001 | Amplitude Canal B |
| 1010 | Amplitude Canal C |
| 1011 | Réglage fin d'enveloppe |
| 1100 | Réglage fort d'enveloppe |
| 1101 | E/S Port A (Sortie seulement) |
| 1111 | E/S Port B |

| | | | |
|---|---|---|---|
| FF 8802 | E/ | \|xxxxxxxx\| | Donnée écrite du PSG |

Si Port A:

| | |
|---|---|
| bit 0 | Sélection Face Lecteur, Face 0/Face 1. |
| bit 1 | Sélection Lecteur 0 |
| bit 2 | Sélection Lecteur 1 |
| bit 3 | Demande à Emettre RS232 |
| bit 4 | Terminal Données Prêt RS232 |
| bit 5 | STROBE Centronics |
| bit 6 | Sortie Usage Général |
| bit 7 | Réservé |

Si Port B:
Donnée Interface Parallèle

MFP 68901

| | | |
|---|---|---|
| FF FA01 | \|xxxxxxxx\| | Registre Général E/S MFP |
| FF FA03 | \|xxxxxxxx\| | Registre Front Actif |
| FF FA05 | \|xxxxxxxx\| | Registre Direction Données |
| FF FA07 | \|xxxxxxxx\| | Reg. Validation Interrup. A |
| FF FA09 | \|xxxxxxxx\| | Reg. Validation Interrup. B |





| | | |
|---|---|---|
| FF FA0B | \|xxxxxxxx\| | Registre Interruption en Attente de Service: A. |
| FF FA0D | \|xxxxxxxx\| | Registre Interruption en Attente de Service: B. |
| FF FA0F | \|xxxxxxxx\| | Registre Interruption en Cours de Service: A. |
| FF FA11 | \|xxxxxxxx\| | Registre Interruption en Cours de Service: B. |
| FF FA13 | \|xxxxxxxx\| | Registre de Masque des Interruptions: A. |
| FF FA15 | \|xxxxxxxx\| | Registre de Masque des Interruptions: B. |
| FF FA17 | \|xxxxxxxx\| | Registre Vecteur. |
| FF FA19 | \|xxxxxxxx\| | Reg. Contrôle Timer A. |
| FF FA1B | \|xxxxxxxx\| | Reg. Contrôle Timer B. |
| FF FA1D | \|xxxxxxxx\| | Reg. Contrôle Timers C et D. |
| FF FA1F | \|xxxxxxxx\| | Registre Donnée Timer A. |
| FF FA21 | \|xxxxxxxx\| | Registre Donnée Timer B. |
| FF FA23 | \|xxxxxxxx\| | Registre Donnée Timer C. |
| FF FA25 | \|xxxxxxxx\| | Registre Donnée Timer D. |
| FF FA27 | \|xxxxxxxx\| | Caractère de Synchronisation. |
| FF FA29 | \|xxxxxxxx\| | Registre de Contrôle USART. |
| FF FA2B | \|xxxxxxxx\| | Registre d'Etat Récepteur. |
| FF FA2D | \|xxxxxxxx\| | Registre d'Etat Emetteur. |
| FF FA2F | \|xxxxxxxx\| | Registre Donnée USART. |

ACIAs MC6850

| | | |
|---|---|---|
| FF FC00 | \|xxxxxxxx\| | Reg. Contrôle ACIA clavier. |
| FF FC02 | \|xxxxxxxx\| | Reg. Données ACIA clavier. |
| FF FC04 | \|xxxxxxxx\| | Reg. Contrôle ACIA MICI. |
| FF FC06 | \|xxxxxxxx\| | Reg. Données ACIA MIDI. |

## 10. Table des Interruptions

La table ci-dessous fournit la liste des interruptions sur le ST ainsi que leur niveau de priorité par ordre décroissant:

VECTEURS D'INTERRUPTIONS DU 68000

| NIVEAU | DESCRIPTION |
|---|---|
| 7 (priorité maximale) | INTERRUPTION NON MASQUABLE<br>MFP MK68901 |
| 6 | |
| 5 | |
| 4 | Synchronisation Verticale |
| 3 | |
| 2 | Synchronisation Horizontale |
| 1 (priorité minimale) | |

CONTROLE D'INTERRUPTIONS DU MFP 68901

| PRIORITE | DESCRIPTION | |
|---|---|---|
| 15 (priorité maxi.) | DETECTION MONITEUR MONOCHROME | I7 |
| 14 | INDICATEUR DE SONNERIE RS232 | I6 |
| 13 | HORLOGE SYSTEME: TIMER A | TA |
| 12 | TAMPON DE RECEPTION RS232 PLEIN | |
| 11 | ERREUR DE RECEPTION RS232 | |
| 10 | TAMPON D'EMISSION RS232 VIDE | |
| 9 | ERREUR A L'EMISSION RS232 | |
| 8 | COMPTEUR DE SYNCHRO. HORIZONTALE | TB |
| 7 | CONTROLEUR DE DISQUE | I5 |
| 6 | ACIAs CLAVIER ET MIDI | I4 |
| 5 | TIMER C | TC |
| 4 | TIMER D: CADENCEMENT RS232 | TD |
| 3 | CONTROLEUR VIDEO: FIN D'OPERATION | I3 |
| 2 | PRET A EMETTRE RS232 | I2 |
| 1 | DETECTION DE PORTEUSE RS232 | I1 |
| 0 (priorité mini.) | LIGNE BUSY CENTRONICS | I0 |

NOTE: En cas d'interruption n°6, le registre d'état des ACIAs MC6850 doit être testé afin de différencier l'interruption MIDI ou CLAVIER.

## 11. Description du boitier

N.d.T.: *La documentation américaine décrivait l' ATARI ST commercialisé dans les pays anglo-saxons. Nous décrirons ici l'ATARI STF tel qu'il est commercialisé en FRANCE.*

L'ATARI ST est constitué d'une unité centrale avec clavier, lecteur et alimentation intégrés.

Le panneau supérieure est constitué des 95 touches du clavier, d'un voyant d'alimentation, d'un voyant de mise en marche du lecteur de disque, et d'une grille de ventilation.

La face arrière comporte l'interrupteur marche-arrêt, le bouton de RESET, la broche d'alimentation, et les connecteurs des interfaces parallèle, série, DMA, vidéo, second lecteur.

La face latérale gauche comporte le connecteur d'extension de ROM et les prises MIDI IN et MIDI OUT.

Dans un logement situé sous la droite du boitier, sont situés les ports souris et manette.





## 12. L'alimentation

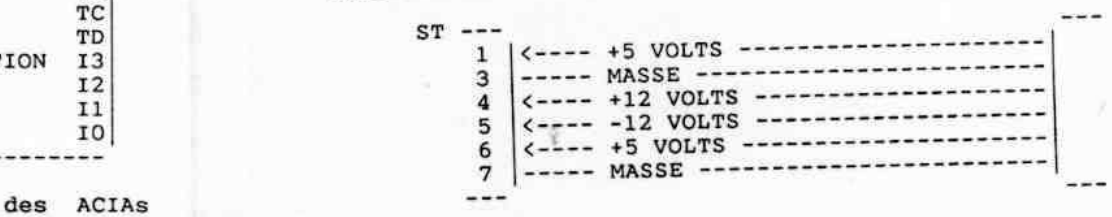
Une alimentation externe (interne et à découpage pour le STF) alimente le système central, la RAM et la ROM d'expansion (ainsi que le premier lecteur sur l'ATARI STF). Une protection contre le sur-voltage est assurée.

EN ENTREE: 220 Volts alternatifs, 50Hz (en France)

EN SORTIE: +5 Volts continus, ± 10%
+12 Volts continus, ± 10%
-12 Volts continus, ± 10%

NOTE: Pour l'ATARI ST uniquement

BROCHAGE DE L'ALIMENTATION ==> CONNECTEUR MALE 7 BROCHES



## ANNEXE A : descriptif du clavier

Le clavier ATARI est constitué de quatre zones ergonomiques distinctes:

* La matrice clavier QWERTY ou AZERTY selon les pays,

* Une zone de dix touches de fonctions,

* Une zone de dix touches de contrôle de curseur,

* Un pavé numérique.

La version américaine du clavier est sensiblement identique au clavier DEC-VT100, avec les exceptions suivantes:

* Touches manquantes:
  [BREAK], [LINE FEED], [NO SCROLL], [SET UP]

* Touches déplacées:
  [CAPS LOCK], touches de contrôle du curseur, touches de fonctions.

* Touches supplémentaires:
  [ALTERNATE], [HELP], [UNDO], [INSERT], [CLEAR/HOME], ainsi que les dix touches de fonctions.

* Perfectionnement:
  Sur le DEC VT-200, le 'T' formé par les touches de déplacement du curseur est inversé; les touches opératoires du pavé numérique sont positionnées différemment.

Sur le clavier américain, la touche [SHIFT] de gauche est plus large afin de permettre l'adjonction d'une touche supplémentaire (Cf. ISO 2530-1975). Sur le clavier français, la touche '<' et '>' a été rajoutée.

Différentes versions du clavier ATARI ont été adaptées pour les pays suivants:

* Royaume-Uni
* Etats Unis d'Amérique
* Finlande
* Norvége / Danemark
* Suéde
* Japon
* Allemagne
* Canada Français
* France
* Italie
* Espagne





## ANNEXE B: Bibliographie

Propos généraux:

* A Hitchiker's Guide to the BIOS.
* Digital Research GEM Software Documentation.

Système Central:

* MOTOROLA MC68000 16-Bit Microprocessor User's Manuel, 4éme édition.
* Mostek MK68901 Multi Function Peripheral Data Sheet.

Sous-Système graphique:

* Adele Goldberg and David Robson, 'Smalltalk-80: The Language and Its Implementation', Addison-Weslay, Reading Massachusetts, 1983, Chapitre 18.

Sous-système musical:

* General Instrument AY-3-8910 Programmable Sound Generator Data Sheet.
* MIDI Musical Instrument Digital Interface Specification 1.0.

Sous-systèmes périphériques:

* Atari Intelligent Keyboard (ikbd) Protocol and Specification.
* Motorola MC6850 Asynchronous Communications Interface Adapter Data Sheet.
* Centronics Parallel Interface Specification.
* Electronic Industries Association RS232C Standard.
* Western Digital WD1770/1772 Floppy Disk Controller Data Sheet.
* Specification of the Atari Computer System Interface (ACSI).
* Specification of the Atari Hard Disk Interface (AHDI).

## ANNEXE C: Notes

* Une 'erreur adresse' survient lorsqu'un mot est lu à une adresse impaire.

Système central

* L'adresse de Base DMA et le Registre Compteur doivent être chargés en suivant cet ordre: bas, moyen, haut.

Sous-système musical

* L'espace mémoire et les registres du PSG YM-2149 doivent être utilisés avec discernement.

Sous-systèmes périphériques

* Le contrôle du processus engagé par le Contrôleur de Disque se fait par l'intermédiaire du MPF 68901. Ne testez pas la ligne BUSY du Contrôleur ou le Registre Compte de Secteurs du DMA.

* Sélectionnez le Registre Compte de Secteurs avant de tester l'octet d'état d'erreur du DMA.

* Pour le WD1772, ne positionnez pas un délai de 30 ms pour l'exécution des commandes de type 2 et 3.

* Une interruption forcée doit être générée après toutes commandes envoyées au WD1772.

* Attendez que le bit 'MOTOR ON' (moteur en marche) soit à l'état bas avant de désélectionner un disque.

* Une table de configuration des lecteurs de disque doit être gérée par logiciel afin de permettre l'adjonction de différents types de lecteurs 3 pouces et demi. Les deux lecteurs les plus couramment employés ont les caractéristiques suivantes:
  - 500 Ko non-formaté, 80 pistes, une tête, 3 ms de pas.
  - 1 Mo non-formaté, 80 pistes, deux têtes, 3 ms de pas.





# I N T E R F A C E  A C S I

(Atari Computer System Interface)

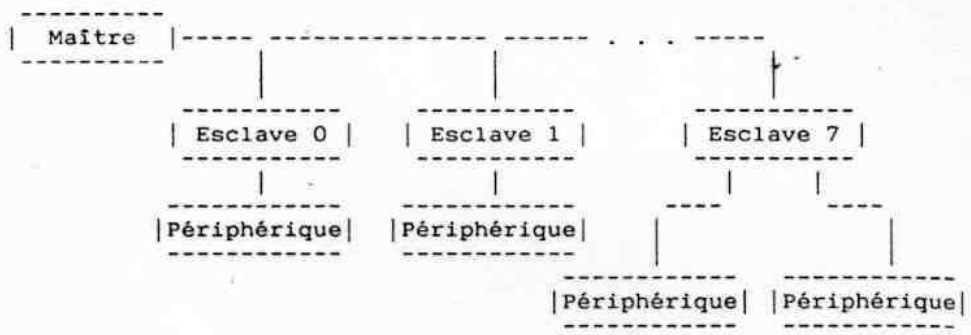


NOTE : L'ensemble de ce chapitre est uniquement constitué de notes brutes sur l'interface ACSI. Il s'agit d'un document préalable, susceptible d'être révisé sans préavis.

## 1. Le bus ACSI

* signaux de contrôle et bus bidirectionnel.
* un esclave ne doit pas recevoir une commande et la laisser en attente alors que le contrôleur est prêt -- une erreur PERIPHERIQUE NON PRET doit immédiatement être envoyée ou bien le maître doit temporiser puis considérer le contrôleur comme inexistant.
* auto-test du contrôleur -- ré-étalonnage, test des mémoires vives, somme de contrôle des mémoires mortes, etc.
* auto-test toujours effectué immédiatement après le reset -- ce qui dispense de la nécessité d'une commande d'auto-test.
* le maître doit temporiser (durée à déterminer) sur une commande et "resetter" l'esclave.
* une fois retourné l'octet d'état, le bus est libre.
* au maximum huit ports peuvent se connecter au bus.
* vitesse de transfert des données maximale : 8 Mbit/sec.

-------- Synopsis du bus ACSI ----------








--------- Signaux de contrôle et de données ----------

| Mnémonique | Nom | Caractéristiques |
|---|---|---|
| RST | Reset | niveaux TTL, actifs bas. |
| A1 | Adresse 1 | niveaux TTL. |
| IRQ | Demande d'interruption | niveaux TTL, actifs bas, résistance de 1Kohm vers 5 volts du côté maître. |
| CS | Sélection | niveaux TTL, actifs bas. |
| R/ W | Lecture/Ecriture | niveaux TTL. |
| DRQ | Demande de Données | niveaux TTL, actifs bas, résistance de 1Kohm vers 5 volts du côté maître. |
| ACK | Accusé de réception | niveaux TTL, actifs bas. |
| DATA | Bus de données (0-7) | niveaux TTL. |

------- Schéma de brochage du port ACSI côté maître -------------

| MAITRE | DB 19S | | ESCLAVE |
|---|---|---|---|
| | 1 | <--- Donnée 0 ----------> | |
| | 2 | <--- Donnée 1 ----------> | |
| | 3 | <--- Donnée 2 ----------> | |
| | 4 | <--- Donnée 3 ----------> | |
| | 5 | <--- Donnée 4 ----------> | |
| | 6 | <--- Donnée 5 ----------> | |
| | 7 | <--- Donnée 6 ----------> | |
| | 8 | <--- Donnée 7 ----------> | |
| | 9 | ---- Sélection ----------> | |
| | 10 | <--- Demande d'interruption ---------- | |
| | 11 | ---- Masse ---------- | |
| | 12 | ---- Reset ----------> | |
| | 13 | ---- Masse ---------- | |
| | 14 | ---- Accusé de réception ----------> | |
| | 15 | ---- Masse ---------- | |
| | 16 | ---- A1 ----------> | |
| | 17 | ---- Masse ---------- | |
| | 18 | ---- Lecture/Ecriture ----------> | |
| | 19 | <--- Demande de données ---------- | |

## 2. Normes ACSI

### 2.1. Niveau 1

* l'esclave ne parle que lorsqu'on lui parle.

* scrutation permanente du bus -- pas de déconnexion.

* abandon du maître via une interruption.

* abandon de l'esclave via un reset -- un reset logiciel doit être ordonné par le maître.

* la DUREE du RESET est de 12 microsecondes.

* le RESET a la priorité la plus forte sur le bus.

* le RESET ne peut être ordonné par l'esclave qu'il soit actif ou inactif.

* le maître doit temporiser durant 100 millisecondes dans l'attente d'un accusé de réception de l'esclave.

* ATTENTION : Si un maître envoie prématurément une commande, c'est-à-dire alors que l'esclave en exécute une, les résultats sont imprévisibles.

* Le gestionnaire de périphérique du maître doit attendre jusqu'à ce que l'octet d'état lui soit renvoyé --- donc temporiser et 'resetter' l'esclave en absence de réponse durant cette temporisation.

* après réception de l'octet d'état, la main est rendue au contrôleur.

* l'esclave a le contrôle complet du bus tant que l'octet d'état n'a pas été renvoyé.

* chaque esclave doit avoir un numéro de contrôle défini par l'utilisateur.

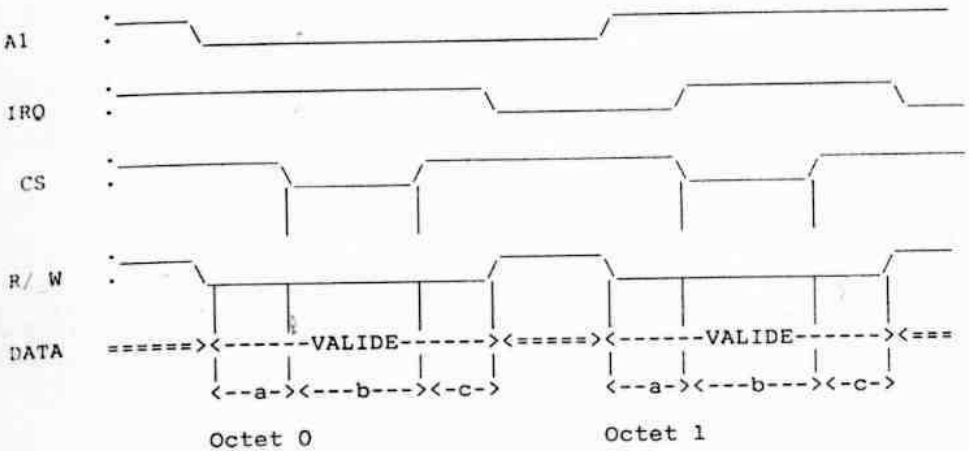




MATERIEL.

----- Phase de Commande --------------------

Direction des Données : du maître VERS l'esclave.

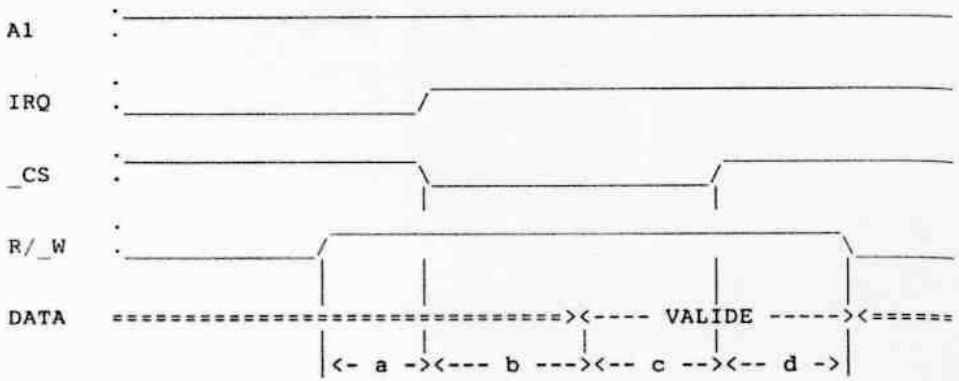



Chronogramme :

a) 60 nanosecondes (maximum)
b) 250 nanosecondes (maximum)
c) 20 nanosecondes (maximum)

----- Phase d'Etat -----------------------

Direction des Données : de l'esclave VERS le maître.




Chronogramme :

a) 50 nanosecondes (maximum)
b) 150 nanosecondes (maximum)
c) 100 nanosecondes (maximum)
d) 80 nanosecondes (maximum)

LOGICIEL.

----- Octet de Sélection du Contrôleur ---------------

Octet 0 |xxx-----|
Numéro du Contrôleur

----- Octet d'Etat retourné --------------------------

Octet 0 |--------|





2.2. Niveau 2

* englobe le niveau 1.
* la commande de 'TEST D'UNITE DISPONIBLE' fonctionne en scrutation.
* 'PAS D'ERREUR' doit être interprété comme contrôleur prêt.
* 'PERIPHERIQUE NON DISPONIBLE doit être interprété comme contrôleur non disponible.

LOGICIEL.

----- Octet de Commande -----------------------------

Octet 0 |xxxxxxxx|
Code opératoire
Numéro du contrôleur

----- Répertoire des Codes de commandes -------------

| Code op. | Commande |
|---|---|
| 0x00 | Test d'unité prête |

----- Octet d'état retourné -------------------------

Octet 0 |xxxxxxxx|
Code d'erreur
Numéro de périphérique

Codes erreurs de périphériques :

0x00 Pas d'erreur
0x04 Périphérique non disponible

Autres Codes erreur :

0x30 Auto-test du contrôleur défaillant

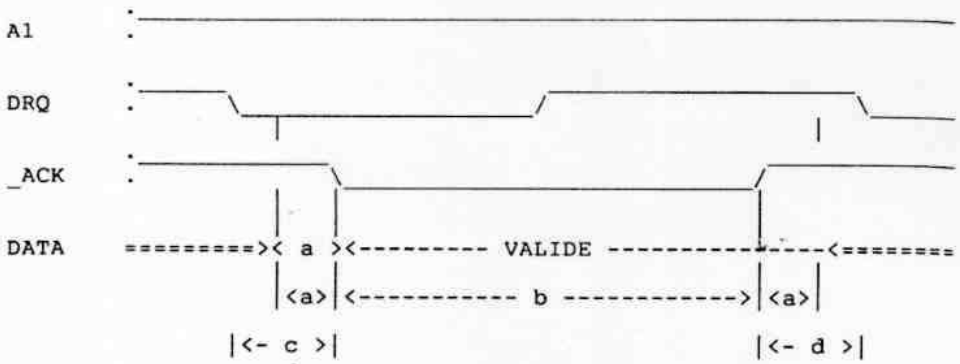
## 2.3. Niveau 3

* englobe le niveau 1 et le niveau 2.

MATERIEL.

----- Phase d'envoi de Données ----------

Direction des Données : du maître VERS l'esclave.




Chronogramme :

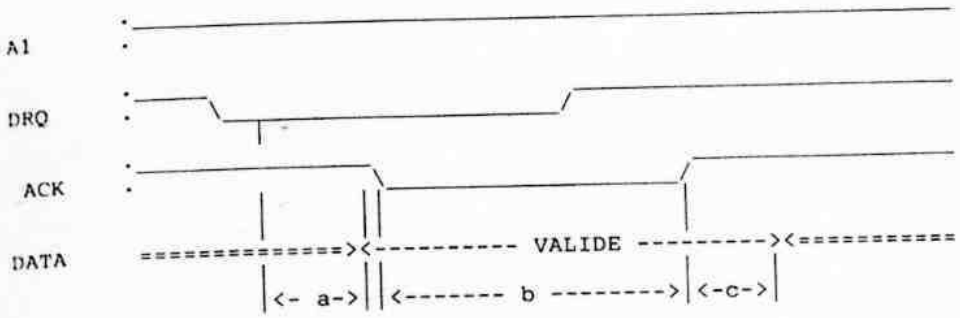
a) 60 nanosecondes (maximum)
b) 250 nanosecondes (maximum)
c) 240 nanosecondes (maximum)
d) 240 nanosecondes (maximum)





----- Réception de Données ----------

Direction des Données : de l'esclave VERS le maître.




Chronogramme :

a) 60 nanosecondes (maximum)
b) 250 nanosecondes (maximum)
c) 50 nanosecondes (maximum)

LOGICIEL

----- Descripteur de Bloc commande ACSI ----------

```
Octet 0   |xxxxxxxx|
          |||||||
          |||------------ Code opératoire
          -------------- Numéro du contrôleur

Octet 1   |xxxxxxxx|
          |||||||
          |||------------ Haut adresse du bloc
          -------------- Numéro de périphérique

Octet 2   |xxxxxxxx|
          |||||||
          -------------- Milieu adresse du bloc

Octet 3   |xxxxxxxx|
          |||||||
          -------------- Bas adresse du bloc

Octet 4   |xxxxxxxx|
          |||||||
          -------------- Nombre de blocs

Octet 5   |xxxxxxxx|
          |||||||
          -------------- Octet de contrôle
```

----- Répertoire des codes de commandes ----------

| Code op. | Commande | |
|---|---|---|
| 0x00 | Test d'Unité disponible | |
| 0x08 | Lecture | * |
| 0x0A | Ecriture | * |
| 0x0B | Recherche | |
| 0x1A | Mode de lecture | |

* transfert multiple de secteurs avec recherche automatique.

------ Codes d'erreur de commande ----------

```
0x20    Commande invalide
0x21    Adresse invalide
0x23    Adresse de fin trop grande
0x24    Argument incorrect
0x25    Numéro de périphérique incorrect
```





1. Initialisation de l'ACSI

* doit transférer les données dans des blocs d'adresse paire

* doit gérer la variable système de verrouillage des drives ($43E, voir le Guide du Bios).

----- Séquence de 'poignée de main' à l'initialisation ---

* Charge le registre d'adresse de base du DMA.

* Inverse le signal Ecriture/_Lecture pour annuler l'état (registre de Contrôle de Mode du DMA).

* Sélection du mode Lecture ou Ecriture du DMA (registre de Contrôle de Mode du DMA).

* Sélection du registre Compte de Secteurs du DMA (registre de Contrôle de Mode du DMA).

* Charge le registre Compte de Secteurs du DMA (déclanche l'opération DMA).

* Sélectionne le registre de Commande interne du contrôleur (registre de Contrôle de Mode du DMA).

* Termine la sélection du contrôleur en plaçant A0 à 0.

* Place A0 à 1 pour la suite des octets de commande.

* Après le dernier octet de commande sélection du contrôleur (registre de Contrôle de Mode du DMA).

* DMA actif jusqu'à ce que le compte de secteurs soit nul (registre d'Etat du DMA, à ne pas scruter lorsque le DMA est actif).

* teste l'octet d'état d'erreur du DMA (registre d'Etat DMA).

* teste l'octet d'état du contrôleur.

Note du traducteur:

Le listing suivant correspond à la gestion élémentaire d'un disque dur (périphérique connecté à l'interface ACSI). Seule l'unité 0 connectée au DMA et une seule partition (C) du disque dur sont gérées ici. Pour une documentation plus complète sur la gestion de plusieurs unités reliées au bus ACSI, on se référera à 'AHDI.PRG' (disquette de lancement du disque dur) ou aux ouvrages mentionnés en bibliographie.

Les commentaires des auteurs de ce programme source ont été conservés, quelques ajouts destinés à faciliter la lecture du listing ont été ajoutés par le traducteur.

```
*-------------------------------------------------------------*
*
*       Gestionnaire de disque dur ST SASI
*       (c)1985 Atari Corp.
*
*-------
*  9-Avr-1985 lmd   Mise en route. "Super, ça semble marcher..."
* 14-Avr-1985 lmd   lien avec le BIOS (*** POUR L'INSTANT ***)
* 20-Avr-1985 lmd   essayé avec le contrôleur WD.
* 24-Jun-1985 jwt   essayé avec Adaptec.
* 01-Jui-1985 jwt   semble marcher, ajout de plus de traitements
*                    des codes erreur et du formatage.
* 22-Jui-1985 jwt   change la temporisation de wdc/wd1 en début
*                    de commande, ajoute des 'move.w #$8A,wd1'
*                    pour changer A1.
* 23-Jui-1985 jwt   utilisation de 'move.l' pour toutes les écri-
*                    tures successives de wdc/wd1 car cela modi-
*                    fie A1 suffisamment rapidement pour que
*                    le DMA (ancien) ne génère pas incorrectement
*                    deux sélections.
*
*-------------------------------------------------------------*

* VARIABLES SYSTEME *

flock           equ  $43E       * état verrouillage FIFO
hdv_init        equ  $46A       * hdv_init()
*                                 ptr routine init. disque
hdv_bpb         equ  $472       * hdv_bpb(periph)
*                                 demande bloc param. disque
hdv_rw          equ  $476       * hdv_rw(rw,tampon,compte,
*                                  numsect,periph)
*                                 lecture/écriture sur disque
hdv_boot        equ  $47A       * hdv_boot()
*                                 lecture du secteur boot
hdv_mediach     equ  $47E       * hdv_mediach(periph)
*                                 renvoie état changement
_drvbits        equ  $4C2       * carte des disques actifs
_dskbufp        equ  $4C6       * ptr tampon système disque
```





```
* DEFINITIONS *

nbessai         equ  3              * nombre d'essais - 1

* -------------- Installation -----------------

     .globl     i_sasi
i_sasi:
     bra        i_sasi2             * vers la vraie initialisation

     dc.b       '@(*)ahdx v0.04',$0D,$0A,0,$1A

* -------------- Fin d'en-tête ----------------

*+
* LONG hbdp(periph) - retourne un pointeur vers le bloc de
*                     paramètres du disque (ou un NUL).
*
* Paramètre d'appel : periph   4(sp).W      numéro du disque
*
*-
hbpb:
     move.w     4(sp),d0            * d0 = numéro du disque
     move.l     o_bpb,a0            * a0 = ancien vecteur hdv_bpb
     lea        _sasi_bpb(pc),a1    * a1 = notre vecteur hdv_bpb
     bra        check_dev           * choix vecteur à appeler

*+
* LONG rw(rw, tampon, compte, numsect, periph)
*
* Paramètres d'appel : periph    $E(sp).W    numéro du disque
*                      numsect   $C(sp).W    numéro 1er secteur
*                      compte    $A(sp).W    nombre de secteurs
*                      tampon    $6(sp).L    adresse du tampon
*                      rw        $4(sp).W    type d'opération
*                                            (0=lecture,1=écrit.)
*
*-
hrw:
     move.w     $E(sp),d0           * numéro du disque
     move.l     o_rw,a0             * a0 = ancien vecteur hdv_rw
     lea        _sasi_rw(pc),a1     * a1 = notre vecteur hdv_rw
     bra        check_dev           * choix du vecteur à appeler

*+
* LONG hmediach(periph)
*
* Paramètre d'appel : periph   4(sp).W      numéro du disque
*
*-
hmediach:
     move.w     $4(sp),d0           * numéro du disque
```

```
     move.l    o_mediach,a0       * a0=ancien vecteur hdv_mediach
     move.l    _sasi_mch(pc),a1   * a1=notre vecteur hdv_mediach

*+
* check_dev - appel de la nouvelle routine si le périphérique est
*             la partition C du disque dur, sinon exécute l'an-
*             cienne routine (vecteur système).
*
* Valeurs reçues : d0.w  = numéro du périphérique
*                  a0    -> ancien vecteur
*                  a1    -> nouveau vecteur
*                  a5    -> 0L (pointeur de base des RAMs)
*
* Sauts vers : (a1) si c'est le périphérique C (disque dur)
*              (a0) sinon
*-
check_dev:
     cmpi.w    #2,d0              * est-ce le périphérique C ?
     bne       chkd_f             * (non, exécute (a0) )
     move.l    a1,a0              * (oui, exécute (a1) )
chkd_f:
     jmp       (a0)               * exécute la routine


* -------------- Gestionnaire du niveau Median -----------------

*+
* _sasi_init - initialise le périphérique
*
* Paramètres d'appel :   aucun
*
* Paramètres de retour : d0 (OK si nul, erreur si négatif)
*
* fonction réalisée par _hinit... mais l'assembleur ne permet pas
*   une référence en avant à _hinit
*-
*    .globl    _sasi_init
*_sasi_init:   equ    _hinit


*+
* _sasi_bpb - retourne le bloc de paramètres du disque dur
*
* Synopsis : LONG _sasi_bpb(periph)
*            WORD periph                numéro du disque
*
* Paramètre de retour : 0 ou un pointeur vers le bloc de param.
*
*-
     .globl    _sasi_bpb
_sasi_bpb:
     move.l    #le_bpb,d0         * voir le_bpb en page suivante
     rts
```





```
*+
*  sasi_rw - lecture/écriture de secteurs sur disque dur
*
* Synopsis : _sasi_rw(rw, tampon, compte, numsect, periph)
*
* Paramètres d'appel : periph    $E(sp).W    numéro du disque
*                      numsect   $C(sp).W    numéro 1er secteur
*                      compte    $A(sp).W    nombre de secteurs
*                      tampon    $6(sp).L    adresse du tampon
*                      rw        $4(sp).W    type d'opération
*                                            (0=lecture,1=écrit.)
*
*-
     .globl    _sasi_rw
 sasi_rw
     move.w    #nbessai,essais    * init. compteur nombre d'essais
sasrw1:
     moveq     #0,d0              * d0 = 0L
     move.w    $C(sp),d0          * numéro du premier secteur (L)
     move.w    $A(sp),d1          * nombre de secteurs (W)
     move.w    $6(sp),d2          * adresse du tampon (L)
     move.w    $4(sp),d3          * type d'opération (W)

     clr.w     -(sp)              * périphérique disque dur = 0
     move.l    d2,-(sp)           * adresse du tampon
     move.w    d1,-(sp)           * compte de secteurs
     move.l    d0,-(sp)           * numéro du secteur de départ
     tst.w     d3                 * lecture ou écriture ?
     bne       sasrw2             * (si écriture -> sasrw3)
     bsr       _hread             * lecture de secteurs
     bra       sasrw3             * est-ce ok ?
sasrw2:
     bsr       _hwrite            * écriture de secteurs
sasrw3:
     add.w     #12,sp             * nettoie la pile
     tst.l     d0                 * y-a-t'il eu erreur ?
     beq       sasrwfin           * non -- tout va bien
     subq.w    #1,essais          * met à jour compteur d'essais
     bpl       sasrw1             * et recommence jusqu'à 4 fois
sasrwfin
     rts

*+
*  sasi_mch - teste si le disque dur a changé (impossible, car le
*             ST ne dispose pas de disque dur amovible!)
*
* Synopsis :  _sasi_mch(periph)
*             WORD periph          numéro du périphérique
*
* Paramètre de retour : toujours 0.
*-
     .globl    _sasi_mch
 sasi_mch
     moveq     #0,d0              * d0 = 0L
     rts
```

```
*+
* Bloc de Paramètres pour une partition de disque dur de 10 Mo
*-
le_bpb:   dc.w      512          * nombre d'octets par secteur
          dc.w      2            * nombre de secteurs par bloc
          dc.w      1024         * nombre d'octets par bloc
          dc.w      16           * nombre de secteurs catalogue
*                                  (niveau 0: 256 entrées)
          dc.w      41           * taille FAT en secteurs
          dc.w      42           * numéro secteur début 2e FAT
          dc.w      99           * nbre secteurs boot + FATs +
*                                  catalogue niveau 0
          dc.w      10190        * nombre de blocs de données
          dc.w      1            * drapeau (entrées FATS sur
*                                  16 bits)

* --------------- Gestionnaire de niveau bas -------------------

* VARIABLES SYSTEME *

flock           equ  $43E        * verrouillage FIFO
_hz_200         equ  $4BA        * compteur interruptions 200Hz

* VARIABLES 'MATERIEL' *

wdc             equ  $FF8604     * registre Compte de secteur DMA
wdl             equ  $FF8606     * registre mode de Contrôle DMA
wdcwdl          equ  wdc         * pour écrire ds les 2 registres
dmahi           equ  $FF8609     * octet 2 adresse de base DMA
dmamid          equ  dmahi+2     * octet 1 adresse de base DMA
dmalow          equ  dmamid+2    * octet 0 adresse de base DMA
gpip            equ  $FFFA01     * registre E/S du 68901

* DEFINITIONS *

ltempo          equ  $80000      * longue temporisation
stempo          equ  $80000      * courte(!) temporisation

*+
* LONG _qdone() - Attend la fin d'une opération
*
* Paramètre d'appel :    aucun
*
* Paramètres de retour : 0  -> OK, temporisation non terminée
*                        <0 -> temporisation terminée
*
* Regitre utilisé : d0
*-
_qdone:
     move.l    #ltempo,temps     * initialise compteur de temps
qd1:
     subq.l    #1,temps          * décrémente compteur de temps
     bmi       qdq               * (si tempo terminée)
```





```
     move.b    gpip,d0           * teste si interruption du
     and.b     #$20,d0           * controleur via le 68901
     bne       qd1               * (non, boucle)

     moveq     #0,d0             * retourne d0=0L (OK)
qdq
     rts

*+
* WORD _endcmd()  - Attend la fin d'une commande SASI
*
* Paramètres d'appel :   d0 -> valeur à écrire sur wdl
*
* Paramètres de retour : d0.W = 0 -> OK
*                        d0.W < 0 -> temporisation épuisée
*                        d0.W > 0 -> erreur DMA
*
*
* Registres utilisés : d0, d1
*-
endcmd:
     move.w    d0,d1             * sauve la valeur à envoyer

     bsr       _qdone            * attend que le DMA soit prêt
     bmi       endce             * si tempo épuisée, fin(dommage)

     move.w    d1,wdl            * envoi de la commande au DMA
     nop                         * 4 cycles de temporisation
     move.w    wdc,d0            * que répond le DMA ?
     and.w     #$00FF,d0         * garde la réponse (octet 0)
endce
     rts

*+
*  hinit(periph)  - Initialise le disque dur
*
* Paramètre d'appel :  WORD periph   numéro du disque
*
* Paramètre de retour : -1 si pas de disque dur, 0 si OK
*-
     .globl    _sasi_init
sasi_init:
hinit:
     pea       actur             * ptr bloc octets commande test
     bsr       _dosahdxc         * envoi de la commande de test
     addq.l    #4,a7             * nettoie la pile
     rts

*+
*  hread(numsect, compte, tampon, periph)   Lecture de secteurs
*
* Paramètres d'appel : periph    $E(sp).W    numéro du disque
*                      tampon    $A(sp).L    adresse du tampon
*                      compte    $8(sp).W    nombre de secteurs
*                      numsect   $4(sp).L    numéro 1er secteur
```

```
*
* Paramètre de retour : D0 = 0  ->  OK
*                       D0 = -1 ->  tempo épuisée
*                       D0 > 0  ->  erreur DMA
*-
     .globl    _hread
_hread:
     st        flock              * verrouille $43E (à $FF)

     move.w    #$88,wdl           * accès au controleur, A1 = 0
     move.l    #$0008008A,wdcwdl  * commande lecture + accès

     move.l    $A(sp),-(sp)       * empile adresse du tampon
     bsr       _setdma            * place adresse de base DMA
     addq.w    #4,sp              * nettoie la pile

     bsr       _setss             * fixe le secteur et la taille
     bmi       _hto               * si tempo épuisée, reset/fin

     move.w    #$190,wdl          * place en Ecriture (inversion)
     nop                          * 4 cycles de temporisation
     move.w    #$90,wdl           * repasse en Lecture (normal!)
     nop                          * 4 cycles de temporisation
     move.w    8(sp),wdc          * écriture compte secteurs ->DMA
     nop                          * 4 cycles de temporisation
     move.w    #$8A,wdl           * vers registre controle DMA
     nop                          * 4 cycles de temporisation
     clr.l     wdcwdl             * octet contrôle wdc=0, wdl=0

     move.w    #$8A,d0            * pour attente lecture
     bsr       _endcmd            * tempo et test gpip 68901
hrx:
     bra       _hdone             * attente IRQ et fin

*+
* _hwrite(numsect, compte, tampon, periph)  Ecriture de secteurs
*
* Paramètres d'appel : periph    $E(sp).W    numéro du disque
*                      tampon    $A(sp).L    adresse du tampon
*                      compte    $8(sp).W    nombre de secteurs
*                      numsect   $4(sp).L    numéro 1er secteur
*
* Paramètre de retour : D0 = 0  ->  OK
*                       D0 = -1 ->  tempo épuisée
*                       D0 > 0  ->  erreur DMA
*-
     .globl    _hwrite
_hwrite:
     st        flock              * verrouille $43E (à $FF)

     move.l    $A(sp),-(sp)       * empile adresse du tampon
     bsr       _setdma            * place adresse de base DMA
     addq.w    #4,sp              * nettoie la pile
```





```
     move.w    #$88,wdl           * accès au controleur, A1 = 0
     move.l    #$0008008A,wdcwdl  * commande lecture + accès

     bsr       _setss             * fixe le secteur et la taille
     bmi       _hto               * si tempo épuisée, reset/fin

     move.w    #$90,wdl           * place en Lecture (inversion)
     nop                          * 4 cycles de temporisation
     move.w    #$190,wdl          * repasse en Ecriture (normal!)
     nop                          * 4 cycles de temporisation
     move.w    8(sp),wdc          * écriture compte secteurs ->DMA
     nop                          * 4 cycles de temporisation
     move.w    #$18A,wdl          * vers registre controle DMA
     nop                          * 4 cycles de temporisation
     clr.l     wdcwdl             * octet contrôle wdc=0, wdl=0

     move.w    #$18A,d0           * pour attente écriture
     bsr       _endcmd            * tempo et test gpip 68901
hwx:
     bra       _hdone             * attente IRQ et fin

*+
* _wd_format - formatage de disque dur
*
* Paramètre d'appel : aucun
*
* Paramètre de retour : 0 si OK, négatif si erreur
*-
     .globl    _wd_format
_wd_format:
     lea       acfmt,a0           * ptr vers bloc commande format
     moveq     #0,d0              * nettoie registre de travail
     st        flock              * verrouille FIFO ($43E = #$FF)
     move.w    #$88,wdl           * accès au contrôleur, A1 = 0
     move.b    (a0)+,d0           * octet 0 du bloc commande
     swap      d0                 * placé dans octet 2 pour wdc
     move.w    #$8A,d0            * pour placer dans wdl
     move.l    d0,wdcwdl          * envoi début de commande

     moveq     #(5-1),d1          * reste 5 octets à envoyer
fmt1:
     bsr       _qdone             * attente tempo, suppose qu'il
     bmi       _hto               * n'y a que l'unité 0
     move.b    (a0)+,d0           * octet de commande suivant
     swap      d0                 * pour le placer dans wdc
     move.w    #$8A,d0            * à envoyer dans wdl
     move.l    d0,wdcwdl          * envoi suite de commande
     dbf       d1,fmt1            * boucle pour les 5 octets
fmt2:
     btst      #5,gpip            * attente de fin via le 68901
     bne       fmt2               * tout le temps nécessaire!

     move.w    wdc,d0             * récupère octet d'état
     andi.w    #$00FF,d0          * seul l'octet est intéressant
```

```
     bra  _hdone                    * tempo et fin après IRQ

*+
* _wd_setup   - Initialise les paramètres WD pour le disque dur
*
*-
     .globl    _wd_setup
_wd_setup:
     st        flock                * verrouille $43E (à $FF)

     pea       adap_parms(pc)       * empile adresse du tampon
     bsr       _setdma              * place adresse de base DMA
     addq.w    #4,sp                * nettoie la pile

     move.w    #$88,wdl             * accès au controleur, A1 = 0
     move.l    #$0015008A,wdcwdl    * commande sélection mode+accès

     bsr       _qdone               * tempo jusque commande reçue
     bmi       _wdx                 * si tempo épuisée, reset/fin
     move.l    #$0000008A,wdcwdl    * adresse de bloc haute
     bsr       _qdone               * tempo jusque commande reçue
     bmi       _wdx                 * si tempo épuisée, reset/fin
      bsr      _qdone                * tempo jusque commande reçue
     bmi       _wdx                 * si tempo épuisée, reset/fin
     move.l    #$0000008A,wdcwdl    * adresse de bloc basse
     bsr       _qdone               * tempo jusque commande reçue
     bmi       _wdx                 * si tempo épuisée, reset/fin
     move.l    #$0016008A,wdcwdl    * 22 octets de paramètres
     bsr       _qdone               * tempo jusque commande reçue
     bmi       _wdx                 * si tempo épuisée, reset/fin

     move.w    #$90,wdl             * place en Lecture (inversion)
     nop                            * 4 cycles de temporisation
     move.w    #$190,wdl            * repasse en Ecriture (normal!)
     nop                            * 4 cycles de temporisation
     move.w    #1,wdc               * un secteur, pas plus
     nop                            * 4 cycles de temporisation
     move.w    #$18A,wdl            * vers registre controle DMA
     nop                            * 4 cycles de temporisation
     move.l    #$00000100,wdcwdl    * octet contrôle wdc=0, wdl=100

     move.w    #$18A,d0             * pour attente écriture
     bsr       _endcmd              * tempo et test gpip 68901
hdx:
     bra       _hdone               * attente IRQ et fin

* ------ Paramètres de sélection de mode pour 10 Mo ------------
*
* taille de bloc: $200, comptage de cylindres: $262,
* comptage tête: $2, cylindre écriture réduit: $100,
* cylindre précompte écriture: $100, position d'atterrissage: $0
* code d'impulsion : $2 (recherches mises en tampon à 12 micros.
```





```
*
adap_parms: dc.b  $00,$00,$00,$08,$00,$00,$00,$00,$00,$00
            dc.b  $02,$00,$01,$02,$62,$02,$01,$00,$01,$00,$00,$02

*+
* LONG _dosahdxc(adresse)    CHAR *adresse
*     Envoi de commande simple vers le disque dur
*-
     .globl    _dosahdxc
_dosahdxc:
     movea.l   4(sp),a0             * pointeur vers bloc de commande
     moveq     #0,d0                * nettoie le registre de travail
     st        flock                * verrouille FIFO ($43E à #$FF)
     move.w    #$88,wdl             * accès au contrôleur, A1 = 0
     swap      d0                   * pour envoyer au wdc
     move.w    #$8A,d0              * pour envoyer au wdl, A1 = 1
     move.l    d0,wdcwdl            * envoi vers le contrôleur

     moveq     #(5-1),d1            * reste 5 octets de commande
dosac1:
     bsr       _qdone               * une unité seulement!
     bmi       _hto                 * si tempo épuisée, reset/fin
     move.b    (a0)+,d0             * octet suivant de commande
     swap      d0                   * pour envoi au wdc
     move.w    #$8A,d0              * pour envoi au wdl (A1 = 1)
     move.l    d0,wdcwdl            * envoi suite de la commande
     dbf       d1,dosac1            * boucle pour les 5 octets

     bsr       _qdone               * attend commande reçue
     bmi       _hto                 * si tempo épuisée, reset/fin

     move.w    wdc,d0               * demande octet d'état
     andi.w    #$00FF,d0            * ne conserve que l'octet

     bra       _hdone               * tempo et fin après IRQ

*+
* void _setdma(adresse)             Place adresse de base du DMA
* LONG adresse;
*-
     .globl    _setdma
_setdma:
     move.b    7(sp),dmalow         * octet 0 de l'adresse
     move.b    6(sp),dmamid         * octet 1 de l'adresse
     move.b    5(sp),dmahi          * octet 2 de l'adresse

*+
* WORD _setss       Fixe le numéro et le nombre de secteur(s)
*-
     .globl    _setss
_setss
     move.w    #$8a,wdl             * accès au controleur (A1 = 1)
     bsr       _qdone               * tempo tant que pas reçu
     bmi       setsse               * si tempo épuisée, fin
```

```
    move.b    $9(sp),d0          * octet 2 numéro de secteur
    move.b    $E(sp),d1          * numéro du disque
    lsl.b     #5,d1              * x 32 numéro du disque
    or.b      d1,d0              * OU avec numéro disque
    swap      d0                 * pour envoi au wdc
    move.w    #$8A,d0            * pour envoi au wdl(A1 = 1)
    move.l    d0,wdcwdl          * envoi de la commande
    bsr       _qdone             * tempo tant que pas reçu
    bmi       setsse             * si tempo épuisée, fin

    move.b    $A(sp),d0          * octet 1 numéro de secteur
    swap      d0                 * pour envoi au wdc
    move.w    #$8A,d0            * pour envoi au wdl (A1 = 1)
    move.l    d0,wdcwdl          * envoi de la commande
    bsr       _qdone             * tempo tant que pas reçu
    bmi       setsse             * si tempo épuisée, fin

    move.b    $B(sp),d0          * octet 0 numéro de secteur
    swap      d0                 * pour envoi au wdc
    move.w    #$8A,d0            * pour envoi au wdl (A1 = 1)
    move.l    d0,wdcwdl          * envoi de la commande
    bsr       _qdone             * tempo tant que pas reçu
    bmi       setsse             * si tempo épuisée, fin

    move.w    $C(sp),d0          * MOT = nombre de secteurs
    swap      d0                 * pour envoi au wdc
    move.w    #$8A,d0            * pour envoi au wdl (A1 = 1)
    move.l    d0,wdcwdl          * envoi de la commande
    bsr       _qdone             * tempo tant que pas reçu
setsse:
    rts


_hto
    moveq     #$FF,d0            * d0 = -1 (tempo épuisée)
_hdone
    move.w    #$80,wdl           * désélection contrôleur d.dur
    nop                          * temporisation de 4 cycles
    tst.w     wdc                * pourquoi pas ?
    sf        flock              * déverrouille FIFO ($43E à 0)
    rts

* BSS (Variables non initialisées) *

savssp:   ds.l    1              * où sauver le pointeur de pile
temps:    ds.l    1              * compteur de temporisation
essais:   ds.w    1              * compteur de nombre d'essais
o_init:   ds.l    1              * ancien vecteur hdv_init
o_bpb:    ds.l    1              * ancien vecteur hdv_bpb
o_rw:     ds.l    1              * ancien vecteur hdv_rw
o_mediach ds.l    1              * ancien vecteur hdv_mediach
```





```
*-
* i_sasi2       Vraie routine d'initialisation
*-
i_sasi2:
    nop

ifne loadable
    clr.l     -(sp)              * si on est en mode utilisateur
    move.w    #$20,-(sp)         * on conserve la pile système
    trap      #1                 * passage en mode SUPERVISEUR
    addq.w    #6,sp              * le tour est joué!
    move.l    d0,savssp          * sauvegarde de USP
endc

    bsr       _sasi_init         * initialise le contrôleur
    tst.w     d0                 * répond-t'il correctement ?
    bne       isase              * (non -> isase)

    moveq     #0,d0              * nettoie le registre de travail
    or.l      _drvbits,d0        * d0 = carte des disques actifs
    or.l      #4,d0              * C devient un disque actif
    move.l    d0,_drvbits        * mise à jour carte des disques

    clr.l     a5                 * pointeur vers la page zéro
    move.l    hdv_bpb(a5),o_bpb  * sauve les vecteurs actuels
    move.l    hdv_rw(a5),o_rw    * des routines disque
    move.l    hdv_mediach(a5),o_mediach

    move.l    #hbpb,hdv_bpb(a5)  * place les nouveaux vecteurs
    move.l    #hrw,hdv_rw(a5)    * des routines disque
    move.l    #hmediach,hdv_mediach(a5)

ifne loadable
    move.l    savssp,-(sp)       * pile USP lors passage superv.
    move.w    #$20,-(sp)         * Retour en mode UTILISATEUR
    trap      #1
    addq.l    #6,a7              * le tour est joué, encore!
endc

ifne loadable
    clr.w     -(sp)              * code de fin = 0
    move.l    #((i_sasi2-i_sasi)+$100),-(sp)
    move.w    #$31,-(sp)         * Ptermres
    trap #1                      * conserve code et données, FIN
endc

    rts                          * FIN normale

isase:
    lea       nodmsg,a0          * ptr message 'pas de disque'
    bsr       msg                * affichage du message

ifne loadable
    move.l    savssp,-(sp)       * récupère ancien USP
```

```
     move.w    #$20,-(sp)          * Retour en UTILISATEUR
     trap      #1
     addq.l    #6,a7               * nettoie la pile
endc

     move.w    #1,-(sp)            * retour avec code erreur 1
     move.w    #$4C,-(sp)          * fonction Pterm()
     trap      #1                  * FIN du programme

msg:
     move.l    a0,-(sp)            * pointeur chaîne à afficher
     move.w    #9,-(sp)            * fonction Cconws()
     trap      #1                  * Affichage de la chaîne
     addq.l    #6,a7               * nettoie la pile
     rts

* DATA  Variables initialisées utilisées par i_sasi2 *

actur:    dc.b    0,0,0,0,0,0     * commande : test Unité prête
acfmt:    dc.b    4,0,0,0,1,0     * commande : formatage disque

nodmsg:   dc.b    'Pas de réponse du DISQUE DUR',$0D,$0A,0

.even

.end
```

Note du traducteur : Le listing ci-dessus a été assemblé avec le système de développement. Il fonctionne très bien mais ne contrôle qu'un lecteur C de 10 Mo (disque dur SH 104).





# L'ACIA MC6850

1. Caractéristiques électriques et chronogrammes
2. Fonctionnement général
3. Fonctions d'entrée-sortie
4. Registres de l'ACIA

## MC6850 ASYNCHRONOUS COMMUNICATIONS INTERFACE ADAPTER (ACIA)

L'ACIA MC6850, circuit d'interface pour communications asynchrones, permet de formater les données et de contrôler l'interfaçage pour des communications série asynchrones selon la procédure du START-STOP avec des systèmes organisés comme le microprocesseur 6800.

L'interface bus du MC6850 comprend les signaux de sélection, d'activation des échanges, d'interruption et la logique d'interfaçage autorisant le transfert de données sur un bus bidirectionnel de 8 bits. Les données reçues en parallèle sur le bus sont transmises en série (de même elles sont reçues en série et converties au format parallèle), suivant un format donné et avec un contrôle d'erreur. La configuration de fonctionnement de l'ACIA est programmable via le bus de données lors de la procédure d'initialisation du système. Un Registre de Contrôle programmable autorise des longueurs de mots variées, une division de la fréquence d'horloge, un contrôle de l'émission, un contrôle de la réception et un contrôle des interruptions. Pour les liaisons avec un périphérique ou un modem, trois lignes de contrôle sont fournies. Ces lignes permettent d'interfacer directement l'ACIA avec le modem numérique 0-600bps MC6860L.

* Transmission sur 8 et 9 bits.
* Parité paire ou impaire, sur option.
* Contrôle des erreurs de parité, de débordement et de synchronisation.
* Registre de Contrôle programmable.
* Taux de division d'horloge par 1, 16 ou 64 sur options.
* Vitesse de transmission jusqu'à 1 Mbs.
* Non prise en compte des faux bits de start.
* Fonctions de contrôle de périphérique/modem.
* Double bufférisation.
* Transmissions avec un ou deux bits de stop.

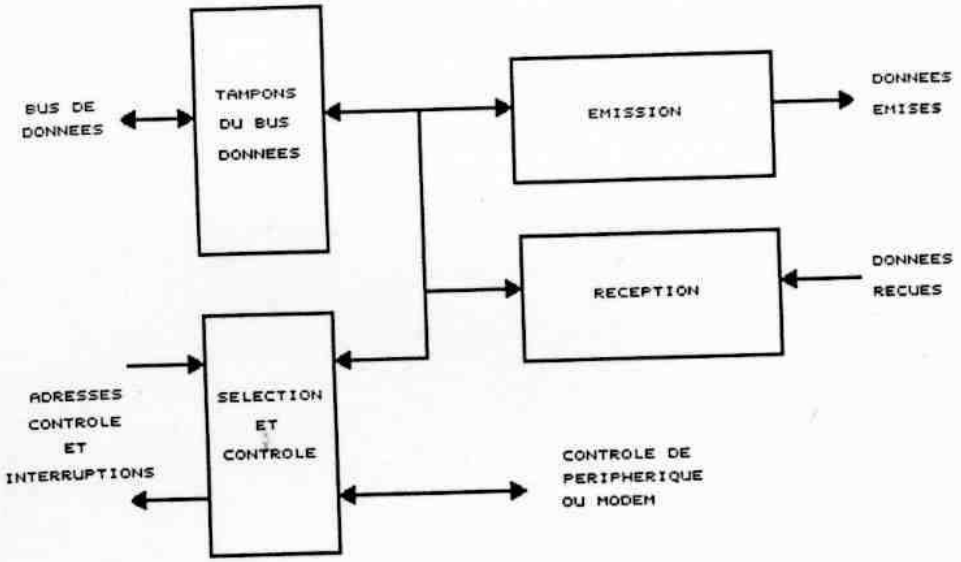
Note du traducteur : Le chapitre suivant recouvre essentiellement la documentation du constructeur du circuit. Rappelons que le ST comporte 2 circuits ACIA MC6850, le premier assurant les communications entre le microprocesseur clavier MC6301 et le circuit 68901, le second assurant les transmissions entre l'interface MIDI et le circuit 68901.



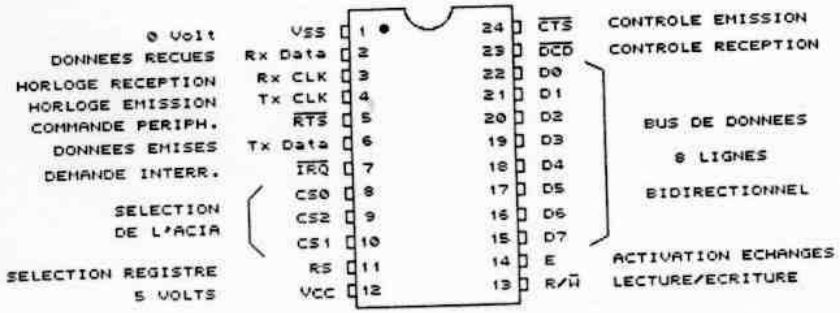


## 1. CARACTERISTIQUES ELECTRIQUES ET CHRONOGRAMMES

DIAGRAMME FONCTIONNEL ELEMENTAIRE DE L'ACIA MC6850




SCHEMA DE BROCHAGE DE L'ACIA MC6850




Circuit MOS (N-Channel, silicon-gate), le MC6850 est implanté sur le ST dans sa version boitier plastique.

VALEURS LIMITES

| Caractéristiques | Symboles | Valeurs | Unités |
|---|---|---|---|
| Tension d'alimentation | VCC | -3,0 à 7,0 | V |
| Tension d'entrée | VIN | -3,0 à 7,0 | V |
| Température de fonctionnement | TAMB | 0 à 70 | °C |
| Température de stockage | TSTG | -55 à +150 | °C |
| Résistance thermique | RTH(J/A) | 120 | °C/W |

Les entrées de ce circuit sont protégées contre les hautes tensions statiques et les champs électriques; toutefois, il est recommandé de prendre les précautions normales pour éviter toute tension supérieure aux valeurs limites sur ce circuit à haute impédance.

CARACTERISTIQUES ELECTRIQUES (VCC=5V±5%, VSS=0, TAMB=0 à 70°C)

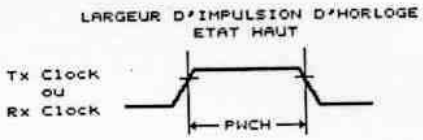
| Paramètres | Symbole | Min. | Typ. | Max. | Unité |
|---|---|---|---|---|---|
| Tension d'entrée état haut | VIH | VSS+2,0 | - | VCC | V |
| Tension d'entrée état bas | VIL | VSS-0,3 | - | VSS+0.8 | V |
| Courant de fuite en entrée R/W,CS0,CS1,CS2,Enable, RS,Rx D,Rx C,CTS,DCD | II | - | 1,0 | 2,5 | uA |
| Courant d'entrée à l'état haute impédance (D0-D7) | ITSI | - | 2,0 | 10 | uA |
| Tension de sortie état haut D0-D7,Tx D, RTS | VOH | VSS+2,4 | - | - | V |
| Tension de sortie état bas | VOL | - | - | VSS+0,4 | V |
| Courant de fuite en sortie IRQ | ILOH | - | 1,0 | 10 | uA |
| Puissance dissipée | PD | - | 300 | 525 | mW |
| Capacités en entrée E,Tx C,Rx C,R/W,RS,Rx D, D0-D7,CS0,CS1,CS2,CTS,DCD | CIN | - | 10<br>7,0 | 12,5<br>7,5 | pF |
| Capacité en sortie RTS,IRQ | COUT | - | - | 5,0 | pF |

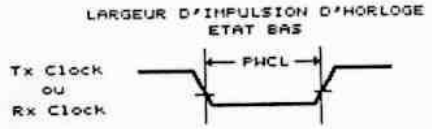


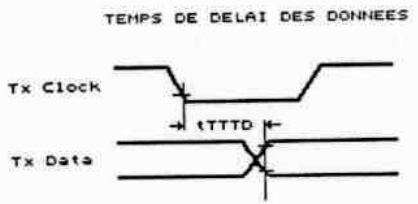


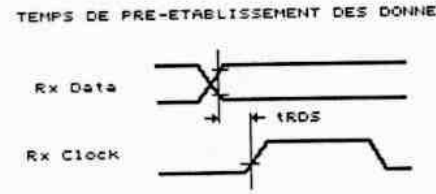
CHRONOGRAMME DES TRANSMISSIONS SERIE

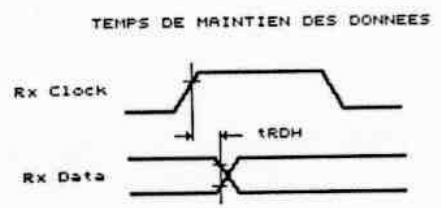
| Paramètres | | Symboles | Min. | Max. | Unités |
|---|---|---|---|---|---|
| Fréquence d'horloge | mode 1/1<br>mode 1/16<br>mode 1/64 | fC | | 500<br>800<br>800 | KHz |
| Largeur d'impulsion état haut | | PWCH | 600 | | ns |
| Largeur d'impulsion état bas | | PWCL | 600 | | ns |
| Temps de délai des données Tx | | tTDD | | 1000 | ns |
| Temps de pré-établissement des données Rx | | tRDSU | 500 | | ns |
| Temps de maintien des données Rx | | tRDH | 500 | | ns |
| Temps de délai de RTS | | tRTS | | 1000 | ns |
| Temps de relâchement de l'IRQ | | tjR | | 1200 | ns |

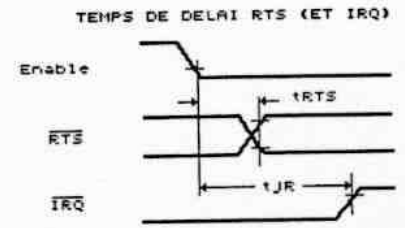


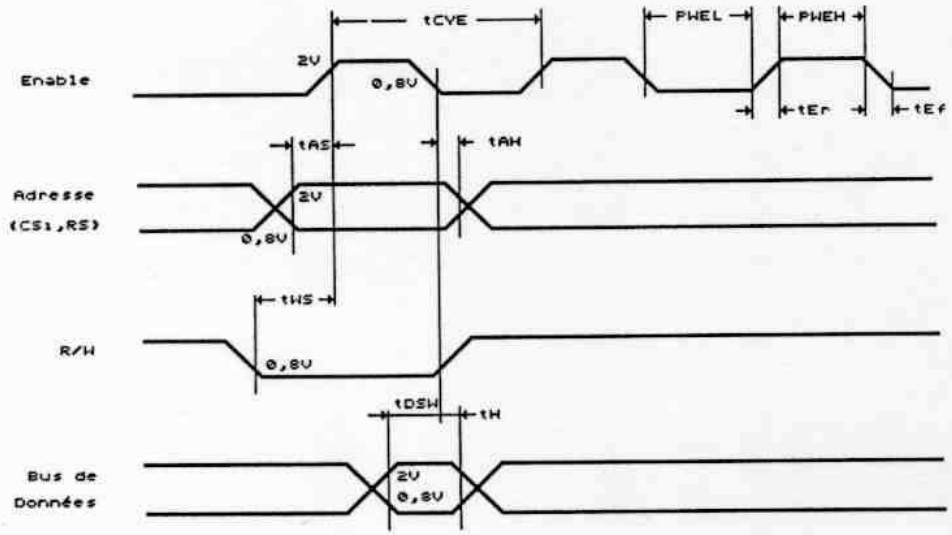



## CHRONOGRAMME D'ECRITURE DES DONNEES PAR LE MPU

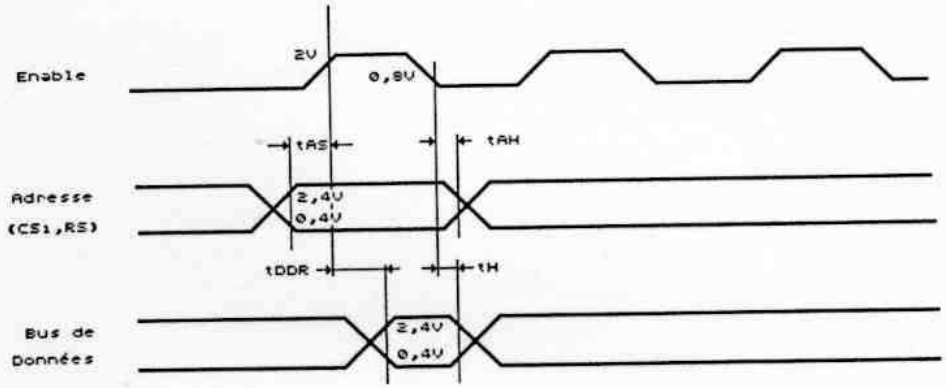



| Paramètres | Symboles | Min. | Max. | Unités |
|---|---|---|---|---|
| largeur de E à l'état haut | PWEH | 450 | 25000 | ns |
| largeur de E à l'état bas | PWEL | 430 | | ns |
| temps montée/descente de E | tER,tEF | | 25000 | ns |
| temps pré-établissent adresse | tAS | 160 | | ns |
| temps pré-établissement R/W | tWS | 130 | | ns |
| temps pré-établissement data | tDWS | 195 | | ns |
| temps de maintien adresse | tAH | 10 | | ns |
| temps de maintien des données | tH | 10 | | ns |
| temps de cycle de E | tCYCE | 1000 | | ns |



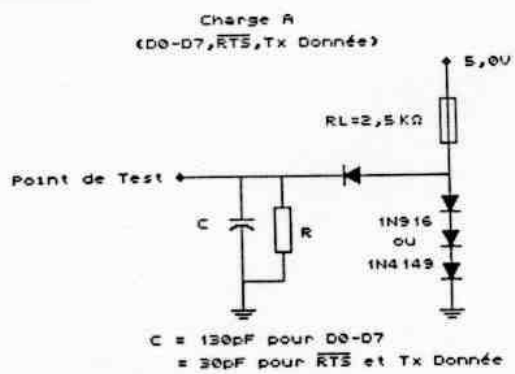


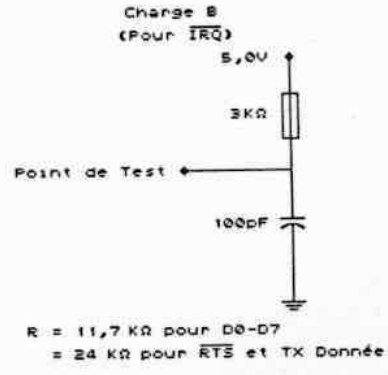
## CHRONOGRAMME DE LECTURE DES DONNEES PAR LE MPU



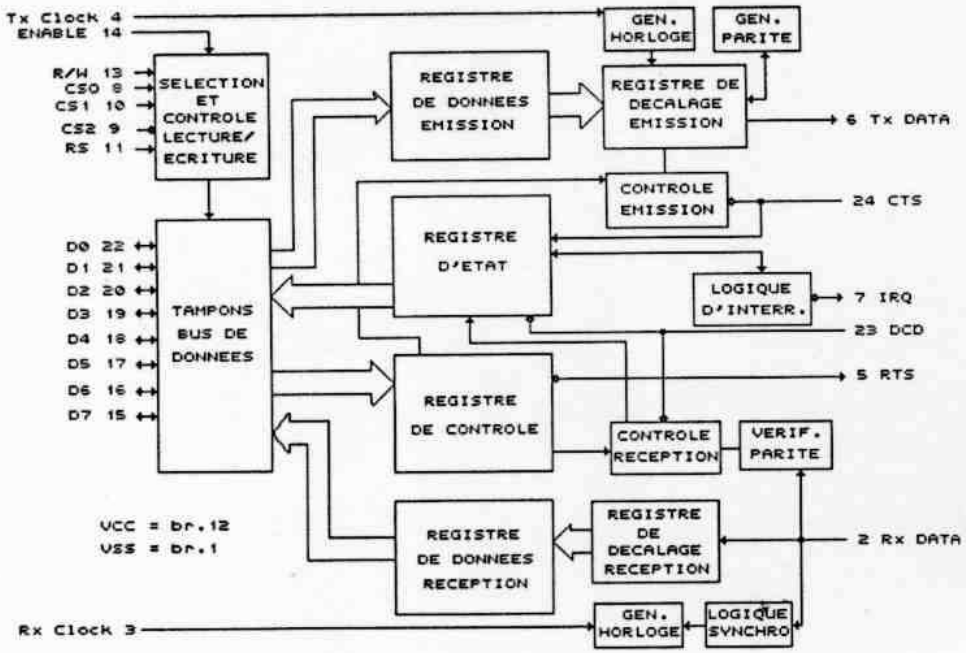

| Paramètres | Symboles | Min. | Max. | Unités |
|---|---|---|---|---|
| temps pré-établissement adr. | tAS | 160 | | ns |
| temps de délai des données | tDDR | | 320 | ns |
| temps de maintien adresse | tAH | 10 | | ns |
| temps de maintien des données | tH | 10 | | ns |

## CHARGES DE TEST



ORGANISATION DETAILLEE DE L'ACIA MC6850




## 2. FONCTIONNEMENT GENERAL

### ADRESSAGE

Pour l'interface bus, l'ACIA est 'vu' comme deux mémoires adressables. Dans sa structure interne, il est constitué de quatre registres, deux n'étant accessibles qu'en écriture et deux en lecture seule. Les registres pouvant seulement être lus sont le registre d'état et le registre de réception. Les registres pouvant seulement être écrits sont le registre de contrôle et le registre d'émission. La première mémoire ($FFFC00 pour l'ACIA clavier et $FFFC04 pour l'ACIA midi, sur le ST) donne accès en lecture au registre d'état et en écriture au registre de contrôle. La seconde mémoire ($FFFC02 pour l'ACIA clavier et $FFFC06 pour l'ACIA midi, sur le ST) donne accès en lecture au registre de réception et en écriture au registre d'émission.

### MISE SOUS TENSION. RESET MAITRE.

L'ACIA contient un circuit de mise sous tension destiné à le maintenir inhibé jusqu'à son initialisation programmée, cela afin d'éviter la transmission d'informations non significatives.





La mise sous tension est obtenue par mise à 0 de TDR et RDR, les bits 5 et 6 du registre de contrôle étant respectivement à 1 et 0, ce qui place un niveau haut sur RTS et inhibe les interruptions de l'émetteur, celles du récepteur étant inhibées par la mise à 0 du bit 7 du registre de contrôle.

Le reset maître s'obtient par la mise à 1 des bits 0 et 1 du registre de contrôle CR. Cette seconde initialisation ne modifie pas l'état des bits 5 et 6 de CR, donc l'état de la ligne RTS. Il est alors possible de programmer un octet de contrôle pour émettre ou recevoir des caractères.

### EMISSION

Une séquence d'émission habituelle commence par une lecture du registre d'état de l'ACIA par le biais soit d'une interruption, soit d'une scrutation permanente de ce registre dans une boucle de test.

Un caractère peut être émis si le bit 1 du registre d'état de l'ACIA est à 1, ce qui indique que le registre d'émission est vide. Ce caractère est alors transmis au registre de décalage d'émission qui le sérialise et l'émet sur la ligne Tx DATA après l'avoir précédé d'un bit de start et en le faisant suivre d'un ou deux bits de stop. Un bit de parité (paire ou impaire) peut être ajouté sur option, il est alors placé entre le dernier bit de données et le premier bit de stop.

Le caractère est transmis en synchronisation avec une horloge selon une fréquence égale à 1/1, 1/16 ou 1/64 de la fréquence de l'horloge appliquée en Tx Clock.

Aussitôt après l'écriture du caractère dans le registre de données émission, le registre d'état (spécialement son bit 1) peut être lu dans l'attente d'une condition 'Registre d'émission vide' (bit 1 à 1). Si cette condition est remplie, un second caractère peut être envoyé même si le premier caractère est alors en cours d'émission (à cause de la double bufférisation). Le second caractère sera automatiquement émis lorsque la transmission du premier aura pris fin.

Durant la transmission, le format peut être changé à tout moment, sauf pendant le transfert interne, sans affecter le caractère en cours d'émission. Par contre, une modification de la parité affectera même le caractère en train d'être émis.

### RECEPTION

Les données sont reçues par le biais du registre de données réception. Un rapport de division par 1 de l'horloge correspond à une synchronisation externe des bits du caractère. Des rapports de division de 1/16 ou 1/64 correspondent à une synchronisation interne des bits du caractère reçu par rapport à l'horloge Rx Clock, laquelle pilote une horloge interne.

Dans le cas d'une synchronisation interne (rapports 1/16 et 1/64), celle-ci est déclanchée par la première transition négative suivant une période de repos. Le niveau bas est échantillonné sur les fronts montants de l'horloge Rx Clock. Si l'entrée est toujours basse au 9ème échantillonnage (1/16) ou au 33ème (1/64),

ce qui représente plus de 50% de la durée d'un bit, le bit reçu est considéré comme un bit de start. Une fois le start validé, les autres bits sont acquis selon le même processus.

La validité du caractère reçu est contrôlée pendant la réception et provoque une mise à jour du registre d'état. Ensuite, le registre de décalage réception envoie l'octet au registre de données émission après suppression des bits de start et de stop ainsi que de la parité. Ceci provoque la mise à 1 du bit 0 du registre d'état, indiquant ainsi qu'un octet est disponible.

Le registre d'état peut être scruté en permanence dans l'attente d'une réception de caractère. Le registre de données réception étant double, il est possible de lire une donnée alors qu'une autre est en cours de réception.

Une absence de premier bit de stop provoque la mise à 1 du bit 4 du registre d'état.

## 3. FONCTIONS D'ENTREE-SORTIE

### SIGNAUX D'INTERFACAGE DE L'ACIA AVEC UN MICROPROCESSEUR

L'ACIA s'interface avec un microprocesseur via un bus de données de huit lignes, trois lignes de sélection de circuit (CS0 à CS2), une ligne de demande d'interruption (IRQ) une ligne de signal lecture/écriture (R/W) et une ligne d'acceptation des échanges (E). Ces signaux offrent au processeur la maîtrise complète de l'ACIA.

BUS DE DONNEES BIDIRECTIONNEL (D0-D7) : Il permet le transfert de données entre un processeur (le 68901 sur le ST) et l'ACIA. Ce bus aboutit dans l'ACIA à un amplificateur qui peut être activé ou placé dans l'état haute impédance selon l'état du signal R/W si l'ACIA est sélectionné.

ACCEPTATION DES ECHANGES (E) : Ce signal est une ligne haute impédance compatible TTL qui autorise les échanges sur le bus de données et sur les entrées horloge.

LECTURE/ECRITURE (R/W) : La ligne de lecture/écriture est une ligne haute impédance compatible TTL destinée à indiquer le sens du flux de données échangées entre le processeur et l'ACIA sur le bus de données D0-D7. Lorsque cette ligne est à l'état haut, il s'agit d'une lecture de registre de l'ACIA. Lorsqu'elle est à l'état bas, il s'agit d'une écriture de registre de l'ACIA.

SELECTION DE CIRCUIT (CS0, CS1, CS2) : Ces trois lignes haute impédance compatibles TTL permettent l'adressage de l'ACIA. L'ACIA est sélectionné lorsque les lignes CS0 et CS1 sont à l'état haut et la ligne CS2 à l'état bas (110). Dans ce cas, les transferts vers ou à partir de l'ACIA sont soumis au contrôle des signaux E, R/W et RS.

SELECTION DE REGISTRE (RS) : Ligne haute impédance compatible TTL, le signal de sélection de registre permet à l'état haut de





sélectionner les registres d'émission et de réception, à l'état bas de sélectionner les registres d'état et de contrôle. L'état de la ligne R/W permet de définir dans chaque cas quel est le registre sélectionné selon le diagramme suivant :

| | R/W = 1 | R/W = 0 |
|---|---|---|
| RS = 1 | Emission Données (TDR) | Réception de données (RDR) |
| RS = 0 | Registre Contrôle (CR) | Registre d'état (SR) |

DEMANDE D'INTERRUPTION (IRQ) : Ligne compatible TTL, active à l'état bas permettant de générer une interruption vers le 68901 (broche 26, I4). La ligne IRQ reste à l'état bas aussi longtemps que la cause d'interruption est présente et le drapeau correspondant est positionné dans le registre d'état. Le bit 7 de ce registre d'état est par ailleurs placé à 1, indiquant l'activation de la ligne IRQ.

Trois sources d'activation de l'interruption IRQ sont possibles :

1) Emission. Les interruptions de l'émetteur sont possibles (bits 6 du registre de contrôle à 0 et bit 5 à 1) et le registre d'émission est vide.
2) Réception. Les interruptions du récepteur sont possibles (bit 7 du registre de contrôle à 1) et le registre de réception est plein.
3) Perte de porteuse. Les interruptions du récepteur sont possibles (bit 7 du registre de contrôle à 1) et la ligne DCD passe à l'état haut.

### ENTREES HORLOGE

Des signaux distincts, haute impédance compatibles TTL, sont destinés à fournir des bases de temps à l'émetteur et au récepteur. Des rapports de division d'horloge de 1/1, 1/16 ou 1/64 peuvent être sélectionnés.

HORLOGE EMISSION (Tx Clock) : L'entrée horloge émission sert de base de temps à l'émission. L'émetteur initialise l'envoi de données sur une transition négative de l'horloge.

HORLOGE RECEPTION (Rx Clock) : L'entrée horloge réception sert de base de temps pour la réception si le rapport de division est de 1/16 ou 1/64. Le récepteur échantillonne les données sur une transition positive de l'horloge.

**REMARQUE** : Sur le ST, les deux ACIA ont leurs deux entrées horloge (Rx Clock et Tx Clock) reliées à un même quartz de 500KHz. Les bases de temps émission et réception sont donc identiques.

LIGNES D'ENTREE-SORTIE SERIE

RECEPTION DE DONNEES (Rx DATA) : La ligne de réception de données, haute impédance compatible TTL, reçoit les données au format série. Une synchronisation avec l'entrée horloge est effectuée lorsqu'un rapport de division d'horloge de 1/16 ou 1/64 a été choisi.

EMISSION DE DONNEES (Tx DATA) : Cette ligne sert au transfert des données provenant d'un périphérique ou d'un modem vers l'ACIA.

CONTROLE DE PERIPHERIQUE OU DE MODEM

CONTROLE DE L'EMISSION (CTS) : Ligne haute impédance compatible TTL, elle permet le controle de la transmission par le périphérique. Un passage à un niveau haut de cette ligne provoque la mise à 0 du bit 1 du registre d'état (Registre d'émission plein), ce qui inhibe par le fait même l'émission.

COMMANDE DE PERIPHERIQUE (RTS): La sortie RTS permet au processeur de contrôler un périphérique via le bus de données. L'état de la sortie RTS dépend de l'état des bits 5 et 6 du registre de contrôle. Elle est à l'état bas (active) lorsque le bit 6 est à 0 ou lorsque les bits 5 et 6 sont à 1.

CONTROLE DE LA RECEPTION (DCD): La ligne DCD, haute impédance compatible TTL, permet le contrôle de la réception par un modem ou un périphérique. A l'état haut, DCD provoque l'inhibition du récepteur et une interruption 'perte de porteuse' (voir IRQ) si le bit 7 du registre de contrôle est à 1.

**4. REGISTRES DE L'ACIA**

REGISTRE DE DONNEES EMISES (TDR) : La donnée est écrite dans ce registre durant la transition négative du signal d'activation des échanges (E) une fois que l'ACIA a été adressé avec RS à l'état haut et R/W à l'état bas. L'écriture d'une donnée dans ce registre entraîne le positionnement à 0 du bit 1 du registre d'état (TDRE). Une donnée peut alors être émise si l'émetteur est au repos et le transfert se produira dans un délai inférieur à la durée d'un bit après le front descendant du signal (E). Si un caractère est en train d'être émis, la donnée sera envoyée dés que l'émission précédente aura pris fin.

REGISTRE DE DONNEES RECUES (RDR) : La donnée est automatiquement transférée du registre de décalage de réception vers le registre de données reçues lorsqu'un caractère complet a été reçu. Cela provoque la mise à 1 du bit 0 du registre d'état (RDRF). La donnée peut alors être lue sur le bus, une fois l'ACIA sélectionné, RS égal à 1 et R/W égal à 0. Le cycle de lecture provoque la remise à zéro du bit 0 du registre d'état. Lorsque le registre de données reçues est plein, le transfert de données du registre de décalage vers le registre de données reçues est inhibé.





| Numéro de ligne bus de données | Lignes d'adressage | | | |
|---|---|---|---|---|
| | RS=1,R/W=1 Registre Emission de données | RS=1,R/W=0 Registre Réception de données | RS=0,R/W=1 Registre de contrôle | RS=0,R/W=0 Registre d'état |
| | Ecriture | Lecture | Ecriture | Lecture |
| 0 | bit donnée 0 | bit donnée 0 | Division du compteur 1 | R.réception plein |
| 1 | bit donnée 1 | bit donnée 1 | Division du compteur 2 | R. émission vide |
| 2 | bit donnée 2 | bit donnée 2 | Format des mots 1 | DCD |
| 3 | bit donnée 3 | bit donnée 3 | Format des mots 2 | CTS |
| 4 | bit donnée 4 | bit donnée 4 | Format des mots 3 | Erreur de format |
| 5 | bit donnée 5 | bit donnée 5 | Contrôle de l'émission 1 | Débordement réception |
| 6 | bit donnée 6 | bit donnée 6 | Contrôle de l'émission 2 | Erreur de parité |
| 7 | bit donnée 7 | bit donnée 7 | Contrôle de la réception | Demande interruption |

REGISTRE DE CONTROLE (CR)

Le registre de contrôle de l'ACIA comprend huit bits qui ne peuvent qu'être écrits. Il est sélectionné lorsque RS=0 et R/W=1. Ce registre contrôle les fonctions de réception, d'émission, d'autorisation d'interruptions et la maitrise du périphérique via RTS.

BITS DE SELECTION DE DIVISION D'HORLOGE (bits 0 et 1): Ces deux bits permettent de définir le rapport de division d'horloge utilisé en réception et en émission. De plus ces bits permettent de provoquer un reset maître de l'ACIA, ce qui nettoie le registre d'état et initialise l'émetteur comme le récepteur. Le reset maître n'affecte pas les autres bits du registre de contrôle. On notera qu'apès une mise sous tension, ces deux bits doivent être mis à 1 afin de provoquer un reset maître de l'ACIA. Après reset, le rapport de division d'horloge sera sélectionné. Les différents rapports possibles sont les suivants :

| CR1 | CR0 | Fonction |
|---|---|---|
| 0 | 0 | 1 / 1 |
| 0 | 1 | 1 / 16 |
| 1 | 0 | 1 / 64 |
| 1 | 1 | Reset maître |

BITS DE SELECTION DE FORMAT DE MOTS (bits 2,3,4): Les bits 2 à 4 servent à définir la longueur de mot, la parité et le nombre de bits de stop, selon le tableau suivant :

| CR4 | CR3 | CR2 | Fonction |
|---|---|---|---|
| 0 | 0 | 0 | 7 bits, parité paire, 2 bits stop |
| 0 | 0 | 1 | 7 bits, parité impaire, 2 bits stop |
| 0 | 1 | 0 | 7 bits, parité paire, 1 bit stop |
| 0 | 1 | 1 | 7 bits, parité impaire, 1 bit stop |
| 1 | 0 | 0 | 8 bits, pas de parité, 2 bits stop |
| 1 | 0 | 1 | 8 bits, pas de parité, 1 bit stop |
| 1 | 1 | 0 | 8 bits, parité paire, 1 bit stop |
| 1 | 1 | 1 | 8 bits, parité impaire, 1 bit stop |

BITS DE CONTROLE D'EMISSION (bits 5 et 6): Ces deux bits offrent la possibilité de contrôler l'interruption résultant de la condition 'registre des données émises vide' et de générer un break (espace) sur la ligne, selon le tableau suivant :

| CR6 | CR5 | Fonction |
|---|---|---|
| 0 | 0 | RTS bas, Interruption Emission inhibée |
| 0 | 1 | RTS bas, Interruption Emission active |
| 1 | 0 | RTS haut, Interruption Emission inhibée |
| 1 | 1 | RTS bas, Interruption Emission inhibée émission d'un niveau break sur la ligne |





BIT DE CONTROLE DE LA RECEPTION (bit 7): Lorsque ce bit est à 1, les interruptions suivantes deviennent possibles : Registre de données reçues plein, Débordement, perte de porteuse sur la ligne DCD.

REGISTRE D'ETAT

Ce registre permet au processeur d'obtenir des informations sur l'état de l'ACIA. Registre en lecture seule, il peut être lu lorsque RS=0 et R/W=0.

REGISTRE DE DONNEES RECUES PLEIN (bit 0, RDRF) : Positionné à 1, ce bit indique qu'une donnée reçue a été placée dans le registre des données reçues. Il est remis à 0 par une lecture du contenu du registre des données reçues par le processeur ou bien par un reset maître. Ce bit est également affecté par DCD. DCD à l'état bas indique qu'une éventuelle porteuse de données est présente et permet au bit 0 de fonctionner comme indiqué précédemment. DCD à l'état haut indique une perte de cette porteuse et force alors le bit 0 à 0, indiquant une réinitialisation du récepteur. SO retrouve son état normal dès que DCD revient à l'état bas.

REGISTRE DE DONNEES EMISES VIDE (bit 1, TDRE) : Positionné à 1, ce bit indique que le contenu du registre de données émises a été transféré au registre de décalage (puis vers le périphérique) et qu'une nouvelle donnée peut être placée dans ce registre. Si ce bit est à 0, il indique que le registre de données émises est plein. Ce bit est affecté par CTS. Si CTS est au niveau bas, TRDE a son fonctionnement normal. Sinon, le bit 1 est forcé à 0, ce qui inhibe l'émetteur.

DETECTION DE PORTEUSE DE DONNEES (bit 2, DCD) : Positionné à 0, ce bit indique qu'une porteuse est présente. S'il est à 1, une perte de porteuse a eu lieu. Dans ce cas, une demande d'interruption IRQ est générée si le bit 7 du registre de contrôle l'autorise (donc s'il est à 1). Il reste à 1 lorsque la ligne DCD est revenue à l'état bas tant qu'il n'y a pas eu lecture du registre d'état et du registre de données ou un reset maître.

CONTROLE DE L'EMISSION (bit 3, CTS) : Ce bit reflète l'état de la ligne CTS. Positionné à 0, il indique qu'il n'y pas de contrôle de l'émission par le périphérique. Son positionnement à 1 indique une désactivation du registre de données émises et s'accompagne d'une mise à 1 des bits 1 (TDRE) et 7 (IRQ) du registre d'état.

ERREUR DE FORMAT (bit 4, FE) : Une erreur de format se produit lorsque la donnée reçue n'est pas correctement précédée d'un bit de start et de bit(s) de stop. Ce bit est placé à 1 tant que la donnée erronée figure dans le registre des données reçues. Il reprendra la valeur 0 dès qu'une donnée correctement formatée aura été reçue. Un reset maître ou un niveau haut sur la ligne DCD repositionne ce bit à 0.

DEBORDEMENT (bit5, OVRN) : Placé à 0, ce bit indique une réception correcte. Placé à 1, il indique un débordement, c'est-à-dire que un ou plusieurs caractères ont été reçus avant la lecture du mot précédent. Cette mise à 1 n'a lieu que lorsque la lecture du caractère précédant le débordement s'est effectuée et elle se produit à partir du milieu du dernier bit du 2ème caractère reçu sans lecture du registre des données reçues. Une nouvelle lecture replace ce bit à 0, de même qu'un niveau haut sur DCD ou un reset maître.

ERREUR DE PARITE (bit 6, PE) : Le drapeau d'erreur de parité indique que le nombre de 1 reçus ne coïncide pas avec la parité paire ou impaire sélectionnée. Une parité paire correspond à un nombre pair de 1 dans le caractère, une parité impaire à un nombre impair, le bit de parité servant à garantir celle-ci. Le drapeau erreur de parité reste à 1 tant que la donnée n'a pas été lue dans le registre des données reçues. Si aucune parité n'a été choisie, le générateur de parité à l'émission et le contrôleur de parité à la réception sont tous deux inactifs. Aucune erreur de parité ne peut alors se produire.

DEMANDE D'INTERRUPTION (bit 7, IRQ) : Le bit IRQ reflète l'état de la ligne d'interruption IRQ. Toute condition d'interruption non inhibée par le registre de contrôle positionne donc ce bit à 1. Le bit IRQ est remis à 0 par une lecture du registre des données reçues ou une écriture vers le registre d'émission des données.





## LE MK 68901

## PERIPHERIQUE MULTI-FONCTIONS

CARACTERISTIQUES
1. INTRODUCTION
2. INTERRUPTIONS
3. TIMERS
4. LA FONCTION USART
5. SPECIFICATIONS ELECTRIQUES

## LE MK 68901

## PERIPHERIQUE MULTI-FONCTIONS

**CARACTERISTIQUES**

1. 8 broches d'entrée/sortie

   * individuellement programmables en entrée ou en sortie.

   * chacune pouvant être source d'interruption.
     - sélection de front actif programmable

2. contrôleur d'interruption 16 sources

   * 8 sources internes

   * 8 sources externes

   * individuellement validables

   * individuellement masquables

   * mode d'interruption programmable
     . par scrutation
     . par vectorisation
     . attente de service possible

   * daisy_chain (chaînage en série)

3. quatre timers avec pré-division individuellement programmables

   * deux timers multimodes
     . mode délai
     . mode mesure d'impulsions
     . mode compteur d'événements

   * deux timers mode délai

   * entrée d'une horloge externe

   * possibilité de sortie de temporisation





4. Un canal USART

   * liaison full duplex

   * asynchrone à 62.5 kbps

   * synchrone par octet à 1 Mbps

   * génération signal d'horloge interne/externe

   * signaux de transfert DMA

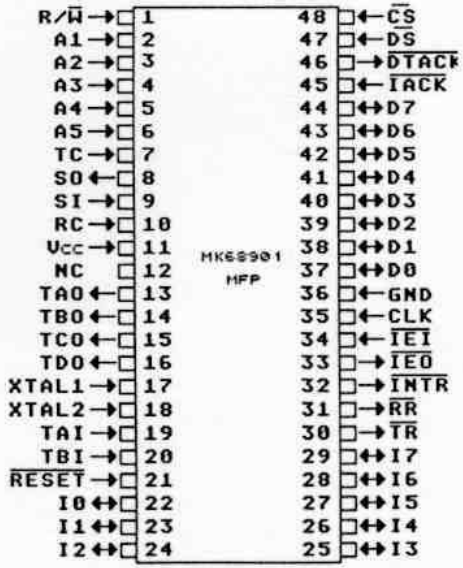
   * contrôle de Modem

   * mode rebouclage

5. Compatible bus 68000

6. Boitier 48 broches

BROCHAGE DU MK68901
Figure 1.



## 1. INTRODUCTION

Le MK68901 MFP (Multi-Function Peripheral) regroupe de nombreuses fonctions de gestion de périphériques nécessaires à un système basé sur un microprocesseur de la famille 68000, à savoir:

huits lignes d'entrée/sortie parallèles

un contrôleur d'interruption pour 16 sources

quatre timers

un canal USART (gestion d'entrée sortie)

L'emploi du MFP peut réduire d'une manière significative le nombre de circuits, et de ce fait le cout du système. Le MFP est entièrement compatible avec le bus 68000. Les 24 registres internes directement adressables fournissent l'interface de contrôle et d'état nécessaire au programmeur.

Le MFP est un dérivé du MK 3801 STI, un périphérique de la famille Z80.

## DESCRIPTION DES BROCHES

| | |
|---|---|
| GND | masse |
| Vcc | +5 volts (+- 5%) |
| CS | sélection de circuit (entrée valide à l'état bas). Sert à sélectionner l'accès aux registres internes du MK68901. CS et IACK ne doivent pas être affirmés en même temps. |
| DS | Entrée des données, active à l'état bas. Sert dans les fonctions de reconnaissance d'interruption et d'accès au circuit. |
| R/W | Lecture/écriture (entrée). R/W est le signal du bus qui indique si le cycle de bus courant est une lecture (haut) ou une écriture (bas). |
| DTACK | Acceptation de transfert de données. Sortie trois état active à l'état bas. Sert à signaler sur le bus que la donnée est prête, ou que la donnée a été prise en compte par le MK68901. |
| A1-A5 | Bus d'adresses en entrée. Le bus d'adresses sert à adresser un des registres internes pendant un cycle de lecture ou d'écriture. |





| | |
|---|---|
| D0-D7 | Bus de données bidirectionnel trois états. Le bus de données sert au transfert de données entre le bus et l'un des registres internes pendant un cycle de lecture ou d'écriture. Il sert aussi à passer un vecteur lors d'un cycle de reconnaissance d'interruption. |
| CLK | Entrée du signal d'horloge. Cette entrée fournit au MK68901 MFP sa base de temps. |
| RESET | Remise à zéro. Entrée active à l'état bas. Le reset désactive le récepteur et le émetteur de l'USART, arrête tous les timers et force les sorties timers à l'état bas. Il désactive tous les canaux d'interruptions et nettoie toutes les interruptions en attente. Les lignes générales d'entrée/sortie (GPIP) sont placée en mode 'trois états'. Tous les registres internes (sauf le timer, les registres de données USART, et le registre d'état de transmission) sont mis à zéro. |
| INTR | Demande d'interruption. Sortie active à l'état bas. INTR est positionné lors d'une demande d'interruption du MK68901. INTR est invalidé pendant le cycle de reconnaissance d'interruption ou en mettant à zéro, par programme, les interruptions en attente. |
| IACK | Acquittement d'interruption. Entrée active à l'état bas. IACK sert à signaler au MK68901 que le processeur est en train de reconnaître une interruption. CS et IACK ne doivent pas être positionnés en même temps. |
| IEI | Entrée de validation d'interruption, active à l'état bas. IEI sert à signaler au MK68901 qu'aucun périphérique ayant une priorité supérieure ne réclame le service d'interruption. |
| IEO | Sortie de validation d'interruption, active à l'état bas. IEO sert à signaler aux périphériques ayant une priorité basse que, ni le MK68901, ni d'autres périphériques ayant une plus haute priorité ne demandent une interruption. |
| I0-I7 | Lignes d'entrée/sortie à usage général. Ces lignes peuvent servir comme sources d'interruptions externes, leur front actif étant programmable, ou comme port d'entrée/sortie. Un registre de direction de données est utilisé pour définir quelles lignes servent en entrée haute impédance et quelles lignes servent en sortie compatible TTL. |
| SO | Sortie série de l'USART |

SI — Entrée série de l'USART.

RC — Horloge de réception des données, contrôle le Timing de réception des données de l'USART.

TC — Horloge de transmission, contrôle la transmission de données de l'USART.

RR — Récepteur prêt. Sortie active à l'état bas. Reflète l'état de l'indicateur 'Récepteur plein' du port numéro 15 pour les opérations DMA.

TR — Transmetteur prêt. Sortie active à l'état bas. Reflète l'état de l'indicateur 'Emetteur prêt' du port numéro 16 pour les opérations DMA.

TAO, TBO
TCO, TDO — Sortie des timers A, B, C, D. Chaque timer pouvant produire un signal carré. La sortie changera d'état à chaque cycle du timer, ainsi une période complète est égale à deux cycles du timer. TAO et TBO peuvent être mis à zéro en écrivant respectivement dans TACR et TBCR (registres de contrôle des timers).

XTAL1
XTAL2 — Entrées d'une horloge externe. Un quartz peut être connecté à XTAL1 et XTAL2 ou une horloge externe TTL peut être reliée à XTAL1. Dans ce cas XTAL2 doit être en l'air. Lors de l'emploi d'un quartz, des condensateurs sont nécessaires. Tous les accès circuits sont indépendants de l'horloge. (la fréquence du quartz monté sur le ST est de 2.4576 MHz).

TAI, TBI — Entrées des timers A et B. Les entrées sont utilisées en modes de fonctionnement, comptage d'événements et mesure de largeur d'impulsions. Les canaux d'interruption I4 et I3 sont respectivement affectés à TAI et TBI. Cependant I4 et I3 peuvent servir pour les entrées/sorties lors du fonctionnement en mode mesure de largeur d'impulsions.

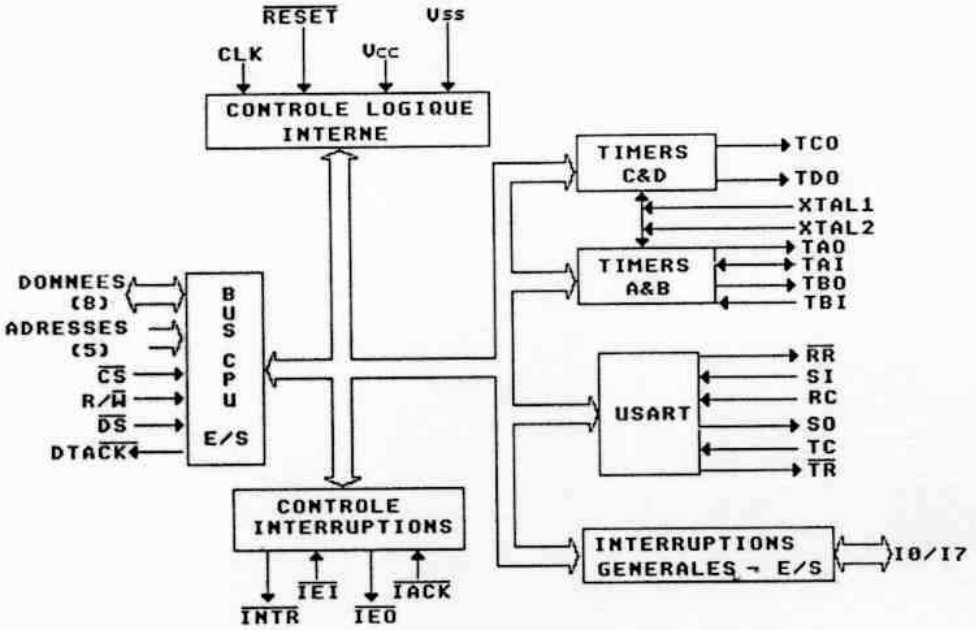




CARTE DES REGISTRES
Figure 1.

| Adresse n° port | Abrévation | Nom des registres |
|---|---|---|
| 0 | GPIP | REGISTRE GENERAL E/S |
| 1 | AER | REGISTRE SELECTION du FRONT ACTIF |
| 2 | DDR | REGISTRE de DIRECTION DES DONNEES |
| 3 | IERA | REGISTRE de VALIDATION INTERRUPTIONS (A) |
| 4 | IERB | REGISTRE de VALIDATION INTERRUPTIONS (B) |
| 5 | IPRA | REGISTRE d' INTERRUPTIONS en ATTENTE (A) |
| 6 | IPRB | REGISTRE d' INTERRUPTIONS en ATTENTE (B) |
| 7 | ISRA | REGISTRE d' INTERRUPTIONS en SERVICE (A) |
| 8 | ISRB | REGISTRE d' INTERRUPTIONS en SERVICE (B) |
| 9 | IMRA | REGISTRE MASQUE des INTERRUPTIONS (A) |
| A | IMRB | REGISTRE MASQUE des INTERRUPTIONS (B) |
| B | VR | REGISTRE VECTEUR |
| C | TACR | REGISTRE de CONTROLE du TIMER A |
| D | TBCR | REGISTRE de CONTROLE du TIMER B |
| E | TCDCR | REGISTRE de CONTROLE des TIMERS C ET D |
| F | TADR | REGISTRE de DONNEES du TIMER A |
| 10 | TBDR | REGISTRE de DONNEES du TIMER B |
| 11 | TCDR | REGISTRE de DONNEES du TIMER C |
| 12 | TDDR | REGISTRE de DONNEES du TIMER D |
| 13 | SCR | REGISTRE CARACTERE de SYNCHRONISATION |
| 14 | UCR | REGISTRE de CONTROLE GENERAL d'USART |
| 15 | RSR | REGISTRE d'ETAT du RECEPTEUR |
| 16 | TSR | REGISTRE d'ETAT du TRANSMETTEUR |
| 17 | UDR | REGISTRE de DONNEES de L'USART |

DIAGRAMME DU MK68901
Figure 2.




## 2. INTERRUPTIONS

Le port général d'interruption entrée/sortie (GPIP) fournit huit lignes qui peuvent être utilisées soit en entrée, soit en sortie, sous contrôle du programme. De plus, chaque ligne peut générer une interruption sur un front montant ou sur un front descendant.

Le GPIP possède trois registres associés. Un regitre programmable sert à définir le front actif pour chaque ligne (un bit par ligne) qui déclenchera une interruption. Un autre registre spécifie la direction des données pour chaque ligne. Le troisième registre est le vrai registre d'entrée/sortie qui sert au transfert des données sur le port. (voir figure 3).





REGISTRES GENERAUX D'ENTREE/SORTIE
Figure 3

REGISTRE de SELECTION du FRONT ACTIF

| | GPIP 7 | GPIP 6 | GPIP 5 | GPIP 4 | GPIP 3 | GPIP 2 | GPIP 1 | GPIP 0 |
|---|---|---|---|---|---|---|---|---|
| PORT 1 (AER) | | | | | | | | |

(1 = MONTANT, 0 = DESCENDANT)

REGISTRE de DIRECTION des DONNEES

| | GPIP 7 | GPIP 6 | GPIP 5 | GPIP 4 | GPIP 3 | GPIP 2 | GPIP 1 | GPIP 0 |
|---|---|---|---|---|---|---|---|---|
| PORT 2 (DDR) | | | | | | | | |

(1 = SORTIE, 0 = ENTREE)

REGISTRE GENERAL D'ENTREE/SORTIE DES DONNEES

| | GPIP 7 | GPIP 6 | GPIP 5 | GPIP 4 | GPIP 3 | GPIP 2 | GPIP 1 | GPIP 0 |
|---|---|---|---|---|---|---|---|---|
| PORT 0 (GPIP) | | | | | | | | |

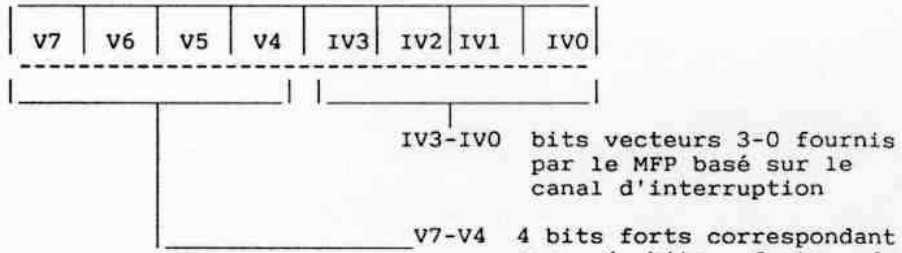
Le registre de sélection du front actif (AER) permet de définir quel sera le front actif sur chacun des registres 'Interruptions Générales' (GPIP). L'écriture d'un 0 sur le bit correspondant de AER autorise l'entrée associée à reconnaître une interruption sur la transition descendante (1 vers 0), alors que l'écriture d'un 1 autorise l'entrée associée à reconnaître une interruption sur la transition montante (0 vers 1). Le bit de front est simplement une entrée sur une porte "OU EXCLUSIF", avec l'autre entrée venant du tampon d'entrée et la sortie allant vers un détecteur de transition 1-0. Cependant selon l'état de l'entrée, l'écriture du bit de sélection de front (AER) peut provoquer une transition qui provoquera une interruption sur le canal associé. Il est sage de dévalider les interruptions avant de configurer AER par l'intermédiaire de IERA et IERB.

Le registre de direction de données DDR (Data Direction Register) sert à définir individuellement le sens d'utilisation de chacune des lignes I0 à I7 (sortie ou entrée). L'écriture d'un zéro sur un bit de DDR positionne la ligne correspondante en entrée Haute Impédance. L'écriture d'un 1 la positionne en sortie "push-pull". Lors de l'écriture de données sur le GPIP, les broches définies comme entrées resteront dans l'état haute impédance, alors que les broches définies comme sorties seront dans l'état (haut ou bas) du bit correspondant dans GPIP. Lorsque GPIP est lu, les données lues proviennent directement des bits correspondants du registre GPIP pour toutes les broches définies commme sorties, alors que les données lues sur les broches définies comme entrées proviendront des tampons d'entrée.

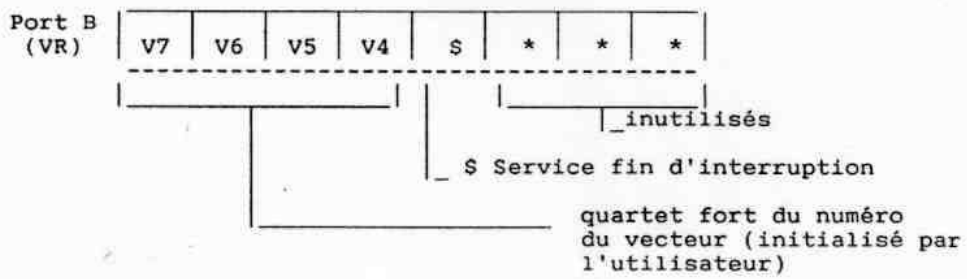
Chaque fonction individuelle du MK68901 dispose d'un vecteur unique d'interruption susceptible de générer une interruption durant un cycle. Le vecteur d'interruption retourné lors du cycle de reconnaissance d'interruption est présenté dans la figure 4 et le registre vecteur dans la figure 5.

Le MK68901 génère d'une manière interne 16 adresses vectorisées, une pour chacun des 16 canaux d'interruptions.

VECTEUR D'INTERRUPTION
Figure 4




REGISTRE VECTEUR
Figure 5




Les registres de contrôle d'interruption (Figure 6) offrent au programmeur la possibilité de contrôler le processus d'interruption de toutes les E/S du MK68901. Ils lui permettent de valider ou d'inhiber n'importe laquelle des 16 interruptions par masquage individuel, et lui donnent accès aux états 'attente de service' et 'en cours de service'. Le mode 'fin d'interruption' est contrôlable par programme. Les niveaux de priorité des interruptions figurent dans le tableau 7.





REGISTRES DE CONTROLE D'INTERRUPTION
Figure 6

REGISTRE de VALIDATION des INTERRUPTIONS

| ADRESSES | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| PORT 3 | A IERA | GPIP 7 | GPIP 6 | TIMER A | RCV Tampon Plein | RCV Erreur | XMIT Tampon Vide | XMIT Erreur | TIMER B |
| PORT 4 | B IERB | GPIP 5 | GPIP 4 | TIMER C | TIMER D | GPIP 3 | GPIP 2 | GPIP 1 | GPIP 0 |

REGISTRE des INTERRUPTIONS en ATTENTE de SERVICE

| | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| PORT 5 | A IPRA | GPIP 7 | GPIP 6 | TIMER A | RCV Tampon Plein | RCV Erreur | XMIT Tampon Vide | XMIT Erreur | TIMER B |
| PORT 6 | B IPRB | GPIP 5 | GPIP 4 | TIMER C | TIMER D | GPIP 3 | GPIP 2 | GPIP 1 | GPIP 0 |

ECRITURE 0 EFFACÉ
ECRITURE 1 INCHANGÉ

REGISTRE des INTERRUPTIONS en COURS de SERVICE

| | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| PORT 7 | A ISRA | GPIP 7 | GPIP 6 | TIMER A | RCV Tampon Plein | RCV Erreur | XMIT Tampon Vide | XMIT Erreur | TIMER B |
| PORT 8 | B ISRB | GPIP 5 | GPIP 4 | TIMER C | TIMER D | GPIP 3 | GPIP 2 | GPIP 1 | GPIP 0 |

REGISTRES MASQUES D'INTERRUPTION

| | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| PORT 9 | A IMRA | GPIP 7 | GPIP 6 | TIMER A | RCV Tampon Plein | RCV Erreur | XMIT Tampon Vide | XMIT Erreur | TIMER B |
| PORT A | B IMRB | GPIP 5 | GPIP 4 | TIMER C | TIMER D | GPIP 3 | GPIP 2 | GPIP 1 | GPIP 0 |

1= NON MASQUÉ 0= MASQUÉ

REGISTRE CONTROLE D'INTERRUPTION
Fig. 7

| Priorité | Canal | Description |
|---|---|---|
| la plus FORTE | 1111 | Ligne d'interruption I7 (GPIP 7) |
| | 1110 | Ligne d'interruption I6 (GPIP 6) |
| | 1101 | Timer A |
| | 1100 | Tampon du récepteur plein |
| | 1011 | Erreur de réception |
| | 1010 | Tampon de émetteur vide |
| | 1001 | Erreur de transmission |
| | 1000 | Timer B |
| | 0111 | Ligne d'interruption I5 (GPIP 5) |
| | 0110 | Ligne d'interruption I4 (GPIP 4) |
| | 0101 | Timer C |
| | 0100 | Timer D |
| | 0011 | Ligne d'interruption I3 (GPIP 3) |
| | 0010 | Ligne d'interruption I2 (GPIP 2) |
| la plus FAIBLE | 0001 | Ligne d'interruption I1 (GPIP 1) |
| | 0000 | Ligne d'interruption I0 (GPIP 0) |

Les interruptions peuvent être traitées soit par **'scrutation'** soit par **'Vectorisation'**. Il est possible de valider ou d'inhiber individuellement chaque canal d'interruption en positionnant à 1 ou à 0 les registres appropriés IERA et IERB (Fig.6). Lorsqu'un canal est 'dévalidé' celui-ci est évidemment totalement inactif. Toute action interne ou externe sur ce canal sera ignorée et toutes les interruptions en attente de service seront annulées, mais cela ne produira aucun effet sur les registre ISRA et ISRB. Ainsi, si les Registres 'Interruptions en Cours de Service' sont utilisés et qu'une interruption est en service sur ce canal lors de sa désactivation, elle restera en service . IERA et IERB peuvent aussi être lus.

Lorsqu'une interruption est reçue sur un canal validé, son bit correspondant dans le registre **'Interruptions en Attente de Service'** (IPRA ou IPRB) est positionné à 1. Quand l'acquittement arrive, le MK68901 passe son numéro de vecteur d'interruption, et le bit correspondant du registre d'interruptions en attente de service est annulé. Il est possible de déterminer si une interruption est en attente de service en lisant IPRA ou IPRB et il est également possible d'annuler une interruption en attente, sans passer par la séquence d'acquittement, en mettant à zéro le bit concerné. Il est possible ainsi de mettre à zéro n'importe quel bit par masquage (écriture de 1 dans tous les bits, sauf pour le bit à annuler). Ainsi un shéma complet de scrutation est possible.

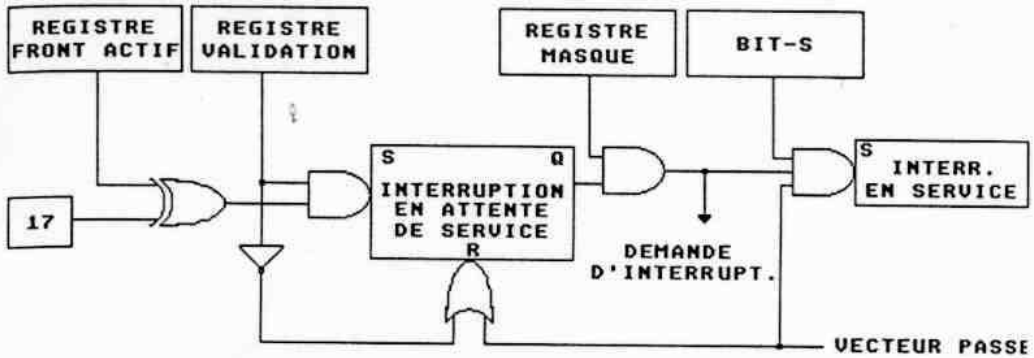
Note: L'écriture d'un 1 dans IPRA/IPRB n'a aucun effet sur le Registre d'Interruptions en Attente de Service.





Les registres **'Masque d'Interruptions'**, IMRA et IMRB, servent à empêcher un canal de solliciter une interruption. Le 'masquage', qui se fait par la mise à 0 du bit correspondant, n'empêche cependant pas le canal de recevoir une interruption (si le canal est valide), celle-ci est alors mémorisée dans le bit correspondant du registre 'Interruptions en Attente de Service', mais aucune demande d'interruption n'est générée par le canal. Si le canal fait une demande d'interruption à l'instant même où le bit correspondant du masque est mis à 0, la demande sera annulée et si aucun autre canal ne fait une demande, INTR sera mis en position inactive. Lorsque le canal est démasqué il peut alors agir librement à moins d'être bloqué par un canal d'interruption de priorité supérieure. Il est possible aussi de lire IMRA et IMRB. La figure 8 montre un circuit type d'un canal d'interruption.

CIRCUIT D'UN CANAL D'INTERRUPTION
Fig.8




Il existe deux modes **'Fin de Traitement d'Interruption'**:

- Le mode automatique
- Le mode programme

Le choix s'effectue par l'intermédiaire du bit S du registre vecteur (VR). Si ce bit est positionné (à 1), le mode programme est sélectionné, sinon le mode automatique est sélectionné et un reset est fait sur tous les bits 'en cours de service'. En mode automatique, le bit 'en attente de service' est mis à 0 lorsque le canal envoie son vecteur et il ne reste alors aucune trace de cette interruption dans le MK68901. En mode programme, le bit 'en cours de service' est positionné et le bit 'en attente de service' est mis à 0 lorsque le canal transmet son vecteur. Lorsque le bit 'en cours de service' est affirmé, aucun canal de priorité inférieure ne peut demander une interruption ou envoyer son vecteur durant un cycle d'acquittement, mais il positionnera son bit 'attente de service'. Par contre un canal ayant une priorité supérieure pourra encore demander une interruption et être pris en compte.

Le bit 'en cours de service' d'un canal peut être annulé en positionnat à 0 dans le bit correspondant du Registre d'Interruptions en Cours de Service (ISRA ou ISRB). Cette instruction devra être exécutée juste avant le retour de la fonction, ainsi aucun canal n'ayant une priorité inférieure ne pourra faire une demande de service tant que le traitement de l'interruption ayant une priorité supérieure ne sera terminé. Si le bit 'en cours de service' est positionné et qu'une seconde interruption est demandée, la requête est enregistrée en positionnant à 1 le bit correspondant du Registre d'Interruptions en Attente de Service et elle ne sera pas traitée tant que la première requête n'est pas entièrement terminée. Il est possible à tout moment de lire les registre ISRA et ISRB, par contre seule l'écriture d'un zéro est possible dans ces registres, ce qui permet d'annuler par programmme le bit 'en cours de service' en fin de routine.

Chaque canal d'interruption renvoie lors de sa phase d'acquittement un vecteur sur 8 bits. Les 4 bits forts sont positionnés par recopie des bits 4 à 7 du registre vecteur VR, les 4 bits faibles correspondent au numéro du canal ayant provoqué l'interruption.

La figure 7 décrit la priorité des 16 canaux d'interruptions. Ainsi l'Interruption Générale numéro 7 a la priorité maximale, alors que l'Interruption Générale numéro 0 a la priorité minimale. La priorité de ces interruptions peut être modifiée par masquage sélectif sous contrôle d'un programme. Les nombres binaires sous 'canal' (fig. 7) correspondent respectivement aux bits IV3, IV2, IV1 et IV0 du 'Vecteur d'Interruption' pour chacun des canaux (figure 4).

Chaque canal a: un bit de validation dans IERA ou IERB, un drapeau d'attente de traitement d'interruption dans IPRA ou IPRB, un bit de masque dans IMRA ou IMRB, et un drapeau de 'traitement d'interruptions en cours' dans ISRA ou ISRB. De plus, les huit lignes d'Interruptions Générales ont: un bit de front dans le Registre de Sélection de Front Actif (AER) , un bit définissant l'état de la ligne (entrée ou sortie) dans le Registre de Direction des Données (DDR) et un bit d'entrée/sortie dans le Registre Général d'Entrée/Sortie de Données (GPIP).

## 3. TIMERS

Le MK68901 possède 4 Timers. Deux d'entre eux, le Timer A et le Timer B, sont des timers complets qui permettent de

- réaliser une temporisation,
- compter des événements,
- mesurer des impulsions,
- générer des signaux carrés.





Les deux autres timers, le Timer C et le Timer D, ne peuvent que réaliser des temporisations programmables. Ils peuvent servir d'horloge pour les transmission de l'USART.
Tous les Timers sont en fait des compteurs avec pré-division, branchés en entrée sur les entrée d'horloge XTAL1 et XTAL2.

Les quatre timers sont programmables via trois Registres de Contrôle Timer et quatre Registres de Données Timer. Les Timers A et B sont respectivement contrôlés par les registres de contrôle TACR et TBCR (fig. 9) et à travers les Registres de Données TADR et TBDR (fig. 10). Les Timers C et D sont contrôlés à travers le registre de contrôle TCDCR (fig. 11) et par deux registres de données, TCDR et TDDR. Les registres de contrôle permettent de sélectionner le mode de fonctionnement et la valeur du pré-diviseur alors que les registres de données servent à la lecture du Timer ou comme tampon lors d'une écriture. Les broches d'entrée TAI et TBI servent aux Timers A et B pour les modes comptage d'événements et mesure d'impulsions.

Lorsque le timer est arrêté, aucun comptage n'est fait. Le contenu du compteur ne sera pas décrémenté pendant un arrêt (à moins qu'il ne soit rechargé en écrivant dans le Registre de Données du Timer), mais le contenu résiduel du pré-diviseur sera perdu.

REGISTRES DE CONTROLE DU TIMER A ET B
Figure 9

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Port C (TACR) | * | * | * | RESET TIMER A | AC3 | AC2 | AC1 | AC0 |
| | * | * | * | RESET TIMER B | BC3 | BC2 | BC1 | BC0 |

| C3 | C2 | C1 | C0 | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | timer stoppé |
| 0 | 0 | 0 | 1 | mode délai; 4 pré-divisions |
| 0 | 0 | 1 | 0 | mode délai; 10 pré-divisions |
| 0 | 0 | 1 | 1 | mode délai; 16 pré-divisions |
| 0 | 1 | 0 | 0 | mode délai; 50 pré-divisions |
| 0 | 1 | 0 | 1 | mode délai; 64 pré-divisions |
| 0 | 1 | 1 | 0 | mode délai; 100 pré-divisions |
| 0 | 1 | 1 | 1 | mode délai; 200 pré-divisions |
| 1 | 0 | 0 | 0 | mode compteur d'événements |
| 1 | 0 | 0 | 1 | génération de signaux; 4 pré-divisions |
| 1 | 0 | 1 | 0 | génération de signaux; 10 pré-divisions |
| 1 | 0 | 1 | 1 | génération de signaux; 16 pré-divisions |
| 1 | 1 | 0 | 0 | génération de signaux; 50 pré-divisions |
| 1 | 1 | 0 | 1 | génération de signaux; 64 pré-divisions |
| 1 | 1 | 1 | 0 | génération de signaux; 100 pré-divisions |
| 1 | 1 | 1 | 1 | génération de signaux; 200 pré-divisions |

* bits inutilisés, lus comme des zéros

REGISTRES DES DONNEES TIMER (A,B,C,D)
Figure 10

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Port F (TADR) | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| Port 10(TBDR) | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| Port 11(TCDR) | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| Port 12(TDDR) | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

REGISTRE DE CONTROLE DES TIMERS C ET D
Figure 11

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Port E (TCDC) | * | CC2 | CC1 | CC0 | * | DC2 | DC1 | DC0 |

| C2 | C1 | C0 | |
|---|---|---|---|
| 0 | 0 | 0 | Timer stoppé |
| 0 | 0 | 1 | Mode délai; pré-division par 4 |
| 0 | 1 | 0 | Mode délai; pré-division par 10 |
| 0 | 1 | 1 | Mode délai; pré-division par 16 |
| 1 | 0 | 0 | Mode délai; pré-division par 50 |
| 1 | 0 | 1 | Mode délai; pré-division par 64 |
| 1 | 1 | 0 | Mode délai; pré-division par 100 |
| 1 | 1 | 1 | Mode délai; pré-division par 200 |

* bits inutilisés; lus comme des zéros

En **mode délai**, le pré-diviseur est toujours actif. Une impulsion de comptage sera envoyée sur le timer principal à chaque fois que le nombre de cycles horloge prescrit sera écoulé. Ainsi, si le pré-diviseur est programmé pour une division par dix, une impulsion sera envoyé au timer principal tous les dix cycles d'horloge.





Le compteur principal est initialisé en écrivant dans le Registre de Données Timer. A chaque impulsion qu'il recevra, le compteur principal sera décrémenté. Lorsque le compteur arrivera à la valeur '01', il ne passera pas à '00' mais sera automatiquement rechargé avec le contenu du Registre de Données Timer. De plus une impulsion TIME OUT est générée, laquelle provoque une interruption si le canal du timer est valide (autorisé et non masqué) et la broche de sortie du timer change d'état logique (TAO, TBO, TCO, TDO). Elle restera dans cet état jusqu'à la prochaine impulsion TIME OUT. La sortie fera, ainsi, un cycle complet toutes les deux impulsions TIME OUT.

Si, par exemple, le pré-diviseur a été programmé pour une division par 10, et que le Registre de Données Timer a été chargé avec 100 (décimal), le compteur principal sera décrémenté toutes les dix périodes d'horloge, une impulsion TIME OUT sera générée toutes les 1000 périodes, et le signal de sortie sera de 2000 périodes.

Le compteur principal est un COMPTEUR 8 BITS. Il peut être lu à tout moment (lecture du Registre de Donnée Timer TDR). L'information lue est celle qui a été enregistrée lorsque le signal DS était à l'état haut, avant le cycle actuel de lecture. L'écriture se fait en chargeant le Registre de Données Timer et le compteur principal, si le timer est arrêté. Dans ce cas, il peut arriver que la donnée écrite dans TDR ne soit pas correctement transférée dans le registre de données Timer. Cela provoquera l'inexactitude de la première séquence de comptage, mais non des autres. Il sera donc conseillé de vérifier que la donnée a été correctement chargée en faisant une lecture. Tant que celle-ci ne sera pas bonne, une boucle devra être réalisée.

Si le Registre de Données Timer est écrit alors que le Timer n'est pas arrêté, le nouveau mot ne sera chargé que lors du passage à '01' du compteur.

Si une valeur est écrite lorsque le compteur est à '01', la valeur chargée sera indéterminée. Il est donc conseillé, avant toute écriture, de vérifier que le compteur n'est pas à 1.

Si le compteur principal est chargé avec la valeur '01', une impulsion TIME OUT sera générée à chaque impulsion envoyée au compteur principal. S'il est chargé avec la valeur '00' une impulsion TIME OUT sera générée toutes les 256 impulsions de compteur.

S'il y a modification du pré-diviseur en cours de comptage, la première impulsion TIME OUT qui suivra se produira à un moment indéterminé compris entre 1 et 200 fois le nombre de cycle horloge correspondant à la valeur du tampon. Les impulsions TIME OUT suivantes seront correctes.

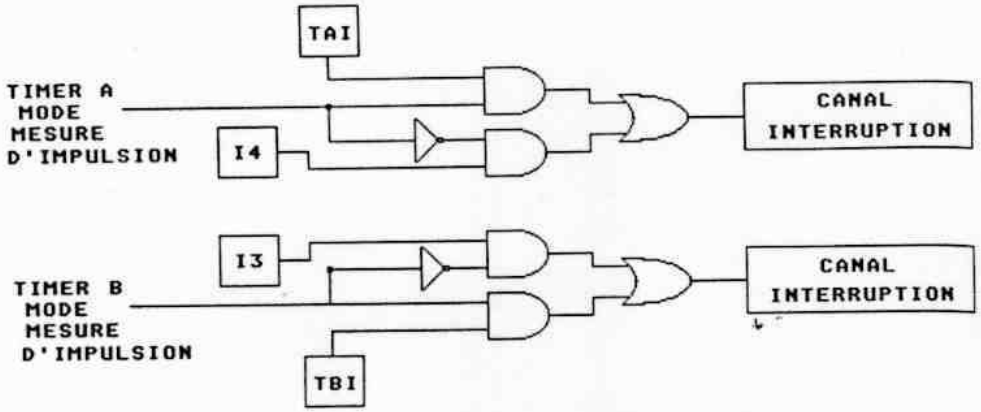
N.D.T.: Sur le ST, les Timers C et D sont utilisés avec une pré-division de 16.

En plus du mode délai, décrit ci-dessus, les Timers A et B peuvent fonctionner selon deux autres modes:

- mesure d'impulsions,
- compteur d'événements.

Ces deux modes nécessitent un signal de contrôle auxiliaire: L'entrée TAI associée au canal d'interruption I4 et l'entrée TBI associée au canal I3.

CIRCUIT TYPE DES TIMERS MFP EN MODE
MESURE D'IMPULSION
Figure 12




Le mode **mesure d'impulsions** ressemble beaucoup au mode délai. Cependant, dans ce mode, le signal de contrôle auxiliaire sur TBI et TAI agit comme une entrée de validation sur le timer. Si l'entrée TAI (ou TBI) est inactive, le timer est figé. Si elle est active, le compteur et le pré-diviseur travaillent. Ainsi la mesure d'une impulsion sur TAI ou TBI est déterminée par le nombre de comptes timer qui se produisent pendant le temps où le timer est autorisé à travailler.

L'état actif du signal sur TAI ou TBI est dépendant du bit de Sélection de Front du Canal d'Interruption associé (GPIP 4 pour TAI et GPIP 3 pour TBI; voir le Registre de Sélection du Front Actif, AER, figure 3). Si le bit de front actif associé est à 1, l'entée TAI ou TBI sera active à l'état haut. Inversement si le bit est à 0, l'entrée TAI ou TBI sera active à l'état bas.

Le canal d'interruption associé au Timer (I3 ou I4) fonctionne encore en mode mesure d'impulsion, cependant les interruptions provoquées par les transitions sur les entrées TAI ou TBI sont générées sur des transitions contraires.





Par exemple, si le bit de sélection de front actif associé à TAI (AER4) est à 1, une interruption devrait être normalement être générée suite à une transition montante (0-1) sur l'entrée I4. Si le timer associé à cette entrée est programmé en mode 'mesure de largeur d'impulsion', l'interruption ne se produira pas sur le front montant, mais sur le front descendant du signal injecté sur l'entrée TAI. En effet, comme l'entrée timer est active à l'état haut (AER4 = 1), le Timer A comptera pendant que cette entrée est à l'état haut.

Lorsque l'entrée TAI passe de l'état haut à l'état bas, le Timer A s'arrête, et l'interruption sera générée (après s'être assuré que le canal est valide). Ceci afin de permettre au CPU de signaler que l'opération de mesure est terminée. Le Timer A peut alors être lu, pour connaître la valeur de la largeur de l'impulsion.

Il est à remarquer que les entrées I4 et I3, associées aux Timers A et B, peuvent toujours être utilisées en entrée/sortie lorsque les Timers A et B sont utilisés en mode 'mesure de largeur d'impulsions'.

Si le Timer est reprogrammé dans un autre mode, les interruptions seront de nouveaux générées sur le front défini normalement.

ATTENTION: Le fait de changer le bit de sélection de front, en passant en mode 'mesure de largeur d'impulsions' ou en sortant de ce mode peut provoquer une interruption sur le canal associé.

Si l'on désire faire plusieurs mesures de largeur d'impulsions consécutives, il faut lire le contenu du timer, puis ré-initialiser le compteur principal (Timer) en écrivant la valeur lue sur le registre de données du timer. Si la donnée est écrite dans le registre de données à l'instant où l'entrée est activée, la valeur chargée dans le registre de données est indéterminée.

Si le Timer est écrit une fois l'entrée activée, le Timer compte à partir de la valeur précédente, et la valeur correcte n'est chargée dans le Timer que lorsque le compte arrive à la valeur '01'. Le compte réalisé avant que le Timer soit chargé est compris dans la mesure de la largeur d'impulsion.

Dans le mode **'comptage d'événements'**, le pré-diviseur est invalide. Chaque fois que l'entrée de contrôle (TAI,TBI) fait une transition, comme définie par le bit correspondant au canal d'interruption dans le Registre de Sélection du Front Actif, une impulsion est générée et le compteur est décrémenté. Excepté cette différence, le Timer fonctionne commme décrit précédemment.

Le fait de modifier un bit du Registre de Sélection du Front Actif peut provoquer une interruption puisque le canal d'interruption associé à l'entrée (I3 pour TBI, I4 pour TAI) fonctionne normalement.

Pour compter les événements de façon sûre, l'entrée doit rester dans un état stable pendant un temps égal au moins à 4 périodes d'horloge. Il est ainsi possible de compter des signaux d'une fréquence allant jusqu'à 1/4 de la fréquence horloge.

La manière dont la sortie change d'état a déjà été préalablement traitée.

Un RESET matériel provoquera la mise à l'état bas de toutes les sorties (TAO, TBO, TCO, TDO).

Les sorties du Timer A et du Timer B changeront d'état à chaque impulsion TIME OUT quelque soit le mode dans lequel se trouve le Timer. De plus les sorties du Timer A et du Timer B peuvent être à tout moment forcées à l'état bas par la mise à 1 du bit de RESET qui leur est associé dans les Registres de contrôle TACR et TBCR.

ex: Pour figer TBO à l'état bas, il faut mettre TACR3 à 1.

Les sorties seront mises à l'état bas durant la période d'écriture et à la fin de cette opération la sortie sera apte à échanger à nouveau d'état si une impulsion TIME OUT survient. Cette configuration permet la génération de signaux carrés.

Durant un reset, les Registres de Données Timer et les compteurs principaux ne sont pas remis à zéro.

## 4. LA FONCTION USART

La fontion de communication série du MK 68901 est fournie par un USART full-duplex à deux tampons qui autorise les communications synchrones et asynchrones. Lors de communications asynchrones, il est possible de configurer par programme la taille des mots ainsi que les bits de start et de stop. Pour les communications synchrones, un caractère de synchronisation est fourni durant les opérations de réception. Ce caractère sera également transmis d'une façon répétitive lorsqu'il n'y a pas de donnée à émettre. De plus le MK68901 permet de retirer les caractères de synchronisation des données reçues durant une transmission synchrone.

Les lignes de contrôle RR (récepteur prêt) et TR (émetteur prêt) servent aussi à des opérations DMA.

Il est possible d'avoir des horloges de réception et d'émission différentes ainsi que des états de réception et d'émission différents, ce qui rend tout à fait indépendantes les opérations d'émission et de réception.





L'USART comprend un Registre de Contrôle, deux Registres d'Etat, et un Registre de Données (UDR). La forme du Registre de Données est représenté dans la figure 13. Le Registre de Contrôle sert à définir les caractéristiques de fonctionnement (fig. 14).

Les registres d'état (RSR et TSR) servent à définir les caractéristiques du récepteur et de l'émetteur (fig.15 et 16).

REGISTRE DE DONNEES USART (UDR)
Figure 13

| | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|
| port 17 (UDR) | | | | | | | | |

REGISTRE DE CONTROLE USART (UCR)
Figure 14

| | UCR7 | | | | | | UCR0 |
|---|---|---|---|---|---|---|---|
| Port 14 | 1=1/16<br>0=1/1 | WL1 | WL0 | ST1 | ST0 | valid. de la parité | 1=pair<br>0=imp. | * |

* bit inutilisé, toujours lu comme 0

UCR7 Mode de division de l'horloge.
Division par 1 (si bit à 0) ou par 16 (si bit à 1) de l'horloge de transmission ou de réception.

WL0-WL1 Contrôle de la longueur du mot utilisé
WL0 et WL1 déterminent le nombre de bits utilisés pour une donnée (bits de stop, de start et de parité non compris)

| WL1 | WL0 | |
|---|---|---|
| 0 | 0 | 8 bits |
| 0 | 1 | 7 bits |
| 1 | 0 | 6 bits |
| 1 | 1 | 5 bits |

ST0-ST1 Format de contrôle
Ces deux bits servent à définir le format de contrôle de transmission.

| | ST1 | ST0 | Bit de Start | Bit de stop | Format |
|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | SYNCHRONE |
| | 0 | 1 | 1 | 1 | ASYNCHRONE |
| * | 1 | 0 | 1 | 11/2 | ASYNCHRONE |
| | 1 | 1 | 1 | 2 | ASYNCHRONE |

* uniquement en mode division par 16

UCR2 **Validation de la parité**
Lorsque UCR2 est à '1', la parité sera vérifiée par le récepteur. La parité sera calculée et le bit de parité sera positionné par le émetteur. Lorsque le bit de parité est à '0', aucun test de parité n'est fait et le bit de parité n'est pas placé par le émetteur.

Lorsque la donnée est stockée sur 8 bits, le MFP calcule et insère automatiquement la parité. Pour les données plus courtes, la parité est stockée dans le Registre de Caractère de Synchronisation (SCR) avec le caractère de synchronisation.

UCR1 Mode de parité
Une parité paire sera utilisée lorsque ce bit est à 1, une parité impaire, lorsqu'il est à 0.

Il est à remarquer que le mode synchrone et le mode asynchrone peuvent être utilisés indépendamment du mode de division d'horloge. Ainsi il est possible de synchroniser les données sur un périphérique tout en employant des bits de start et de stop. La donnée sera introduite après un bit de start et un bit de stop sera placé en bout, afin de cadrer dans le format. Inversement, il est possible de chronométrer les données en mode asynchrone en utilisant le format synchrone. Le circuit d'horloge du récepteur est capable de détecter la transition logique de données et de re-synchroniser l'horloge interne à chaque transmission de données. Dans ce mode, toutes les autres configurations fonctionnent normalement. Cette re-synchronisation n'est valide qu'en mode de division d'horloge par 16.

**Le RECEPTEUR de l'USART**

Le récepteur est configuré par le registre de contrôle général (UCR) comme décrit précédemment. L'état du récepteur de l'USART peut être déterminé par la lecture et l'écriture du **Registre d'Etat du Récepteur** (RSR).

Le RSR est configuré comme suit:

REGISTRE D'ETAT DU RECEPTEUR
Figure 15

| | $RSR_7$ | | | | | | | $RSR_0$ |
|---|---|---|---|---|---|---|---|---|
| PORT 15 (RSR) | TAMPON PLEIN | ERREUR DE SURCHARGE | ERREUR DE PARITE | ERREUR DE FORMAT | RECHERCHE BREAK | RECEPT. CARACT. SYNCHRO. | VALIDAT. CARACT. SYNCHRO. | VALIDAT. DU RECEPTEUR |





RSR7 Tampon du récepteur plein. Lorsqu'il est à 1, le tampon récepteur est plein. Le bit est mis à 0 par la lecture du Registre de Données de l'USART ou par un RESET. Ce bit ne peut qu'être lu.

RSR6 Erreur de surcharge. Ce drapeau est positionné à 1 lorsque le caractère est reçu mais qu'il ne peut être transféré dans le registre de données parce que le dernier caractère dans ce registre n'a pas encore été lu. Lorsque cette configuration se présente le nouveau caractère n'écrase pas celui qui se trouve déjà dans le registre de données. Il est à remarquer que le drapeau d'état reflète l'état du caractère de donnée actuellement présent dans le tampon de réception (UDR). Le bit n'est positioné que lorsque le caractère présent a été lu. L'interruption associée à cette erreur ne sera pas générée tant que l'ancien caractère dans le tampon de réception n'a pas été lu.
RSR6 est mis à zéro par la lecture du registre d'état récepteur (RSR) ou par un RESET.

RSR5 Erreur de parité. Ce drapeau est mis à 1 si une erreur de parité est détectée sur le caractère reçu. Le bit est remis à 0 lors de la réception du prochain caractère ayant une bonne parité.

RSR4 Erreur de Format. Ce bit ne sert qu'en mode asynchrone. Il indique une erreur de format lorsqu'il est mis à 1 comme, par exemple, une absence de bit de stop. RSR4 est remis à 0 lors de la réception du premier caractère sans erreur qui suit.

RSR3 Recherche de 'BREAK' en **mode synchrone.**
Lorsqu'il est mis à zéro, le récepteur se met à la recherche de l'octet de synchronisation. Le compteur de taille du caractère est désactivé. Lorsque l'octet est trouvé, RSR3 est automatiquement remis à 1, le compteur de taille du caractère est réactivé et la saisie des caractères recommence. Une interruption sera générée sur le canal Erreur de Réception lorsque l'octet est trouvé. Le caractère de synchonisation n'est pas transféré dans le registre de réception.

'BREAK' en **mode asynchrone.**
Cet indicateur est mis à 1 par le MFP lorsque un BREAK a été reçu (un caractère ne comprenant que des zéros suivi par un bit de stop). Ce drapeau restera dans cet état tant qu'un bit à 1 ne sera pas reçu et que le registre RSR n'aura pas été lu au moins une fois après le positionnement à 1 de ce bit.

RSR2 Réception d'un caractère de synchronisation en mode synchrone.
Ce drapeau est mis à 1 toutes les fois qu'un caractère reçu est le caractère de synchronisation. Il est remis à zéro dès qu'un caractère différent du caractère de synchronisation se présente dans le tampon de réception.

Détection du bit de Start en mode asynchrone.
RSR2 est mis à 1 lors de la détection du bit de start et remis à zéro à la fin du caractère.

RSR1 Acceptation d'un caractère de synchronisation.
Si ce bit est à 1 les caractères de synchronisation ne seront pas chargés dans le tampon de réception et aucun signal 'tampon plein' ne sera généré.

RSR0 Validation du récepteur.
Valide le récepteur si le bit est à 1 et l'inhibe s'il est à 0. Lorsqu'un 0 est écrit dans ce bit, le récepteur se débranche immédiatement et tous les bits du RSR sont mis à 0. Lorsqu'il est remis à 1, le récepteur est validé normalement. L'horloge du récepteur doit être active pour que le récepteur puisse être validé.

Deux canaux d'interruption sont associés au récepteur. Un canal est utilisé pour avertir que le tampon de récepteur est plein (avertissement) alors que l'autre spécifie une erreur de réception. Seule une interruption est générée par caractère reçu, mais aux deux canaux correspondent des vecteurs différents, l'un pour la condition normale, l'autre pour les conditions d'erreur. Les erreurs qui produisent une interruption sur le canal d'erreur sont : erreur de surcharge, erreur de parité, erreur de format, caractère de synchronisation trouvé, et 'break'. Si le canal d'erreur est validé et qu'une erreur se produit, une interruption ne sera générée que sur le canal d'erreur.

A chaque transfert de caractère dans le tampon récepteur les drapeaux de RSR sont positionnés. Aucun drapeau (sauf RSR2), n'est modifié tant que le caractère dans le tampon de réception n'a pas été lu. La lecture de tampon de réception permet l'introduction d'un nouveau caractère reçu. Ainsi l'on doit d'abord lire le registre RSR, puis lire le tampon de réception UDR, et vérifier que la donnée est correcte. Si l'opération est effectuée dans le sens inverse, le positionnement des drapeaux du registre RSR peut correspondre au caractère suivant mis dans le tampon de réception; de plus si une erreur de surcharge se produit, aucune autre donnée ne pourra être transférée dans le registre de données tant que RSR n'aura pas été lu.

Si un 'break' est réceptionné (mode asynchrone) alors que le drapeau d'erreur de surcharge est à 1, le drapeau de 'break' sera lui aussi mis à 1.





Un 'break' génère une interruption quand la condition se produit et également lorsqu'elle prend fin. Si la condition de break prend fin avant sa reconnaissance, par une lecture de RSR, l'interruption de fin de break ne sera envoyée qu'après la lecture de RSR.

Chaque fois que le format asynchrone est sélectionné, la détection du bit de start est valide. Si une division par 16 de l'horloge a été choisie en format asynchrone, la détection du faux bit de start est aussi valide. Une transition valide doit rester stable au moins pendant 3 fronts d'horloge récepteur positifs. Pour qu'un bit de start soit considéré comme valide, une transition 0-1 ne doit pas se produire pendant 8 transitions d'horloge après la première transition valide 1-0.

Après la détection d'un bit de start valide, les transitions valides sont constamment testées. Lorsqu'une transition valide est détectée, le compteur est forcé à l'état zéro et aucun autre test de transition n'est fait tant que le compteur n'est pas à 4. Quand le compteur est à 8, un test logique est fait dans le récepteur.

Une conséquence de cette logique de re-synchronisation est qu'il est possible de travailler en mode asynchrone sans bit de start ni de stop, s'il y a suffisamment de transitions valides dans le flux des données. Cela permet aussi d'être plus tolérant avec l'horloge pour des communications asynchrones qu'avec un périphérique qui n'emploierait qu'une synchronisation avec des bits de start.

## Le TRANSMETTEUR de l'USART

L'état du émetteur peut être déterminé par la lecture ou l'écriture du Registre d'Etat du Transmetteur (TSR).

REGISTRE D'ETAT DU TRANSMETTEUR
Figure 16

| | $TSR_7$ | | | | | | | $TSR_0$ |
|---|---|---|---|---|---|---|---|---|
| PORT 16 (TSR) | TAMPON VIDE | ABSENCE DE CARACT. A TRANSM. | VALIDAT. AUTOMAT. DU RECEPT. | FIN DE TRANSMIS. | TRANSMIS. D'UN BREAK | HAUT | BAS | VALIDAT. DU TRANSMET. |

TSR7 Tampon vide.
Ce bit de d'état est positionné à 1 lorsque le caractère stocké dans le tampon du émetteur est transféré dans son registre à décalage de sortie. Le tampon peut alors être rechargé avec le caractère suivant. Ce drapeau est remis à 0 lors du rechargement d'un caractère dans le tampon (en écrivant dans le registre UDR)

TSR6 Absence de caractère dans le tampon du émetteur.
Ce drapeau est mis à 1 lorsqu' un caractère placé dans le registre à décalage a été entièrement transmis et qu'aucun caractère n'a été rechargé dans le tampon du émetteur.

Ce bit peut être remis à 0 soit en lisant le registre TSR, soit en inhibant le émetteur. Après son positionnement à 1, ce bit ne peut être remis à 0 par lecture avant un cycle horloge émetteur complet. Le timing de certains systèmes autorise une lecture du registre TSR avant que le cycle complet d'horloge n'ait été exécuté, ce qui a pour conséquence de ne pas permettre la mise à zéro du bit avant la prochaine lecture. Une lecture factice du registre TSR peut être faite à la fin du cycle afin d'éviter ce genre de problème.

TRS5 Validation automatique du récepteur.
Lorsque ce drapeau est mis à 1, le récepteur est validé automatiquement (s'il était inhibé) en fin de transmission du dernier caractère. TRS5 est remis à 0 à la fin de la transmission.

TSR4 Fin de transmission.
Lorsque le émetteur est inhibé et qu'un caractère se trouve encore dans le registre à décalage, la transmission continuera jusqu'à la transmission complète de ce caractère. Une fois le caractère tranmis, TSR4 est positionné. Si aucun caractère n'est en attente de transmission lors de l'inhibition du émetteur, celui-ci sera arrêté lors du front montant de l'horloge interne qui suit.
Ce drapeau est remis à 0 par la validation du émetteur.

TSR3 Emission d'un BREAK
La mise à 1 de ce bit provoque l'émission d'un 'BREAK' une fois le caractère en cours d'émission entièrement transmis, ceci tant que le bit est à 1. L'écriture d'un 0 dans TRS3 fait repartir la transmission normalement. Ce drapeau n'est pas pris en compte en mode synchrone. TSR7(tampon vide) est désactivé lorsque TSR3 est positionné. Ainsi, si un caractère se trouvait dans le tampon lors de la mise à 1 du drapeau de BREAK, il y restera jusqu'à la fin du BREAK. Le caractère sera alors émis (si le émetteur est encore valide). Si le tampon est vide à la fin du BREAK, le drapeau d'absence de caractère à transmettre sera positionné.

Le bit de BREAK ne peut être positionné que lorsque le émetteur est activé et qu'il a eu assez de temps (un cycle horloge) pour faire son reset interne et ses fonctions d'initialisation.





TSR2-TSR1 Etat de la sortie.
Sert à configurer l'état de la sortie du émetteur quand ce dernier est inhibé.

| TSR2 | TSR1 | Etat de la sortie |
|---|---|---|
| 0 | 0 | Haute impédance |
| 0 | 1 | Etat bas |
| 1 | 0 | Etat haut |
| 1 | 1 | Bouclage local. La sortie SO est connectée sur l'entrée SI et TC est branché sur l'horloge de émetteur. Lorsque le émetteur est inhibé la sortie SO est forcée à l'état haut. |

Changer ces deux bits, alors que TRS0 a été positionné, modifiera l'état de la sortie tant que TSR4 sera à 0. Ces bits devront être positionnés avant la validation du émetteur. L'état de ces bits détermine l'état du premier caractère transmis après la validation du émetteur. Si le mode haute impédance a été sélectionné avant que le émetteur n'ait été validé, le premier bit transmis est indéterminé.

TSR0 Validation du émetteur
Ce bit sert à valider (mise à 1) et inhibé (mise à 0) le émetteur. Après une dévalidation, tout caractère se trouvant dans le tampon sera émis intégralement. Si un BREAK est émis quand TSR0 est mis à 0, le émetteur sera déconnecté à la fin du caractère de BREAK et le bit de stop ne sera pas envoyé. L'horloge de transmission doit être active avant que le émetteur soit validé. Un bit à 1 précède toujours le premier caractère émis après une validation du émetteur.

De même que pour le récepteur, deux canaux d'interruption sont associés au émetteur. Le tampon émetteur vide provoque une interruption sur un canal alors qu'une erreur provoquera une interruption sur l'autre canal. Lorsqu'une absence de caractère à transmettre est détectée en format synchrone, le caractère dans SCR sera transmis jusqu'à ce qu'un nouveau soit chargé dans le tampon de transmission. En mode asynchrone, une suite de 1 sera transmise continuellement lorsqu'une absence de caractère se produira.

Le tampon de transmission peut être chargé avant sa validation. Lorsque le émetteur est désactivé, le caractère en cours d'émission sera transmis intégralement, mais le caractère dans le tampon restera dans le tampon et ne sera pas envoyé. Si le tampon était déjà vide, le drapeau 'Tampon Vide' (TSR7) sera positionné et le restera. Si le émetteur est désactivé avec un caractère dans le registre de sortie, mais aucun dans le tampon, une erreur absence de caractère sera générée à la fin de l'envoi.

Il est souvent nécessaire d'envoyer un 'BREAK' pendant quelques périodes. Pour aider le timing durant la transmission d'un 'BREAK', une erreur de transmission sera générée à la fin de la période de temps correspondant à l'envoi d'un caractère normal. Le registre d'état ne sera pas affecté par cette erreur. Il est à remarquer que s'il existe une erreur absence de caractère, le bit correspondant doit être mis à 0 dans TSR, et le Registre d'Interruption en Attente doit être nettoyé avant l'envoi d'un 'BREAK', sinon aucune interruption ne pourra être générée.

Si le format synchrone est sélectionné, le caractère de synchronisation doit être chargé dans le Registre de Caractère de Synchronisation (SCR Fig. 17). Ce caractère est comparé au caractère reçu pendant une recherche de 'break' et sera continuellement envoyé lors de l'absence de caractère à transmettre.

REGISTRE DE CARACTERE DE SYNCHRONISATION
Figure 17

| | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|
| Port 13 (SCR) | | | | | | | | |

Tous les drapeaux de RSR et de TSR continueront à être positionnés normalement même si les canaux d'interrutions qui leurs sont associés sont désactivés.

Les conditions d'erreurs dans l'USART sont déterminées en surveillant le Registre d'Etat du Receveur et le Registre d'Etat du Transmetteur. Ces erreurs portent sur le dernier caractère et ne restent pas positionnées après lecture. Lors de transferts de blocs de données, il est nécessaire de gérer toutes les erreurs afin de les tester à la fin de l'envoi. Il est possible de se servir du contrôleur d'interruptions du MK68901 pour sauver les conditions d'erreur durant la transmission d'un bloc en validant les interrupions d'erreur sur le canal désiré (canal d'erreur Receveur ou Transmetteur). Une fois le transfert terminé, le Registre d'Interruptions en Attente peut être testé afin de déterminer la présence d'une interruption d'erreur en attente.





Les bits inutilisés du caractère de synchronisation sont mis à 0. Toutefois, dans certains cas, la longueur du caractère devra être fixée avant l'écriture du caractère de synchronisation. La longueur du caractère de synchronisation est égale à la longueur d'un caractère normal plus 1 lorsque la parité est valide. L'utilisateur a la charge de déterminer la parité du caractère de synchronisation lorsque la longueur de caractère choisie n'est pas de 8 bits. Le MK68901 n'ajoute pas le bit de parité au caractère de synchronisation si la longueur d'un caractère est inférieure à 8 bits. Le bit supplémentaire dans le caractère de synchronisation est transmis comme étant le bit de parité. Lorsque la longueur de caractère est de huit et que la parité est sélectionnée, la parité du mot du caractère de synchronisation est calculée et ajoutée automatiquement par le MK68901.

RR RECEPTEUR PRET

RR est mis à 1 lorsque le bit 'Tampon Plein' est positionné dans RSR, à moins qu'une erreur de parité ou de format n'ait été détectée par le récepteur.

TR TRANSMETTEUR PRET

TR est mis à 1 lorsque le bit 'Tampon Vide' est positionné dans TSR, à moins qu'un 'break' ne soit en cours de transmission.

ACCES AUX REGISTRES

Tous les accès aux registres sont dépendants du signal d'horloge (CLK), comme décrit dans les chronogrammes. Pour la lecture, CS et DS doivent être affirmés, et R/W doit être haut. Le signal interne de contrôle de lecture est essentiellement une combinaison de CS, DS, et RD/WR. Ainsi une opération de lecture débute quand CS et DS deviennent actifs et finit lorsqu'ils deviennent inactifs. L'adresse sur le bus doit être stable avant le début de l'opération et doit rester stable durant toute celle-ci. A moins qu'une opération de lecture ou un cycle d'acquittement d'interruption ne soit en cours, le bus de donnée (D0-D7) reste en condition 'trois états'.

Pour écrire un registre, CS et DS doivent être affirmés et R/W doit être bas. L'adresse sur le bus doit être stable avant le commencement de l'opération et doit le rester tout le long de celle-ci. Une fois le MK68901 ait été affirmé, le CPU inverse DS. A cet instant le MFP vérrouille le bus de données et écrit le contenu dans le registre approprié. Aussi lorsque DS est inversé, le MFP annule DTACK.

Lors d'un acquittement d'interruption, l'opération débute lorsque IACK est mis à l'état bas, et finit lorsque IACK est remis à l'état haut. Le bus de données est mis en l'état 'trois états' lorsque IACK ou DS sont mis à l'état haut.

## 5. SPECIFICATIONS ELECTRIQUES DU MK68901

### TAUX MAXIMA

| | |
|---|---|
| Températures admises sous tension | -25°C à +100°C |
| Températures admises en stockage | -65°C à +150°C |
| Voltage sur les broches (par rapport à la masse) | -0.3V à +7V |
| Puissance dissipée | 1.5W |

### CARACTERISTIQUES COURANT CONTINU
(TA = 0°C à 70°C; Vcc = +5V+-5%)

| SYM | PARAMETRES | MIN | MAX | UNIT. | CONDITIONS DE TEST |
|---|---|---|---|---|---|
| $V_{IH}$ | Voltage Haut d'Entrée | 2,0 | $V_{cc}$+.3 | V | |
| $V_{IL}$ | Voltage Bas d'Entrée | -0,3 | 0,8 | V | |
| $V_{OH}$ | Volt. Haut en Sortie (sauf $\overline{DTACK}$) | 2,4 | | V | $I_{OH}$ = -120 µA |
| $V_{OL}$ | Volt. Bas en Sortie (sauf $\overline{DTACK}$) | | 0,5 | V | $I_{OL}$= 2.0 mA |
| $I_{LL}$ | Puissance fournie | | 180 | mA | Sorties ouvertes |
| $I_{LI}$ | Pertes de Courant en Entrée | | ±10 | µA | $V_{IN}$ = 0 à $V_{CC}$ |
| $I_{LOH}$ | Pertes en Sortie Haute Imped. | | 10 | µA | $V_{OUT}$ = 2,4 à $V_{CC}$ |
| $I_{LOL}$ | Pertes en Sortie Haute Imped. | | -10 | µA | $V_{OUT}$ = 0,5V |
| $I_{OH}$ | Courant Source en Sortie $\overline{DTACK}$ | | -400 | µA | $V_{OUT}$ = 2,4 |
| $I_{OL}$ | Courant de fin en Sortie $\overline{DTACK}$ | | 5,3 | mA | $V_{OUT}$ = 0,5 |



CARACTERISTIQUES ELECTRIQUEs, COURANT CONTINU
(Vcc= 5.0V +-5%, GND = 0V, TA = 0°C à 70°C)

| N° | PARAMETRES | MK68901 MIN | MK68901 MAX | UNIT. | FIG | NOTE |
|---|---|---|---|---|---|---|
| 1 | temps à l'état haut $\overline{CS}$, $\overline{DS}$ | 50 | | ns | 21-22 | |
| 2 | R/$\overline{W}$, A1-A5 Valides à Chute $\overline{CS}$ (établis.) | 0 | | ns | 21-22 | |
| 3 | Donnée Valide avant montee $\overline{DS}$ (établis.) | 280 | | ns | 21 | |
| 4 | $\overline{CS}$, $\overline{IACK}$ Valide avant Chute CLK(établis.) | 50 | | ns | 21-24 | 3 |
| 5 | CLK bas à $\overline{DTACK}$ bas | | 220 | ns | 21-22 | |
| 6 | $\overline{CS}$, $\overline{DS}$ ou $\overline{IACK}$ haut à $\overline{DTACK}$ haut | | 60 | ns | 21-24 | |
| 7 | $\overline{CS}$, $\overline{DS}$, $\overline{IACK}$ haut à $\overline{DTACK}$ haute imped. | | 100 | ns | 21-24 | |
| 8 | $\overline{CS}$, $\overline{DS}$, $\overline{IACK}$ à données invalides | 0 | | ns | 21-24 | |
| 9 | $\overline{CS}$, $\overline{DS}$, $\overline{IACK}$ haut à données haute imped. | | 50 | ns | 21-24 | |
| 10 | $\overline{CS}$, $\overline{DS}$ haut à R/$\overline{W}$, A1-A5 invalides | 0 | | ns | 21-22 | |
| 11 | Données valides à partir de $\overline{CS}$ bas | | 310 | ns | 21 | |
| 12 | Lect.Donnée Valide à $\overline{DTACK}$ bas(établis.) | 50 | | ns | 21 | |
| 13 | $\overline{DTACK}$ bas à $\overline{DS}$, $\overline{CS}$, $\overline{IACK}$ haut(maintient) | 0 | | ns | 21-24 | |
| 14 | $\overline{IEI}$ bas à chute de $\overline{CLK}$ (établissement) | 50 | | ns | 23 | 2 |
| 15 | Valid. $\overline{IEO}$ à partir de CLK bas (délai) | | 180 | ns | 23 | 1 |
| 16 | Donnée Valide à partir CLK bas (délai) | | 300 | ns | 23 | |
| 17 | $\overline{IEO}$ Invalide à partir $\overline{IACK}$ haut (délai) | | 150 | ns | 23-24 | |
| 18 | $\overline{DTACK}$ bas à partir de CLK haut (Délai) | | 180 | ns | 23-24 | 2 |
| 19 | $\overline{IEO}$ Valide à partir de $\overline{IEI}$ bas (Délai) | | 100 | ns | 24 | 1 |
| 20 | Donnee Valide à partir de $\overline{IEI}$ bas (Délai) | | 220 | ns | 24 | |
| 21 | Temps d'un cycle horloge | 250 | 1000 | ns | 21-24 | |
| 22 | Largeur horloge bas | 110 | | ns | 21-24 | |
| 23 | Largeur horloge haut | 110 | | ns | 21-24 | |
| 24 | $\overline{CS}$, $\overline{IACK}$ inactifs à CLK montant (montage) | 100 | | ns | 21-24 | 4 |
| 25 | Largeur Impuls. Active Minimum d'E/S | 100 | | ns | | |
| 26 | Largeur $\overline{IACK}$ Haut | 150 | | ns | 23-24 | |
| 27 | Donnée E/S Valide à partir $\overline{CS}$, $\overline{DS}$ montant | | 450 | ns | 21-22 | |
| 28 | Délai Recepteur Pret à partir RC montant | | 600 | ns | 21 | |
| 29 | Délai Transmet. Pret à partir TC montant | | 600 | ns | 21 | |
| 30 | Sortie Timer bas de front montant $\overline{CS}$, $\overline{DS}$ | | 450 | ns | 21-24 | |
| 31 | Validat. $T_{OUT}$ à partir Timeout interne | | $2t_{CLK}$ +300 | ns | 23 | |
| 32 | Durée Timer bas | 110 | | ns | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 33 | Durée Timer haut | 110 | | ns | 23 | |
| 34 | Durée cycle horloge Timer | 250 | 1000 | ns | 23-24 | |
| 35 | Durée $\overline{\text{RESET}}$ bas | 2 | | µs | | |
| 36 | Délai à la chute de $\overline{\text{INTR}}$ à partir de E.I.A.T. | | 380 | ns | | 5 |
| 37 | T.I.I.D. à partir front mont. ou desc. de TC | 550 | | ns | | 6 |
| 38 | R.B.F.I.T.D. à partir front mont. ou desc. de RC | 800 | | ns | | 7 |
| 39 | R.E.I.T.D. à partir front descendant de RC | 800 | | ns | | 8 |
| 40 | S.I.S.U.T. jusqu'au front montant de RC | 80 | | ns | | 9 |
| 41 | D.H.T. à partir du front montant de RC | 350 | | ns | | 10 |
| 42 | S.O.D.V. à partir du front descend. de TC | | 440 | ns | | 11 |
| 43 | Durée signal horloge bas Transmetteur | 500 | | ns | | |
| 44 | Durée signal horloge haut Transmetteur | 500 | | ns | | |
| 45 | Durée Cycle horloge Transmetteur | 1,05 | ∞ | µs | | |
| 46 | Durée signal horloge bas Récepteur | 500 | | ns | | |
| 47 | Durée signal horloge haut Récepteur | 500 | | ns | | |
| 48 | Durée cycle horloge Récepteur | 1,05 | ∞ | µs | | |
| 49 | Largeur bas $\overline{\text{CS}}$, $\overline{\text{IACK}}$, $\overline{\text{DS}}$ | | 80 | $T_{CLK}$ | | |
| 50 | S.O.D.V. à partir du front montant de TC | | 490 | ns | | 12 |

NOTES:

1. IEO passe à l'état bas si aucune interruption n'est en attente. Si IEO passe à l'état bas, DTACK et le bus de données restent en état 'trois états'.

2. Si la spécification 14 est rencontrée, DTACK passe à l'état bas en A. Sinon DTACK passe à l'état bas en B.

3. Si le temps de pré-établissement n'est pas rencontré, CS et IACK ne seront pas reconnus avant le prochain front descendant d'horloge.

4. Si le temps de pré-établissement est rencontré (pendant plusieurs cycles consécutifs), on obtiendra une durée minimun 'hold-off' de un cycle horloge. Sinon cette durée sera de deux cycles.

5. E.I.A.T. Transition active d'interruption externe

6. T.I.I.D. Délai d'interruption émetteur interne

7. R.B.F.I.T.D. Délai de transition d'interruption 'Tampon Receveur plein'.





8. R.E.I.T.D. Délai de transition d'interruption 'Erreur Receveur'.

9. S.I.S.U.T. Temps de pré-établissement de données séries (division par 1 seulement)

10. D.H.T. Temps de maintien de données (Division par 1 seulement)

11. S.O.D.V. Données valides sur la Sortie Série (Division par 1 seulement)

12. S.O.D.V. Données valides sur le Sortie Série (Division par 16)

CAPACITES
TA = 25°C, f = 1MHz
Les broches non mesurées sont mises à la terre

| SYM | PARAMETRES | MAX | UNITE | CONDITIONS DE TEST |
|---|---|---|---|---|
| $C_{IN}$ | Capacité Entrée | 10 | pf | Broches non mesurées branchées à la terre |
| $C_{OUT}$ | Capacité Sortie Haute Impédance | 10 | pf | |

CARACTERISTIQUES TIMER

Définitions:

Erreur = Valeur du temps indiquée - Valeur temps réel
tpsc = tCLK x Valeur de pré-division

Mode Timer Interne

| | |
|---|---|
| Erreur simple intervale (free running)(1) | ±100 ns |
| Erreur interne cumulée | 0 |
| Erreur entre deux lectures du Timer | ±(tpsc+ 4 tCLK) |
| Départ Timer à Erreur arrêt Timer | 2tCLK+100ns à -(tpsc+6tCLK+100ns) |
| Départ Timer à erreur lecture Timer | 0 à -(tpsc+6tCLK+400ns) |
| Départ Timer à demande d'interruption erreur (2) | -2tCLK à -(4tCLK+800ns) |

Mode mesure de largeur d'impulsion

| | |
|---|---|
| Précision de mesure | 2tCLK à -(tpsc+4tCLK) |
| Largeur minimum d'impulsion | 4tCLK |

Mode compteur d'événements

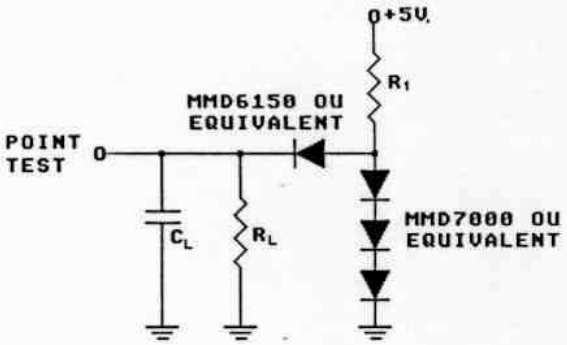
| | |
|---|---|
| Durée d'activité minimum de TAI et TBI | 4tCLK |
| Durée d'inactivité minimum de TAI et TBI | 4tCLK |

Notes.

1. Erreur en respectant TOUT ou INT si (2) est vrai
2. Etant assuré que le Timer peut faire une demande d'interruption immédiatement

SORTIE TYPE
Figure 18.

SORTIE TYPE

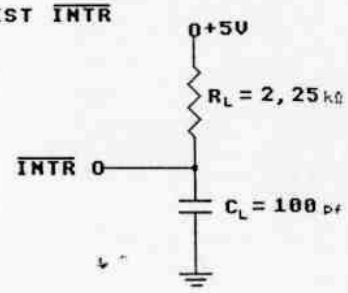



POUR TOUTES LES SORTIES SAUF $\overline{DTACK}$

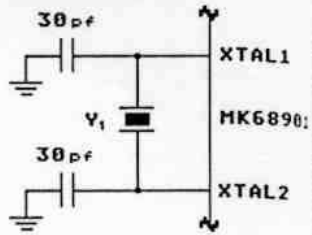
$C_L$ = 100 pf
$R_L$ = 20 kΩ
$R_L$ = 1.90 kΩ

POUR $\overline{DTACK}$ $C_L$ = 130 pf
$R_L$ = 6 kΩ
$R_L$ = 740 Ω

TEST $\overline{INTR}$




COMPOSANTS DE L'OSCILLATEUR EXTERNE DU MK68901




PARAMETRES DU QUARTZ

Résonance parallele, mode fondamental AT coupé

$R_s$ < 150Ω ($F_R$ = 2,8-4,0 MHz)
$R_s$ < 300Ω ($F_R$ = 2,0-2,7 MHz)

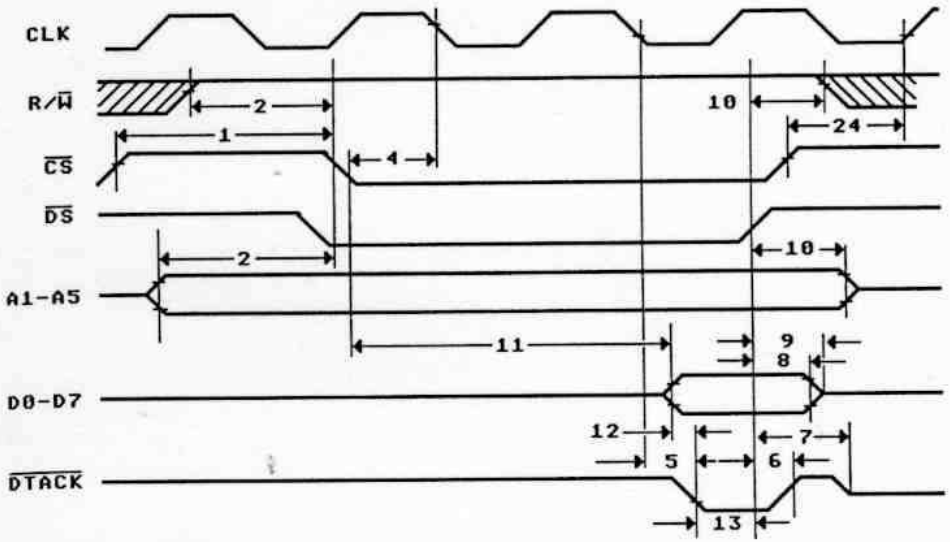
$C_L$ = 18pf $C_M$ = 0,02pf $C_h$ = 5pf $L_M$ = 96 MHz

$F_{R(type)}$ = 2,4576 MHz

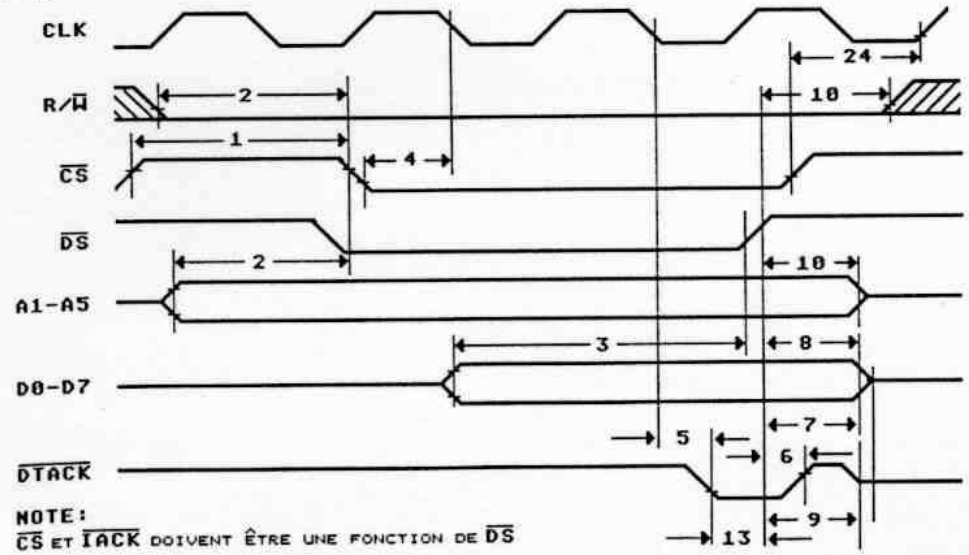




CYCLE DE LECTURE
Figure 21.




CYCLE D'ECRITURE
Figure 22.




NOTE:
$\overline{CS}$ ET $\overline{IACK}$ DOIVENT ÊTRE UNE FONCTION DE $\overline{DS}$

Figure 23.

RECONNAISSANCE D'INTERRUPTION (IEI BAS)

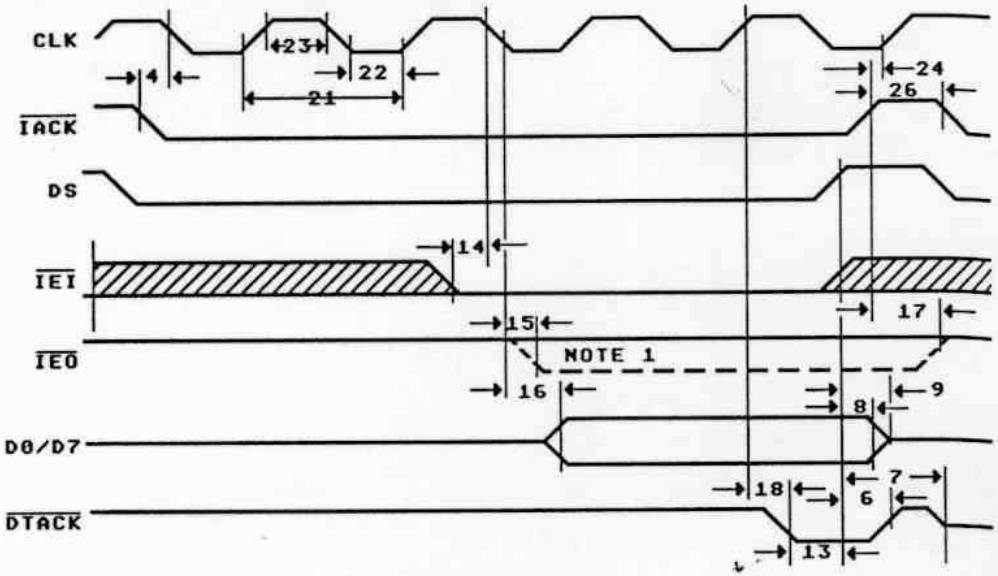
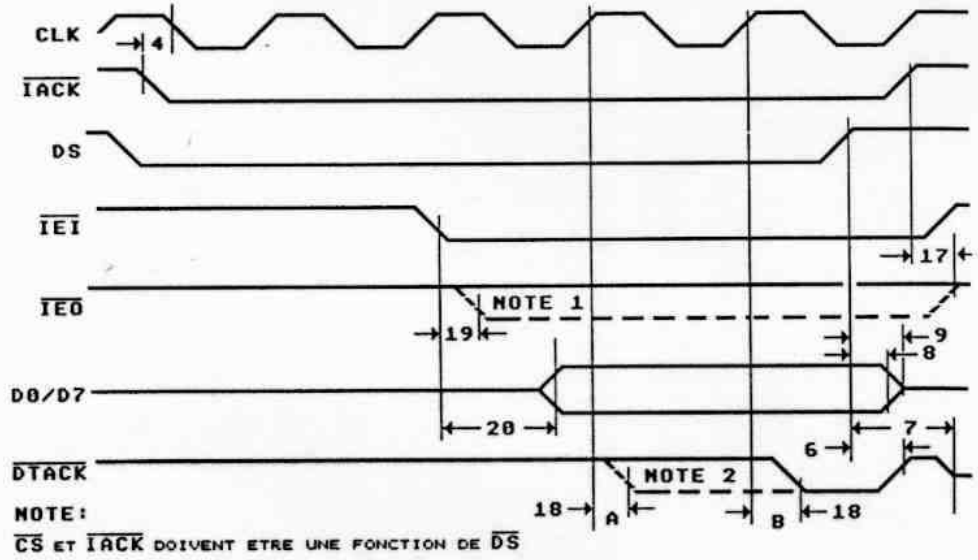



Figure 24.

RECONNAISSANCE D'INTERRUPTION (IEI HAUT)




NOTE:
CS ET IACK DOIVENT ETRE UNE FONCTION DE DS





# L E  A Y - 3 - 8 9 1 0

# 1. INTRODUCTION

Au premier abord tout microprocesseur est capable de produire des sons acceptables avec un simple convertisseur si le processeur n'a pas d'autres tâches à assurer tant que le son est soutenu. En réalité le microprocesseur s'occupe aussi du rafraîchissement de l'écran, de la scrutation du clavier, etc. Or, pour produire par exemple un simple son sur la neuvième octave (8372Hz), il faut contrôler le son durant six microsecondes environ. Un logiciel capable de produire ce simple effet sonore et en même temps de réaliser d'autres actions sera à coup sûr très complexe sinon irréalisable. Un son aléatoire réclame a fortiori une attention plus continue.

Cette nécessité de produire des sons sans un contrôle permanent du microprocesseur est désormais satisfaite grâce à la disponibilité des Générateurs Programmables de Sons AY-3-8910 et AY-3-8912 de General Instrument.

## 1.1. Description

Le Générateur Programmable de Sons (PSG) est un circuit intégré haute densité qui peut produire une grande variété de sons complexes sous contrôle logiciel.

Le circuit AY-3-8910/8912 est fabriqué en technologie N-channel. Son fonctionnement nécessite une alimentation 5V, une horloge à niveau compatible TTL et un contrôleur ou un microprocesseur comme un 16 bits CP 1600/1610 ou un 8 bits de la série PIC 1650.

Le PSG est facilement interfaçable avec tout système orienté bus. Sa flexibilité permet son utilisation dans des applications comme la synthèse musicale, la génération d'effets sonores, les alarmes sonores ou les modems acoustiques. Les ports sonores peuvent fournir 4 bits de conversion numérique-analogique, ce qui contribue fortement à améliorer la plage de sons.

Afin d'autoriser des effets sonores tout en laissant le processeur poursuivre d'autres tâches, le PSG offre la possibilité de produire des sons par une commande de départ fournie par le microprocesseur. La production d'effets sonores réalistes nécessitant souvent plus d'un canal, le PSG met à votre disposition trois canaux sonores contrôlables séparément.

Tous les circuits de contrôle du circuit sont numériques et autorisent un contrôle direct du processeur. Cela signifie qu'un PSG peut produire l'étendue complète des sons voulus sans aucun





changement dans les circuits d'interface. En effet la fréquence de sortie du PSG pouvant varier de l'infra-son pour la plus basse fréquence à l'ultra-son pour la plus haute, il y a peu de sons qui ne peuvent être reproduits par les connexions électriques les plus élémentaires.

La plupart des applications d'un système PSG/microprocesseur nécessitant un interfacage entre le monde extérieur et ce système, cette possibilité a été implémentée sur le PSG. Le AY-3-8910 a deux ports d'entrée-sortie 8 bits, il est conditionné dans un boitier 40 broches. Le AY-3-8912 a un port et 28 broches.

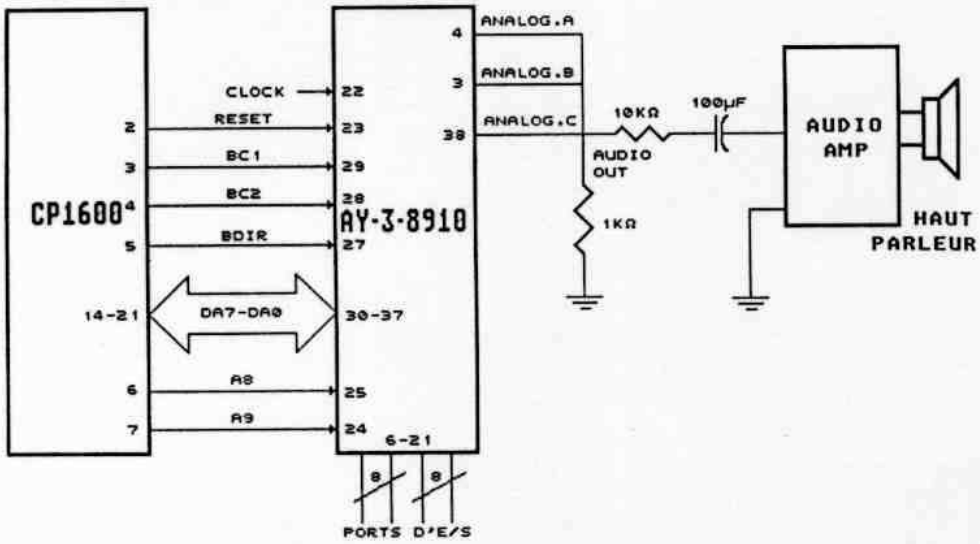
## 1.2. Caractéristiques

* Contrôle logiciel total de la génération de sons.
* Interfaces vers de nombreux microprocesseurs 8 et 16 bits.
* Trois sorties analogiques programmables indépendamment.
* Deux ports d'entrée-sortie à usage général (AY-3-8910).
* Un port d'entrée-sortie à usage général (AY-3-8912).
* Une seule alimentation +5V nécessaire.

## 1.3. Généralités

Cette documentation est destinée à présenter les techniques nécessaires pour utiliser le Générateur Programmable de Sons avec toute l'étendue de ses possibilités. Tous les programmes, les exemples et les schémas techniques ont fait l'objet de tests, lesquels garantissent que les méthodes exposées ne sont pas que pure théorie.

Bien que les techniques décrites permettent de produire des résultats étonnants, l'étendue des sons pouvant être synthétisés est si vaste et les potentialités du PSG sont si nombreuses que ce guide ne saurait être considéré que comme un simple manuel d'introduction aux applications possibles du PSG.




## 2.ARCHITECTURE

Le AY-3-8910/8912 est un Générateur de Sons Programmable (PSG) orienté registres. Les échanges de données entre le processeur et le PSG sont basés sur le concept des Entrées-Sorties à adresse mémoire. Les commandes de contrôle sont envoyées au PSG par écriture dans 16 registres adressables. Chacun des registres du PSG peut également être lu de sorte que le processeur peut connaître, quand cela est nécessaire, leur état ou leur contenu.

Toutes les fonctions du PSG sont contrôlables à travers ces 16 registres, lesquels, une fois programmés, génèrent et suivent les sons, déchargeant ainsi le processeur de ces tâches.

### 2.1. Blocs fonctionnels de base

Un diagramme fonctionnel complet est fourni pages 128 et 129.

#### 2.1.1. TABLE DES REGISTRES

La partie principale du PSG est une table de 16 registres de contrôle accessibles en lecture et écriture. Ces 16 registres sont vus par le processeur maître comme un bloc de mémoire et comme tel ils occupent un bloc de 16 mots parmi 1024 adresses possibles. Les dix bits d'adresse (8 bits sur le bus commun de données/adresses, et deux lignes d'adresse distinctes A8 et A9) sont décodés comme suit :

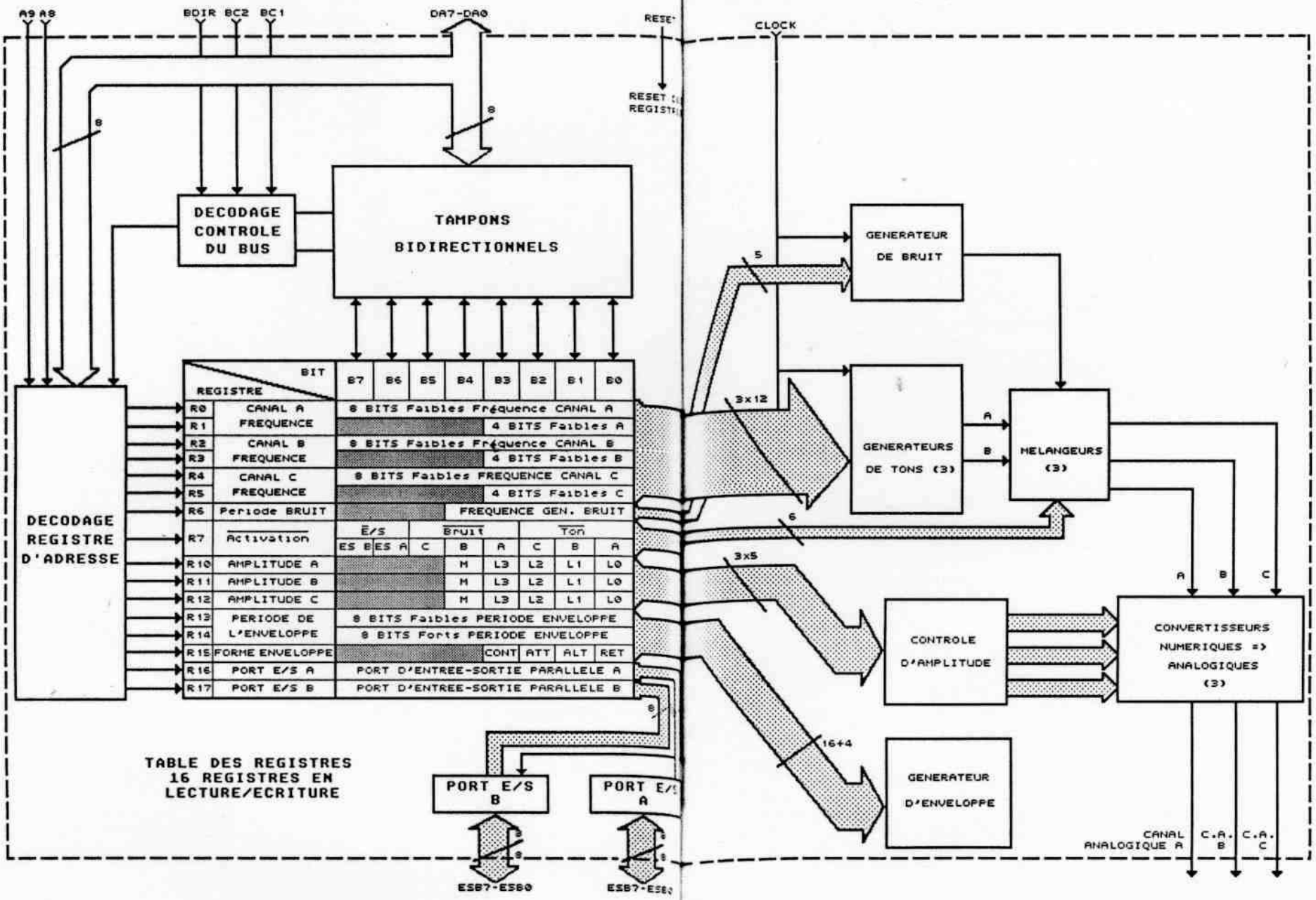




| *A9 | A8 | DA7 | DA6 | DA5 | DA4 | DA3 | DA2 | DA1 | DA0 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| JUSQU'A | | | | | | | | | |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| PARTIE HAUTE (Sélection circuit) | | | | | | PARTIE BASSE (Numéro de Registre) | | | |

*A9 n'est pas disponible sur le AY-3-8912.

Les quatre bits faibles de l'adresse permettent de choisir l'un des seize registres (R0-R17). Les six bits forts de l'adresse servent à la "sélection de circuit", c'est-à-dire à contrôler les tampons bidirectionnels trois-états (lorsque les bits forts de l'adresse ont une valeur incorrecte, les tampons bidirectionnels sont forcés à l'état haute impédance). Les lignes A8 et A9 sont forcées par le PSG pour toujours être à la valeur 01; les bits forts DA7 à DA4 peuvent être placés à toute combinaison possible de ces 4 bits lors de la fabrication du circuit. Sans spécification contraire, les bits d'adresse DA7 à DA4 sont placés à 0000.




| REGISTRE | | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---|---|---|---|---|---|---|---|---|---|
| R0 | CANAL A FREQUENCE | 8 BITS Faibles Fréquence CANAL A | | | | | | | |
| R1 | | | | | | 4 BITS Faibles A | | | |
| R2 | CANAL B FREQUENCE | 8 BITS Faibles Fréquence CANAL B | | | | | | | |
| R3 | | | | | | 4 BITS Faibles B | | | |
| R4 | CANAL C FREQUENCE | 8 BITS Faibles FREQUENCE CANAL C | | | | | | | |
| R5 | | | | | | 4 BITS Faibles C | | | |
| R6 | Periode BRUIT | | | | FREQUENCE GEN. BRUIT | | | | |
| R7 | Activation | E/S | | Bruit | | | Ton | | |
| | | ES B | ES A | C | B | A | C | B | A |
| R10 | AMPLITUDE A | | | | M | L3 | L2 | L1 | L0 |
| R11 | AMPLITUDE B | | | | M | L3 | L2 | L1 | L0 |
| R12 | AMPLITUDE C | | | | M | L3 | L2 | L1 | L0 |
| R13 | PERIODE DE L'ENVELOPPE | 8 BITS Faibles PERIODE ENVELOPPE | | | | | | | |
| R14 | | 8 BITS Forts PERIODE ENVELOPPE | | | | | | | |
| R15 | FORME ENVELOPPE | | | | | CONT | ATT | ALT | RET |
| R16 | PORT E/S A | PORT D'ENTREE-SORTIE PARALLELE A | | | | | | | |
| R17 | PORT E/S B | PORT D'ENTREE-SORTIE PARALLELE B | | | | | | | |

TABLE DES REGISTRES
16 REGISTRES EN
LECTURE/ECRITURE



Circuit AY-8910

Une adresse haute valide place les 4 bits faibles dans le registre de contrôle et de décodage d'adresse. Une adresse valide le demeure jusqu'à la réception de l'adresse suivante, autorisant donc de multiples lectures et/ou écritures du même registre sans adressage redondant.

La gestion du registre de contrôle et de décodage d'adresse et des tampons bidirectionnels est assurée par le bloc de contrôle du bus, lequel détecte la fonction bus requise (désactivation, écriture, lecture).

Les fonctions de chacun des 16 registres du PSG et les flux de données partant de ces registres sont décrits sur le schéma général des pages précédentes et expliqués en détail dans le chapitre 3, "Fonctionnement". Afin de faciliter vos recherches, la table descriptive des fonctions des registres est fournie à part en page suivante.

2.1.2. Blocs de génération de sons.

Les blocs de base du PSG qui produisent des sons programmés sont :

Les générateurs sonores — produisent un signal carré de fréquence programmable (canaux A, B, C).

Le générateur de bruit — produit une fréquence modulée pseudo-aléatoire avec un signal carré.

Les mélangeurs — combinent les sorties des générateurs sonores et du générateur de bruit. Autant de mélangeurs que de générateurs.

Le contrôleur d'amplitude — fournit aux convertisseurs numériques-analogiques une matrice d'amplitude soit fixe soit variable. Une amplitude fixe est sous le contrôle direct du processeur; une amplitude variable est produite par le générateur d'enveloppe.

Le générateur d'enveloppe — fournit un modèle d'enveloppe pouvant servir à moduler l'amplitude de chacune des sorties des trois mélangeurs.

Les convertisseurs N/A — Les trois convertisseurs numériques-analogiques produisent un signal de sortie sur 16 niveaux contrôlables par le contrôleur d'amplitude.

2.1.3. Ports d'entrée-sortie

Deux blocs supplémentaires figurent sur le schéma général du PSG. Ils n'ont rien à voir avec la production de sons - ce sont





les deux ports d'entrée-sortie (A et B). Etant donné que toutes les opérations de génération de sons réclament un interfaçage entre le système et le monde extérieur, cette facilité a été offerte par le PSG. Les données lues ou écrites par le processeur peuvent être lues ou écrites sur chacun des ports 8 bits sans qu'en soit affectée une fonction quelconque du PSG. Les ports A et B sont compatibles TTL et disposent de résistances internes de rappel. Deux ports sont disponibles sur le AY-3-8910, seul le port A est disponible sur le AY-3-8912.

SYNOPTIQUE DES FONCTIONS DES REGISTRES

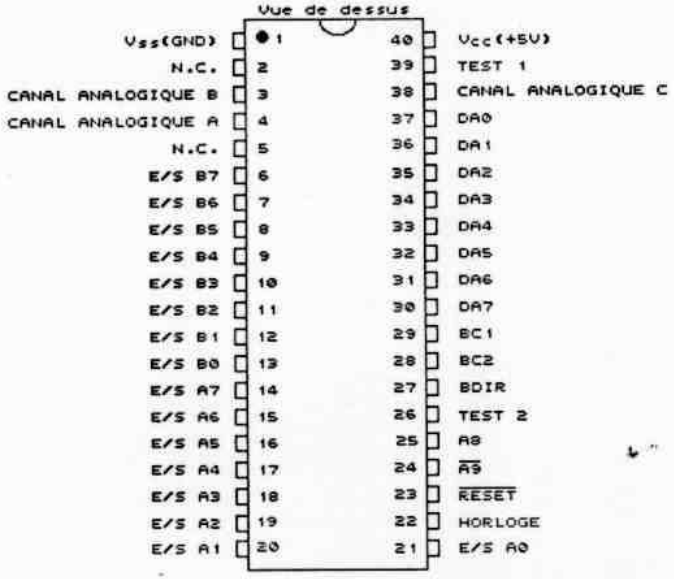
| REGISTRE \ BIT | | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---|---|---|---|---|---|---|---|---|---|
| R0 | Période CANAL A | 8 bits faibles Fréquence CANAL A | | | | | | | |
| R1 | | | | | | 4 bits forts A | | | |
| R2 | Période CANAL B | 8 bits faibles Fréquence CANAL B | | | | | | | |
| R3 | | | | | | 4 bits forts B | | | |
| R4 | Période CANAL C | 8 bits faibles Fréquence CANAL C | | | | | | | |
| R5 | | | | | | 4 bits forts C | | | |
| R6 | Période Bruit | | | | Fréquence Génér. Bruit | | | | |
| R7 | $\overline{\text{Activation}}$ | Ent/Sort | | Bruit | | | Tonalité | | |
| | | ES B | ES A | C | B | A | C | B | A |
| R10 | Amplitude A | | | | M | L3 | L2 | L1 | L0 |
| R11 | Amplitude B | | | | M | L3 | L2 | L1 | L0 |
| R12 | Amplitude C | | | | M | L3 | L2 | L1 | L0 |
| R13 | Période de l'enveloppe | 8 bits faibles Fréquence Enveloppe | | | | | | | |
| R14 | | 8 bits forts Fréquence Enveloppe | | | | | | | |
| R15 | Forme enveloppe | | | | | CONT | ATT | ALT | RET |
| R16 | PORT D'E/S A | PORT PARALLELE 8 bits E/S A | | | | | | | |
| R17 | PORT D'E/S B | PORT PARALLELE 8 bits E/S B | | | | | | | |

N.D.T.: Le circuit AY-3-8910 de General Instrument est souvent remplacé sur le ST par un circuit YM2149 de Yamaha, lequel offre dans un boitier de 40 broches les mêmes fonctions que le circuit étudié ici. L'un des ports d'entrée-sortie sert à l'interfaçage Centronics de l'imprimante, l'autre est utilisé pour diverses fonctions d'interface série, disque, etc.

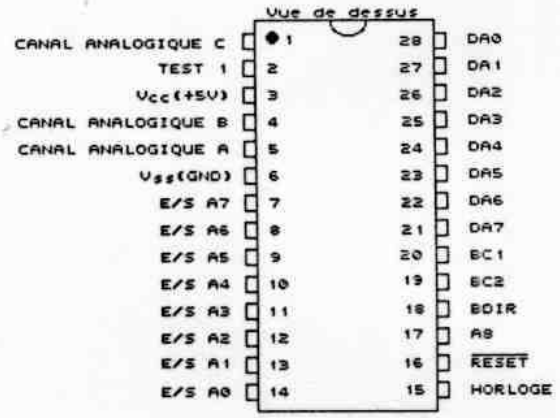
## 2.2. Schéma de brochage

Le circuit AY-3-8910 se présente comme un boitier 40 broches dont le schéma de brochage est fourni ci-dessous. Le circuit AY-3-8912 est fourni sous forme d'un boitier à 28 broches dont le schéma figure également ci-dessous.

AY-3-8910




AY-3-8912








## 2.3. Fonctions des broches

DA7-DA0 (entrées-sorties haute impédance)
broches 30 à 37(AY-3-8910), broches 21-28(AY-3-8912)

Lignes de données et d'adresses 7--0:
Ces 8 lignes constituent le bus bidirectionnel 8 bits utilisé par le processeur maître pour lire et écrire des données vers le PSG. En mode données, DA7-DA0 correspond à la table des registres B7--B0 (voir schéma général). En mode adresse, DA3--DA0 sélectionne le numéro de registre (0-17) et DA7--DA4 associés à A8 et A9 forment l'adresse haute (sélection de circuit).

A8 (entrée)
broche 25(AY-3-8910), broche 17(AY-3-8912)
$\overline{A9}$ (entrée)
broche 24(AY-3-8910), broche inexistante sur le AY-3-8912.

Lignes d'adresse A8 et $\overline{A9}$:
Ces bits d'adresse "supplémentaires" permettent de positionner le PSG (lequel requiert 16 mots de mémoire) dans une zone de 1024 mots de mémoire plutôt que dans une zone de 256 mots comme les seuls registres DA7--DA0 le permettraient. Si la taille mémoire ne nécessite pas l'utilisation de ces lignes d'adresse accessoires, elles peuvent rester non connectées. Toutefois, dans un environnement difficile, il est recommandé de les relier respectivement à la masse et à l'alimentation +5V si elles sont inutilisées.

N.D.T.: Le ST n'utilise pas ces deux lignes et comme il était conseillé, la ligne A8 est reliée à l'alimentation et la ligne A9 à la masse...

$\overline{RESET}$ (entrée)
broche 23(AY-3-8910), broche 16(AY-3-8912)

Lors de l'initialisation ou de la mise sous tension, appliquer un 0 logique (masse) à la broche Reset réinitialisera tous les registres à 0.

HORLOGE (entrée)
broche 22(AY-3-89190), broche 15(AY-3-8912)

Cette entrée compatible TTL fournit la base de temps nécessaire aux générateurs sonores, au générateur de bruit et au générateur d'enveloppe.

BDIR,BC2,BC1 (entrées)
broches 27,28,29(AY-3-8910), broches 18,19,20(AY-3-8912)

Direction de bus, Contrôle du bus 1 et 2.
Ces signaux de contrôle du bus sont générés directement par les microprocesseurs de la série CP1600 afin de

contrôler toutes les opérations sur le bus du PSG. Lorsque l'on utilise un autre processeur que le CP1600, ces signaux peuvent être utilisés soit pour une gestion comparable du bus, soit pour simuler les signaux sur les lignes d'entrée-sortie du processeur. Le PSG décode ces signaux selon la table suivante :

| BDIR | BC2 | BC1 | Fonction CP1600 | Fonction PSG |
|---|---|---|---|---|
| 0 | 0 | 0 | NACT | DESACTIVATION (voir 010 [IAB] plus bas) |
| 0 | 0 | 1 | ADAR | Verrouillage ADRESSE (voir 111 [INTAK]) |
| 0 | 1 | 0 | IAB | Désactivation. Bus inactif. Lignes DA7 à l'état haute impédance |
| 0 | 1 | 1 | DTB | LECTURE à partir du PSG. Ce signal amène le contenu du registre courant adressé sur le bus. DA7-DA0 sont en état sortie. |
| 1 | 0 | 0 | BAR | Verrouillage ADRESSE (voir 111 [INTAK]) |
| 1 | 0 | 1 | DW | Désactivation (voir 010 [IAB] plus haut) |
| 1 | 1 | 0 | DWS | ECRITURE vers le PSG. Ce signal indique que le registre adressé va être écrit avec le contenu du bus DA7-DA0. |
| 1 | 1 | 1 | INTAK | VERROUILLAGE D'ADRESSE. Ce signal indique que le bus contient une adresse de registre qui doit être verrouillée. |

Etant donné que l'interfaçage d'un PSG avec un autre microprocesseur nécessitera simplement la simulation des signaux de contrôle du bus décrits ci-dessus, on notera certaines redondances des signaux de commande puisque seuls 4 signaux sont nécessaires pour 8 combinaisons de bits. Cela devrait nettement simplifier la programmation de la gestion du bus, seuls deux signaux de contrôle étant indispensables (BDIR et BC1) et le signal BC2 pouvant tout bonnement être relié à l'alimentation +5V:

| BDIR | BC2 | BC1 | Fonction PSG |
|---|---|---|---|
| 0 | 1 | 0 | DESACTIVATION |
| 0 | 1 | 1 | LECTURE DU PSG |
| 1 | 1 | 0 | ECRITURE DU PSG |
| 1 | 1 | 1 | VEROUILLAGE ADRESSE |

N.D.T.: L'implantation de ce circuit sur le ST adopte cette gestion simplifiée du contrôle du bus. BC2 est relié au 5V.





CANAUX ANALOGIQUES A, B, C (entrée)
broches 4,3,38(AY-3-8910), broches 5,4,1(AY-3-8912)

Chacun de ces signaux correspond à l'une des sorties des trois convertisseurs numérique-analogique et fournissent un signal jusqu'à 1V crête à crête représentant le son complexe généré par le PSG.

ENTREES-SORTIES ES A7 - ES A0 (entrée-sortie)
broches 14-21(AY-3-8910), broches 7-14(AY-3-8912)
ENTREES-SORTIES ES B7 - ES B0 (entrée-sortie)
broches 6-13(AY-3-8910), inexistant sur l'AY-3-8912

Lignes d'entrée-sortie A7--A0, B7--B0
Chacun de ces ports d'entrée-sortie fournit 8 lignes parallèles de données disponibles du processeur via le PSG en lecture comme en écriture. Chaque broche est reliée à une résistance de rappel de sorte qu'en mode "entrée", toutes les broches doivent être à l'état haut. C'est pourquoi la méthode recommandée pour interroger des commutateurs externes, par exemple, serait de relier les bits d'entrée à la masse.

TEST1 broche 39(AY-3-8910), broche 2(AY-3-8912)
TEST2 broche 26(AY-3-8910), inexistante sur l'AY-3-8912

Ces broches sont destinées à des tests matériels et doivent être laissées en l'air. Ne les utilisez pas.

Vcc broche 40(AY-3-8910), broche 3(AY-3-8912)

Alimentation nominale de +5V fournie au PSG.

Vss broche 1(AY-3-8910), broche 6(AY-3-8912)

Masse de référence du PSG.

## 2.4. Chronogramme du Bus

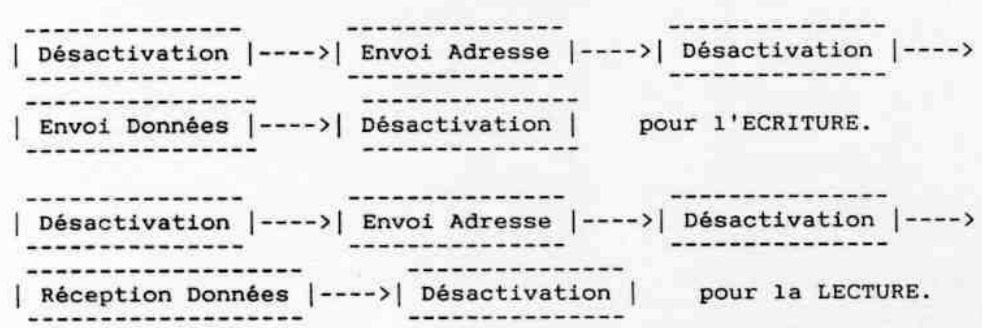
Etant donné que les fonctions du PSG sont contrôlées par des commandes envoyées par le processeur, le bus commun DA7--DA0 données/adresses nécessite d'être défini à chaque moment. C'est ce qu'effectue le processeur grâce aux signaux de contrôle du bus, précédemment décrits, lesquels définissent l'état du bus : le PSG décode ensuite ces signaux pour accomplir la tâche demandée.

Le maniement de ces signaux de contrôle est semblable à ce qu'il serait si le processeur désirait contrôler des mémoires vives : (1) le processeur génère une adresse mémoire; (2) le processeur lit ou écrit une donnée à cette adresse. Dans le cas du PSG, la "mémoire" est l'un des seize registres accessibles en lecture et écriture.

Les chronogrammes relatifs aux signaux de contrôle du bus sont étudiés dans les pages qui suivent.

## 2.5. Chronogramme d'état

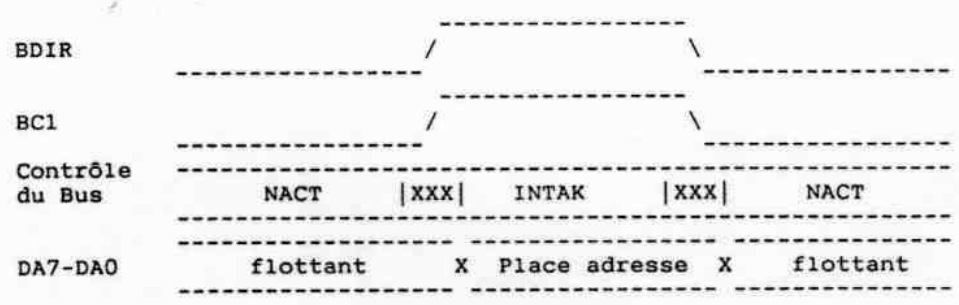
Le contrôle des lignes d'état par nombre de microprocesseurs pouvant être gêné par d'autres opérations, la séquence des événements nécessaires pour contrôler le PSG a été simplifiée et dépouillée. Chacune des séquences principales (verrouillage d'adresse, écriture vers le PSG et lecture du PSG) est composée de plusieurs opérations (indiquées par des blocs rectangulaires dans la figure ci-dessous) définies en fonction de l'état des signaux de contrôle du bus (BDIR, BC2, BC1).




Le détail des opérations et leurs chronogrammes font l'objet des paragraphes suivants (dans tous les exemples, la ligne BC2 est supposée être reliée à +5V [niveau logique 1]).

### 2.5.1. Séquence d'adressage de registre PSG

La séquence de "verrouillage d'adresse" est en principe partie intégrante des séquences de lecture ou d'écriture mais, pour des raisons de simplification, fait l'objet ici d'une étude distincte. Bien que dépendant du processeur utilisé, la séquence requiert normalement quatre micro-états principaux : (1) envoi de NACT (désactivation); (2) envoi de INTAK (verrouillage d'adresse); (3) placement de l'adresse sur le bus; (4) envoi de NACT (désactivation). [Note: avec les contraintes de base de temps qui s'imposent, voir section 7, les étapes (2) et (3) peuvent être interverties.]







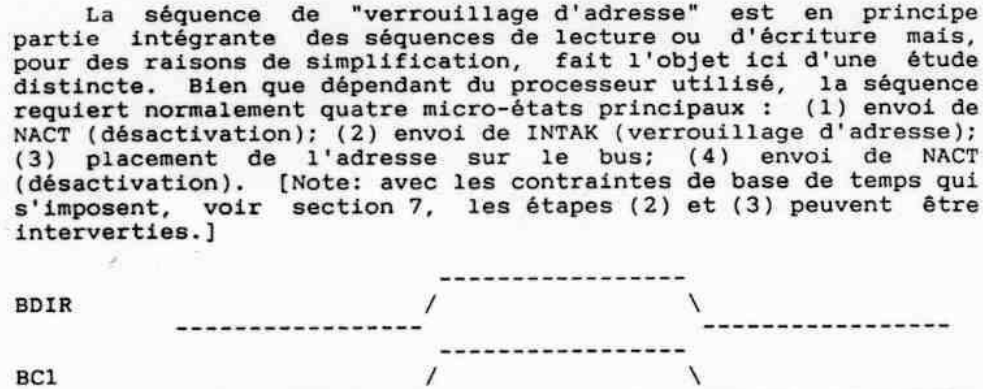

### 2.5.2. Séquence d'écriture de données vers le PSG

La séquence d'écriture de données vers le PSG, qui doit normalement être précédée d'un verrouillage d'adresse, nécessite quatre micro-états: (1) envoi de NACT (désactivation); (2) pose de la donnée sur le bus; (3) envoi de DWS (écriture vers le PSG); (4) envoi de NACT (désactivation).




### 2.5.3. Séquence de lecture de données venant du PSG

De même que la séquence d'écriture, la séquence de lecture de données est normalement précédée d'une séquence de verrouillage d'adresse. Les quatre micro-états de la séquence de lecture sont : (1) envoi de NACT (désactivation); (2) envoi de DTB (lecture du PSG); (3) lecture de la donnée sur le bus; (4) envoi de NACT (désactivation).




### 2.5.4. Séquence de lecture-écriture d'un port d'entrée-sortie

Les deux ports d'entrée-sortie (A et B) ayant chacun un registre de données pour le stockage, la lecture ou l'écriture sur l'un de ces ports s'effectue comme la lecture ou l'écriture de l'un des registres du PSG. En conséquence, les séquences d'état correspondant à ces opérations sont exactement semblables à celles décrites ci-dessus.

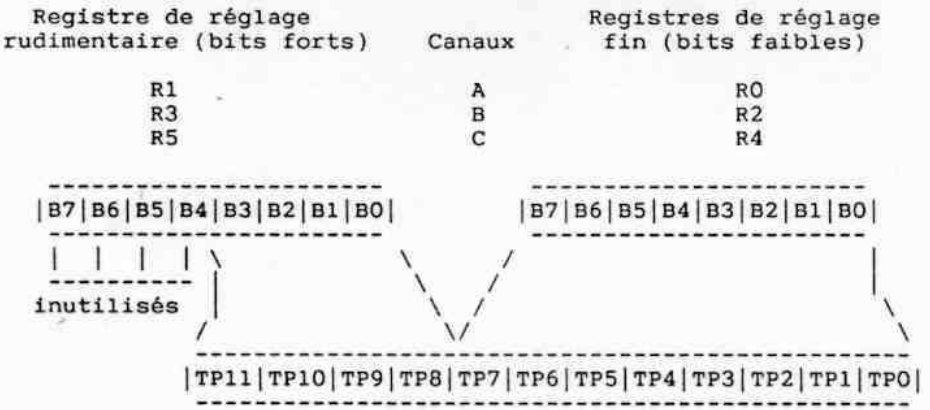
## 3. Fonctionnement

Etant donné que les fonctions du PSG sont contrôlées par le processeur maître à travers une série de registres, il semble préférable pour une description détaillée du fonctionnement du PSG d'exposer chaque fonction du PSG à partir du registre lui correspondant. La fonction de création ou de programmation d'un son donné ou d'un effet sonore suit logiquement la séquence suivante:

| Section | Opération | Registres | Fonction |
|---|---|---|---|
| 3.1. | Générateurs sonores | R0-R5 | Fréquences sonores |
| 3.2. | Générateur de bruit | R6 | Fréquence de bruit |
| 3.3. | Contrôle Mélangeurs | R7 | Activation son/bruit sur canaux corresp. |
| 3.4. | Contrôle d'amplitude | R10-R12 | Sélection d'amplitude fixe ou variable |
| 3.5. | Générateur et contrôle enveloppe | R13-R15 | Fréquence et forme de l'enveloppe |

### 3.1. Générateurs sonores (registres R0, R1, R2, R3, R4, R5)

La fréquence de chaque signal carré généré par l'un des trois générateurs sonores (correspondant aux canaux A, B et C) est calculée par le PSG après une division d'horloge d'un facteur 16 par décomptage d'une valeur de fréquence programmée sur 12 bits. Chaque valeur sur 12 bits est obtenue par le PSG en combinant les contenus des registres de bits forts et de bits faibles comme décrit ci-dessous :




Il est à noter que la valeur sur 12 bits placée dans les registres de réglage rudimentaire et de réglage fin est une valeur de période, c'est-à-dire qu'à une haute valeur correspondra une fréquence faible.

On notera également qu'étant donné ce mode de stockage, la valeur la plus basse de la période est %000000000001 (division par 1) et la plus haute %111111111111 (division par 4095).





Les équations fournissant la relation entre la fréquence de sortie de son désirée, la fréquence d'entrée de l'horloge et la valeur de période placée dans les registres sont :

$$fT = \frac{fHORL}{16TP10} \qquad TP10 = 256CT10 + FT10$$

où : fT = fréquence de sortie de son désirée
fHORL = fréquence d'entrée horloge
TP10 = période (équivalent décimal de TP11-TP0)
CT10 = bits forts période (TP11-TP8)
FT10 = bits faibles période (TP7-TP0)

Au vu de ces équations, chacun comprendra que la fréquence de sortie du son peut varier d'un minimum de $\frac{fHORL}{65520}$ (si TP10 = 4095) à un maximum de $\frac{fHORL}{16}$ (si TP10 = 1). En utilisant par exemple un signal d'horloge à 2 Mhz, on dispose donc d'une plage de fréquences allant de 30,5 Hz à 125 KHz (N.D.T.: Le ST cadence précisément le circuit son à partir d'un signal d'horloge à 2 MHz).

Pour calculer les valeurs des contenus des registres de réglage rudimentaire et de réglage fin correspondant à un signal d'horloge et à des fréquences sonores donnés, nous inverserons simplement les termes des équations, soit:

$$TP10 = \frac{fHORL}{16fT} \qquad CT10 + \frac{FT10}{256} = \frac{TP10}{256}$$

Exemple 1: fT = 1 KHz fHORL = 2MHz

$$TP10 = \frac{2x106}{16(103)} = 125$$

$$\text{d'où: } CT10 + \frac{FT10}{256} = \frac{125}{256}$$

On a donc : CT10 = 0 = %0000 (B3-B0)
FT10 = 125 = %01111101 (B7-B0)

Exemple 2: fT = 100 Hz fHORL = 2MHz

$$TP10 = \frac{2x106}{16(102)} = 1250$$

$$\text{d'où: } CT10 + \frac{FT10}{256} = \frac{1250}{256} = 4 + \frac{226}{256}$$

On a donc : CT10 = 4 = %0100 (B3-B0)
FT10 = 226 = %11100010 (B7-B0)

## 3.2. Générateur de bruit (R6)

La fréquence du générateur de bruit est obtenue, après division par 16 du signal d'horloge, par décomptage de la valeur contenue dans les 5 bits du registre de ce générateur. Cette valeur sur 5 bits est contenue dans les cinq bits faibles du registre R6 comme décrit ci-dessous:

Registre R6 (générateur de bruit)

| B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---|---|---|---|---|---|---|---|
| | | | <---------- période du générateur de bruit ----------> | | | | |
| inutilisés | | | | | | | |

On notera que cette valeur sur 5 bits correspond à une période, c'es-à-dire qu'à la valeur la plus haute dans ce registre correspondra la fréquence la plus basse. On notera aussi que, de la même façon que pour les générateurs sonores, la valeur la plus faible est %00001 (division par 1) alors que la valeur la plus élevée est %11111 (division par 31).

L'équation donnant la fréquence du bruit est:

$$fn = \frac{fHORL}{16NP10}$$

où: fN = fréquence désirée du bruit
fHORL = fréquence du signal d'horloge
NP10 = période (bits B4-B0 du registre R6)

A partir de cette équation, on peut déduire que la fréquence du bruit peut varier entre $\frac{fHORL}{496}$ (lorsque NP10 = 31) jusqu'à un maximum de $\frac{fHORL}{16}$ (lorsque NP10 = 1). Ainsi, en utilisant un signal d'horloge de 2MHz, les fréquences de bruit peuvent varier entre 4 KHz et 125 KHz.

Pour calculer la valeur à placer dans le registre R6 pour un signal d'horloge et une fréquence de bruit donnés, nous inversons simplement les opérateurs de l'équation précédente, d'où :

$$NP10 = \frac{fHORL}{16fN}$$





## 3.3. Contrôle des mélangeurs, activation des ports d'E/S (R7)

Le registre 7 est un registre multi-fonctions chargé d'activer les trois mélangeurs bruit/son et les deux ports d'entrée-sortie.

Les mélangeurs, comme décrit en 2.1.2., combinent les fréquences de sons et de bruit issues du générateur de bruit et des trois générateurs sonores. La façon dont s'effectue cette combinaison est fonction de l'état des bits B5-B0 du registre R7.

La direction (entrée ou sortie) des deux ports d'entrée-sortie (E/S A et E/S B) est quant à elle déterminée par l'état des bits B6 et B7 du registre R7.

Les tables ci-dessous illustrent ces fonctions:

Registre R7 (contrôle mixage, activation ports E/S)

B7 : Régulation des échanges sur le port d'E/S B.
B6 : Régulation des échanges sur le port d'E/S A.
B5-B3 : Activation bruit respectivement pour canaux C, B, A.
B2-B0 : Activation son respectivement pour canaux C, B, A.

Activation du son

| bits R7 | | | Son actif sur canal | | |
|---|---|---|---|---|---|
| B2 | B1 | B0 | | | |
| 0 | 0 | 0 | C | B | A |
| 0 | 0 | 1 | C | B | - |
| 0 | 1 | 0 | C | - | A |
| 0 | 1 | 1 | C | - | - |
| 1 | 0 | 0 | - | B | A |
| 1 | 0 | 1 | - | B | - |
| 1 | 1 | 0 | - | - | A |
| 1 | 1 | 1 | - | - | - |

Activation du bruit

| bits R7 | | | Bruit actif sur canal | | |
|---|---|---|---|---|---|
| B5 | B4 | B3 | | | |
| 0 | 0 | 0 | C | B | A |
| 0 | 0 | 1 | C | B | - |
| 0 | 1 | 0 | C | - | A |
| 0 | 1 | 1 | C | - | - |
| 1 | 0 | 0 | - | B | A |
| 1 | 0 | 1 | - | B | - |
| 1 | 1 | 0 | - | - | A |
| 1 | 1 | 1 | - | - | - |

| bits R7 | | Etat ports d'E/S | |
|---|---|---|---|
| B7 | B6 | port B | port A |
| 0 | 0 | Entrée | Entrée |
| 0 | 1 | Entrée | Sortie |
| 1 | 0 | Sortie | Entrée |
| 1 | 1 | Sortie | Sortie |

Note: La désactivation du son et du bruit sur un canal ne DESACTIVE PAS un canal. La désactivation d'un canal ne peut s'effectuer que par la mise de 0 dans le registre de contrôle d'amplitude lui correspondant (voir page suivante).

## 3.4. Contrôle d'amplitude (R10,R11,R12)

Les amplitudes des signaux générés par chacun des trois convertisseurs numérique-analogique (un par canal A, B ou C) sont déterminées par les contenus des 5 bits faibles (B4-B0) des registres R10, R11, R12 selon les régles suivantes:

R10 : Contrôle d'amplitude du canal A
R11 : Contrôle d'amplitude du canal B
R12 : Contrôle d'amplitude du canal C

B7-B5 : bits inutilisés
B4 : mode d'amplitude
B3-B0 : niveau d'amplitude

Le bit de mode d'amplitude (bit 4) permet de sélectionner soit un niveau fixe d'amplitude (s'il est à 0), soit un niveau variable d'amplitude (s'il est à 1). Il s'ensuit que les bits B3 à B0, définissant le niveau d'amplitude, ne sont actifs que lorsque le bit de mode est NUL. Lorsque le niveau d'amplitude fixe est sélectionné, il est fixe seulement dans la mesure où le niveau d'amplitude est sous le contrôle direct du processeur (à travers les bits D3 à D0). Une variation d'amplitude alors que l'on se trouve en mode d'amplitude fixe suppose, en tout état de cause, l'intervention directe du processeur, lequel doit générer une séquence de verrouillage d'adresse et d'écriture de donnée sur le bus, afin de modifier les données D3 à D0.

Lorsque le bit 4 est à 1 (sélection du mode d'amplitude variable), l'amplitude de chacun des canaux est déterminée par la forme de l'enveloppe comme indiqué par les 4 bits E3 E2 E1 et E0 du générateur d'enveloppe.

Le mode d'amplitude peut également être appréhendé comme un bit "d'activation de l'enveloppe" puisque lorsqu'il est nul l'enveloppe n'est pas prise en compte alors qu'elle est active s'il est à 1. (Une description complète de la fonction du Générateur d'Enveloppe est effectuée plus loin).

Le tableau suivant illustre la fonction de chacun des 5 bits d'un générateur d'amplitude dans les différents cas possibles:

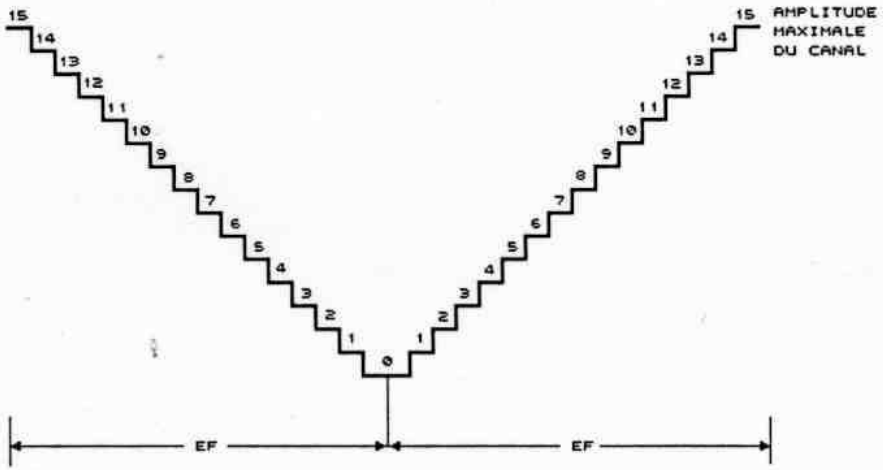
| B4 | B3 | B2 | B1 | B0 | Sortie Contrôle d'amplitude | |
|---|---|---|---|---|---|---|
| V | V | V | V | V | | |
| M | L3 | L2 | L1 | L0 | | |
| 0 | 0 | 0 | 0 | 0 | 0 0 0 0 | l'amplitude est |
| . | . | . | . | . | . . . . | fixée à l'un des |
| . | . | . | . | . | . . . . | seize niveaux |
| . | . | . | . | . | . . . . | possibles déter- |
| . | . | . | . | . | . . . . | minés par l'état |
| 0 | 1 | 1 | 1 | 1 | 1 1 1 1 | des bits L3-L0 |
| 1 | x | x | x | x | E3 E2 E1 E0 | amplitude variable |





La figure ci-dessous illustre un ensemble de niveaux variables d'amplitude pour lesquels les 16 niveaux sont directement le reflet du générateur d'enveloppe. Un niveau fixe d'amplitude correspondra seulement à l'un de ces niveaux, celui-ci étant fixé par la valeur des bits L3 à L0.

Contrôle d'amplitude variable (M=1)




EF désigne la fréquence d'enveloppe.

N.D.T. : Les registres d'amplitude ont été notés R10,R11,R12. De même les registres d'enveloppe et de contrôle des ports seront numérotés de R13 à R17, conformément à la documentation du constructeur. Toutefois, pour la programmation du circuit et spécialement de l'YM2149 équipant le ST, les registres devront être traités selon une numérotation **continue**. On aura donc :

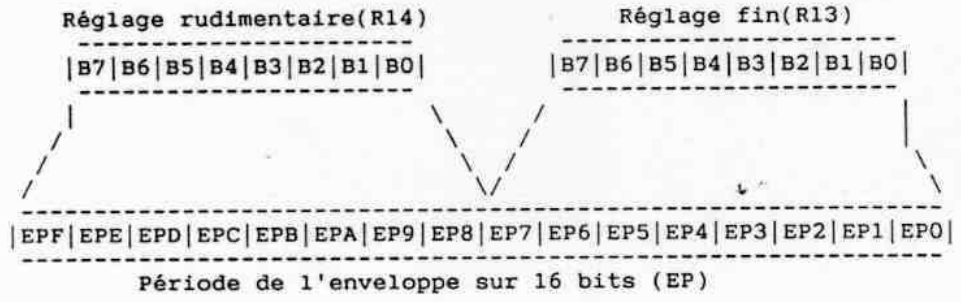
R8 : registre d'amplitude du canal A (R10 ici)
R9 : registre d'amplitude du canal B (R11 ici)
R10: registre d'amplitude du canal C (R12 ici)
R11: registre de bits faibles période d'enveloppe (R13 ici)
R12: registre de bits forts période d'enveloppe (R14 ici)
R13: registre de forme et de type d'enveloppe (R15 ici)
R14: registre de gestion du port d'entrée-sortie A (R16 ici)
R15: registre de gestion du port d'entrée-sortie B (R17 ici)

## 3.5. Contrôle du générateur d'enveloppe (R13,R14,R15)

Pour réaliser la mise en oeuvre de modèles d'enveloppe très complexes, deux méthodes différentes sont utilisables avec le PSG. D'une part il est possible de faire varier la fréquence de l'enveloppe à l'aide des registres R13 et R14, d'autre part la forme et le type de l'enveloppe peuvent être modifiés à l'aide du registre R15. Les paragraphes suivants expliquent en détail les fonctions de contrôle de l'enveloppe, en commençant par la période de l'enveloppe et en poursuivant par la forme et le type.

### 3.5.1. Contrôle de la période de l'enveloppe (R13 et R14)

La fréquence de l'enveloppe est calculée par le PSG, après une division par 256 du signal d'horloge, par décomptage d'une valeur programmée sur 16 bits. Cette valeur, la période de l'enveloppe, est obtenue par combinaison des registres R14 et R13, correspondant respectivement aux bits forts et faibles, à un réglage rudimentaire et à un réglage fin.



Période de l'enveloppe sur 16 bits (EP)

On notera que cette valeur programmée sur 16 bits par combinaison des registres R14 et R13 est une valeur de période - à la valeur la plus forte correspond la fréquence d'enveloppe la plus faible.

On relévera également que, comme pour la période des générateurs sonores, la valeur de période la plus faible est égale à %0000000000000001 (division par 1) tandis que la valeur de période la plus élevée est égale à %1111111111111111 (division par 65535).

Les équations définissant la fréquence d'enveloppe sont:

$$fE = \frac{fHORL}{256EP10} \qquad EP10 = 256CT10 + FT10$$

où : fE = fréquence d'enveloppe désirée
fHORL = fréquence d'entrée horloge
EP10 = période (équivalent décimal de EP15-EP0)
CT10 = bits forts période (EP11-EP8)
FT10 = bits faibles période (EP7-EP0)





Au vu de ces équations, chacun comprendra que la fréquence de l'enveloppe peut varier d'un minimum de $\frac{fHORL}{16776960}$ (si EP10= 65535) à un maximum de $\frac{fHORL}{256}$ (si EP10 = 1). En utilisant par exemple un signal d'horloge à 2 Mhz, on dispose donc d'une plage de fréquences d'enveloppe allant de 0,12 Hz à 7812,5 Hz (N.D.T. et rappel: Le ST cadence précisément le circuit son à partir d'un signal d'horloge à 2 MHz).

Pour calculer les valeurs des contenus des registres de réglage rudimentaire et de réglage fin correspondant à un signal d'horloge et à des fréquences sonores donnés, nous inverserons simplement les termes des équations, soit:

$$EP10 = \frac{fHORL}{256fE} \qquad CT10 + \frac{FT10}{256} = \frac{EP10}{256}$$

Exemple 1: fE = 0,5 Hz fHORL = 2MHz

$$EP10 = \frac{2x106}{256(0,5)} = 15625$$

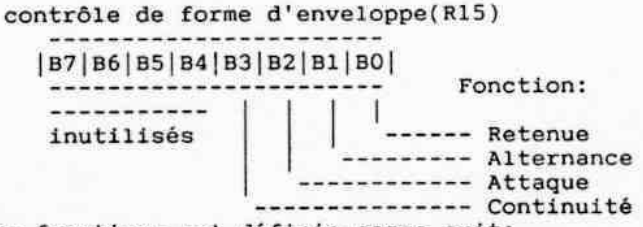
$$\text{d'où: } CT10 + \frac{FT10}{256} = \frac{15625}{256} = 61 - \frac{9}{256}$$

On a donc : CT10 = 61 = %00111101 (B7-B0)
FT10 = 9 = %00001001 (B7-B0)

### 3.5.2. Registre de forme et de type d'enveloppe(R15)

Le générateur d'enveloppe, après division de la fréquence d'enveloppe par 16, génère une matrice d'enveloppe à seize états par cycle de type défini par les 4 bits faibles du registre R15. La spécificité et le déroulement de la forme d'enveloppe sont entièrement fonction de ces 4 bits, lesquels définissent en particulier le caractère cyclique ou non de la forme, à travers un compteur de 4 bits E3 E2 E1 E0.

Chacun des 4 bits faibles du registre R15 contrôle l'une des fonctions du générateur d'enveloppe, selon le schéma suivant:



Chacune des fonctions est définie comme suit:

Retenue : si ce bit est à 1, l'enveloppe est limitée à 1 cycle, à partir du dernier compte du compteur d'enveloppe, soit 0000 ou 1111 selon que le compteur d'enveloppe est en mode décrémentation ou incrémentation respectivement.

Alternance : si ce bit est à 1, le compteur d'enveloppe inverse le sens du comptage (incrémentation-décrémentation) à la fin de chaque cycle.
Note: Lorsque les deux bits 'Retenue' et 'Alternance' sont à 1, le compteur d'enveloppe est réinitialisé à sa valeur de départ avant retenue.

Attaque : si ce bit est à 1, le compteur d'enveloppe sera incrémenté (attaque) de E3 E2 E1 E0 = 0000 jusqu'à E3 E2 E1 E0 = 1111; si ce bit est à 0, le compteur sera décrémenté (déclin) de 1111 jusqu'à 0000.

Continuité : si ce bit est à 1, la matrice de cycle sera définie par le bit de 'Retenue'; si ce bit est à 0, le générateur d'enveloppe sera réinitialisé à 0 après un cycle et maintenu à cette valeur.

Pour décrire précisément les fonctions ci-dessus, il faudrait suivre les séquences de comptage et de décomptage de E3 E2 E1 E0 pour chaque combinaison de 'Retenue', 'Alternance', 'Attaque' et 'Continuité'. Toutefois, étant donné que ces fonctions servent (lorsqu'elles sont sélectionnées par le contrôleur d'amplitude) à moduler les sorties des mélangeurs, une compréhension plus claire peut résulter d'une présentation graphique de leurs effets, à travers chacune des conditions possibles:

Détail de 2 cycles correspondant à une modulation 1010, soit 'Alternance' et 'Continuité' à 1, 'Retenue' et 'Attaque' à 0.

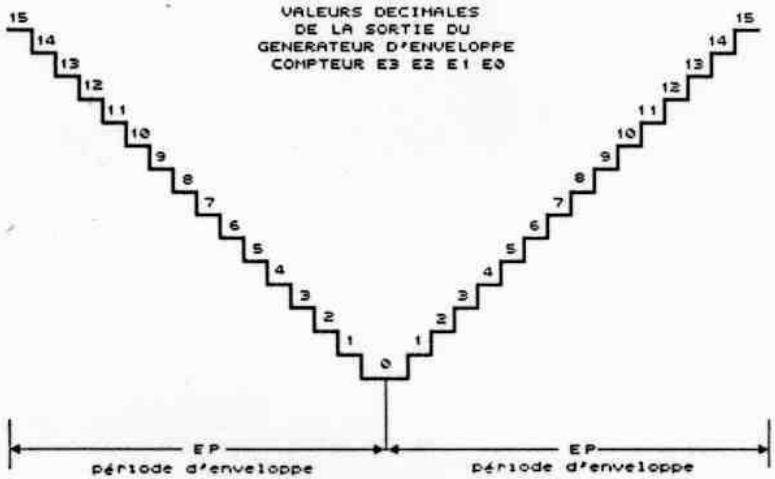







Schéma des différents types d'enveloppe possibles:

| BITS du REGISTRE R13 B3 | B2 | B1 | B0 | REPRESENTATION GRAPHIQUE DES SORTIES DU GENERATEUR D'ENVELOPPE E3 E2 E1 E0 |
|---|---|---|---|---|
| CONTINUITE | ATTAQUE | ALTERNANCE | RETENUE | |
| 0 | 0 | x | x | |
| 0 | 1 | x | x | |
| 1 | 0 | 0 | 0 | |
| 1 | 0 | 0 | 1 | |
| 1 | 0 | 1 | 0 | |
| 1 | 0 | 1 | 1 | |
| 1 | 1 | 0 | 0 | |
| 1 | 1 | 0 | 1 | |
| 1 | 1 | 1 | 0 | |
| 1 | 1 | 1 | 1 | |

## 3.6. Registres de transfert de données (R16,R17)

Les registres R16 et R17 fonctionnent comme zones de stockage temporaire des données entre le bus de données du PSG (DA0-DA7) et les deux ports d'entrée-sortie (ESA7-ESA0 et ESB7-ESB0). Les deux ports sont disponibles sur le circuit AY-3-8910, un seul est disponible sur le AY-3-8912. L'utilisation des registres R16 et R17 pour le stockage temporaire des données est sans effet sur la gestion des sons.

Pour envoyer une donnée à partir du processeur maître vers le port d'entrées-sorties A, seules les étapes décrites ci-dessous sont nécessaires:

1. Verrouillage de l'adresse de R7 (registre d'activation).
2. Ecriture de la donnée sur le bus (bit B6 de R7 placé à 1).
3. Verrouillage de l'adresse de R16 (sélection du port A).
4. Ecriture de la donnée sur le bus (donnée vers le port A).

Pour lire une donnée provenant du port A à partir du processeur maître, les opérations suivantes sont nécessaires:

1. Verrouillage de l'adresse de R7 (registre d'activation).
2. Ecriture de la donnée sur le bus (bit B6 de R7 placé à 0).
3. Verrouillage de l'adresse de R16 (sélection du port A).
4. Lecture de la donnée sur le bus (donnée venant du port A).

Une fois chargé dans l'un des registres de transfert de données R16 ou R17, la donnée à envoyer sur le port y restera tant qu'une autre donnée n'y aura pas été placée ou tant qu'il n'y aura pas de réinitialisation (mise à la masse de la broche Reset) ou de changement de mode écriture/lecture.

On notera aussi qu'en mode lecture, le contenu des registres R16 et R17 reflétera les signaux relevés sur les ports d'entrée-sortie. Toutefois, l'envoi de ces données vers le processeur maître nécessitera une opération de lecture selon le protocole décrit ci-dessus.



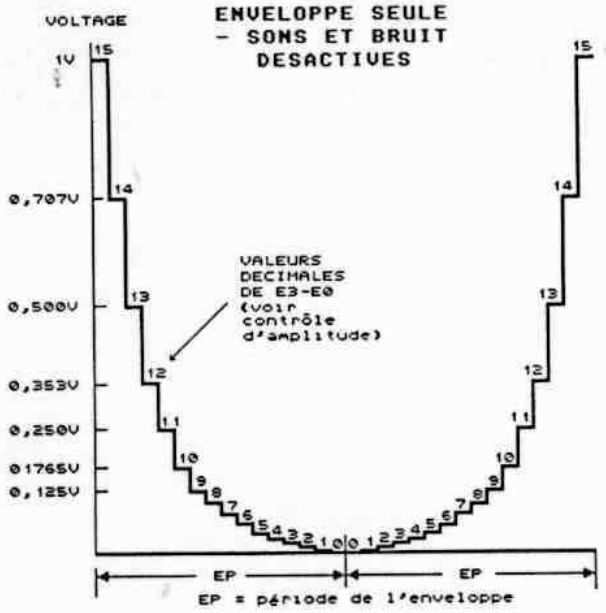


## 3.7. Fonctionnement des convertisseurs numérique-analogique

Etant donné que la fonction principale du PSG soit de produire des sons pour le mécanisme de détection d'onde fortement imparfait qu'est l'oreille humaine, la conversion du numérique à l'analogique est effectuée par pas selon une échelle logarithmique selon un voltage variant entre 0 et 1 Volt. Le contrôle de l'amplitude spécifique de chacun des convertisseurs N/A est réalisé par les trois quartets (4 bits) du bloc de contrôle d'amplitude, les mélangeurs fournissant le signal de fréquence de base (son ou/et bruit).

La figure ci-dessous décrit les sorties des convertisseurs N/A résultant d'une désactivation des générateurs de son et de bruit, avec une amplitude d'enveloppe variable.

Sorties des convertisseurs numérique-analogique:



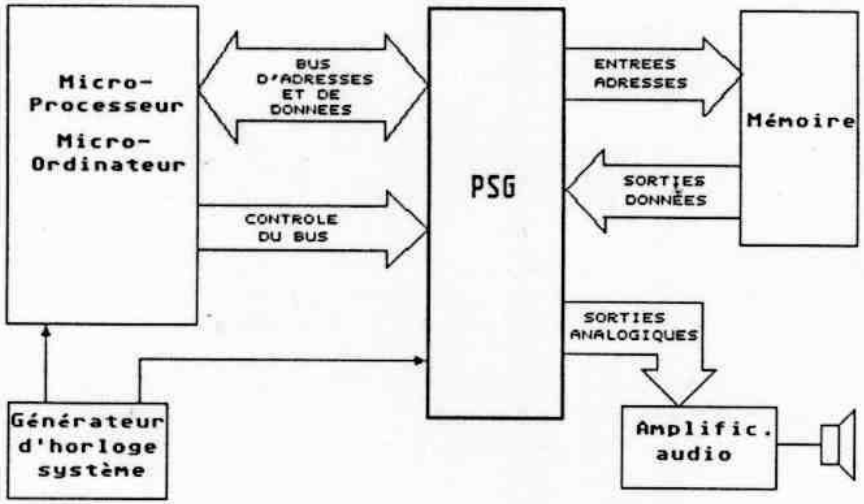
## 4. Interfaçage

### 4.1. Introduction

Etant donné que le PSG AY-3-8910 doit être utilisé en association avec d'autres composants, l'interfaçage du circuit est une fonction indispensable. Le PSG est conçu pour être contrôlé par un microprocesseur ou un micro-ordinateur et contrôler lui-même les circuits de sortie analogique audio. Il constitue le lien entre le processeur et un haut-parleur pour la génération de sons ou d'effets sonores résultant d'entrées numériques.

Les paragraphes suivants fournissent des exemples et des illustrations de la méthode extrêmement simple d'interfaçage du PSG avec un microprocesseur ou un micro-ordinateur.

DIAGRAMME FONCTIONNEL DU SYSTEME D'INTERFACAGE:



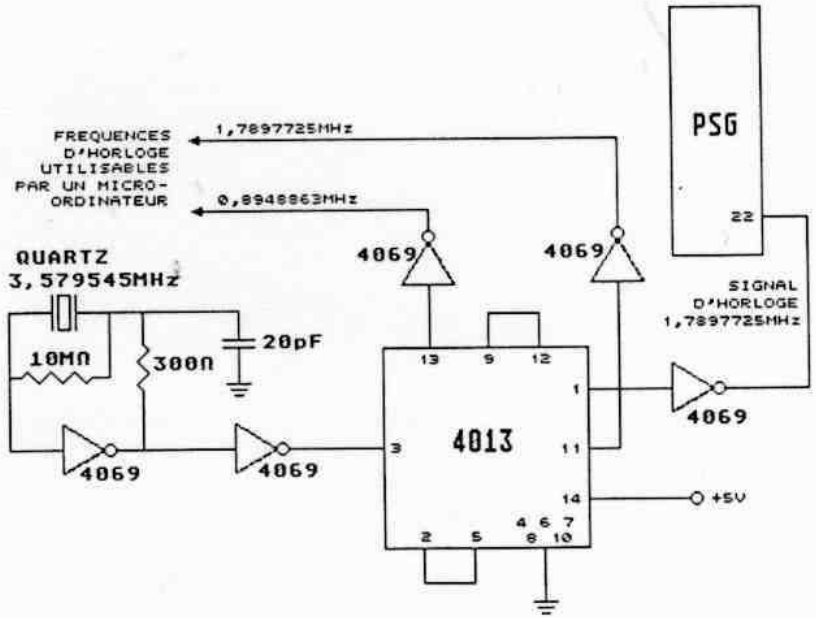





### 4.2. Entrée horloge

Une solution économique d'implémentation du signal d'horloge sur l'entrée du PSG est fournie en figure ci-dessous. Elle est constituée d'un quartz standard à 3,579545MHz, d'un inverseur CMOS CD4069 et d'un circuit CD4013 destiné à diviser la fréquence d'horloge par deux. Le signal d'horloge à l'entrée du PSG est donc cadencé à 1,7897725MHz. En fonction du micro-ordinateur utilisé, cette horloge pourra être fixée à une valeur donnée.

GENERATION DE L'HORLOGE:



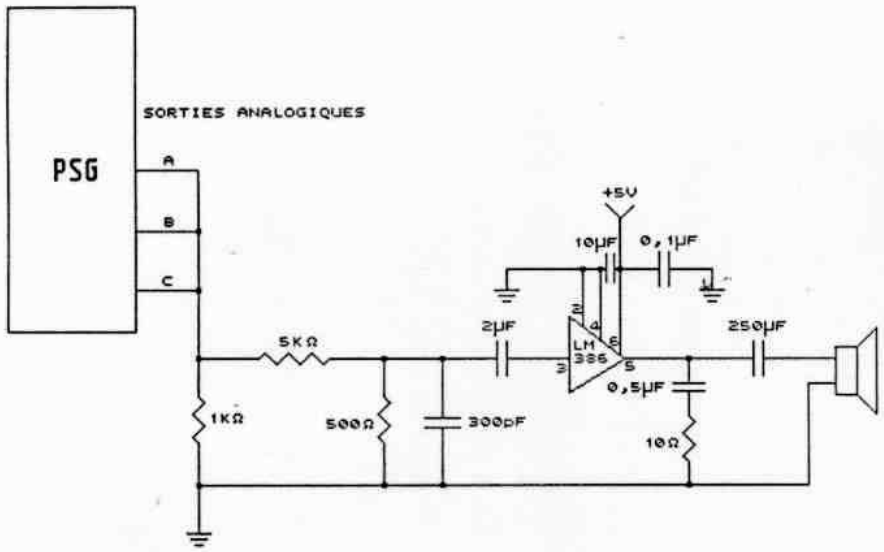

N.D.T.: Sur le ST, le signal d'horloge est fourni au circuit AY-3-8910 par un quartz à 2MHz.

## 4.3. Interface de sortie audio

La figure ci-dessous illustre l'interfaçage du PSG avec un amplificateur audio LM386 disponible sur le marché. On constate que les canaux A, B et C sont reliés ensemble afin de générer des ondes complexes et amplifiés par un simple amplificateur externe. Ces canaux peuvent également être amplifiés séparément afin de produire des sons plus sophistiquées.

Chaque canal de sortie est contrôlé individuellement par un registre d'amplitude (R10, R11, R12) et par un registre d'activation (R7).

INTERFACE DE SORTIE AUDIO:








## 4.4. Accès à des mémoires externes

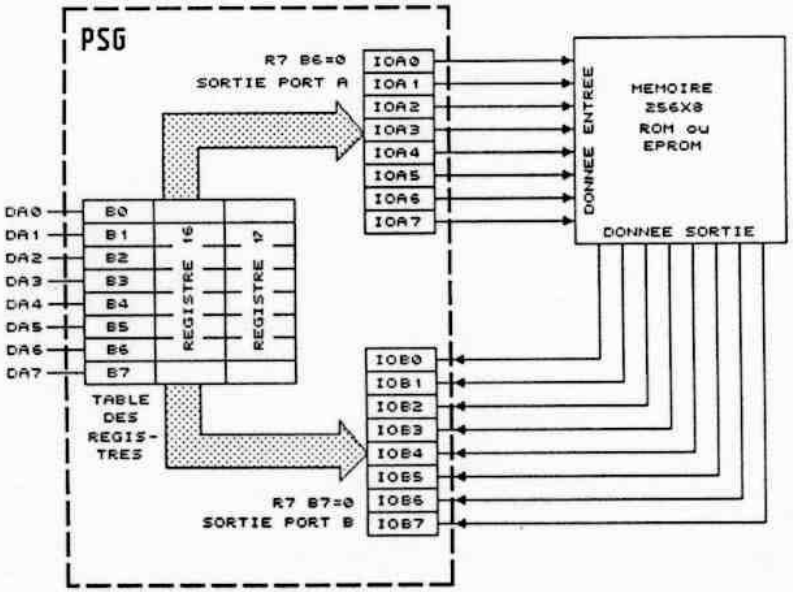
La connection de ROM ou de PROM, illustrée par le schéma ci-dessous, est l'une des possibilités du PSG pour fournir plus d'informations au processeur maître. Les deux registres d'entrée-sortie du PSG servent dans ce cas à adresser la mémoire à travers le port A (8 bits) et à lire les données en mémoire à travers le port B (8 bits).

Voici un exemple de séquence de contrôle permettant l'adressage et la lecture d'une mémoire externe connectée aux ports A et B comme indiqué sur la figure ci-dessous:

| Contrôle du Bus | BDIR | BC2 | BC1 | état du Bus (DA7-DA0) |
|---|---|---|---|---|
| Verrouillage adresse | 1 | 1 | 1 | 00000111 Verrouille R7 |
| Ecriture vers PSG | 1 | 1 | 0 | 01000000 B7=0, B6=1 |
| Verrouillage adresse | 1 | 1 | 1 | 00001110 Verrouille R16 |
| Ecriture vers PSG | 1 | 1 | 0 | 00000001 Adresse donnée |
| Verrouillage adresse | 1 | 1 | 1 | 00001111 Verrouille R17 |
| Lecture de la donnée | 0 | 1 | 1 | XXXXXXXX Contenu donnée |

Des mémoires vives peuvent également être connectées, au lieu de mémoires mortes. Le port B servira alors à lire les mémoires en entrée et à les écrire en sortie.

ACCES MEMOIRE EXTERNE



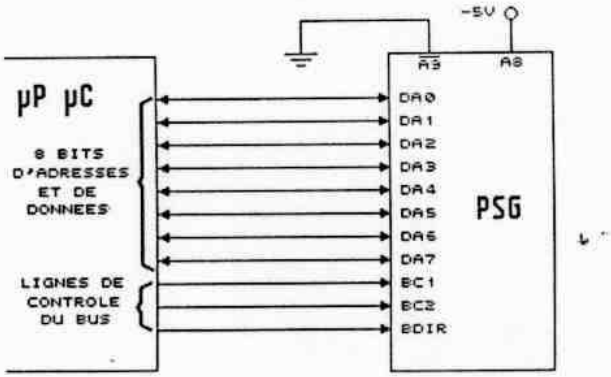
## 4.5. Interfaçage avec un microprocesseur

Dans la figure ci-dessous, les lignes notées DA7 à DA0 correspondent aux bits d'entrée-sortie 7-0 du bus. Ce bus de 8 bits sert au transfert de toutes les données et adresses entre l'AY-3-8910/8912 et le processeur.

BC1, BC2 et BDIR sont les signaux de contrôle du bus générés par le processeur pour la commande des opérations sur le bus. Ces opérations sont le verrouillage d'adresse, l'écriture vers le PSG, la lecture vers le PSG et la désactivation.

Les sections suivantes détaillent les interfaces spécifiques à de nombreux microprocesseurs connus.

INTERFACAGE AVEC UN MICROPROCESSEUR








## 4.6. Interfaçage avec le PIC 1650

Le schéma en fin de cette section illustre l'interfaçage d'un AY-3-8910 avec un contrôleur principal, le PIC1650. Ce processeur sert à scruter le clavier, à aller lire les PROMs, à écrire des données vers l'AY-3-8910 et à fournir la base de temps à ce circuit.

L'interfaçage s'effectue directement, le PIC 1650 et l'AY-3-8910 opérant avec des niveaux de voltage identiques.

Ce schéma particulier montre comment un micro-ordinateur avec des mémoires additionnelles peut produire une musique réelle et des effets sonores.

Les connections d'interfaçage s'effectuent directement entre les broches de sortie du 1650 et les broches BDIR, BC1 et BC2. Le logiciel a ensuite la responsabilité de contrôler ces signaux afin d'ordonner au PSG le verrouillage d'adresse, l'écriture ou la lecture.

Les routines listées ci-après fournissent le code d'un exemple d'utilisation simple. Cet exemple illustre les capacités du PSG tant en ce qui concerne la musique que les effets sonores et le contrôle d'entrées-sorties. On notera qu'en général ces routines opèrent le verrouillage d'adresse avant chaque lecture.

Le programme de "LECTURE DE ROM" illustre l'utilisation de routines de lecture et écriture pour accéder au monde extérieur via le PSG.

### 4.6.1. ROUTINE D'ECRITURE DE DONNEES

```
 80                ECRITURE DU 1650 VERS LE 8910
 81                ADRESSE DU REGISTRE DU 8910 DANS ADRESSE
 82                DONNEE A ECRIRE DANS DONNEE
 83  024   0066 WRIT1 MOVWF ADDRES
 84  025   1026 WRITE MOVF  ADDRES.W PREND NUMERO DU REGISTRE
 85  026   0045       MOVWF IOA      PLACE L'ADRESSE
 86  027   1006       MOVF  IOB.W    SAISIT ETAT BC1,BC2,BDIR,ETC
 87  030   7370       ANDLW 370
 88  031   6404       IORLW 4        FIXE CONTENU
 89  032   0046       MOVWF IOB      ENVOI CONTENU
 90  033   7370       ANDLW 370
 91  034   0046       MOVWF IOB      ENVOI DE NACT (DESACTIVATION)
 92  035   1027       MOVF  DATA.W   DONNEE
 93  036   0045       MOVWF IOA      PLACE DONNEE SUR LE BUS
 94  037   1006       MOVF  IOB.W
 95  040   7370       ANDLW 370
 96  041   6406       IORLW 6
 97  042   0046       MOVWF IOB      ENVOI DE DWS (ECRITURE)
 98  043   7370       ANDLW 370      PLACE CODE DESACTIVATION
 99  044   0046       MOVWF IOB      ENVOI DE NACT (DESACTIVATION)
100  045   4000       RET            RETOUR A LA ROUTINE D'APPEL
```

## 4.6.2. ROUTINE DE LECTURE DE DONNEE

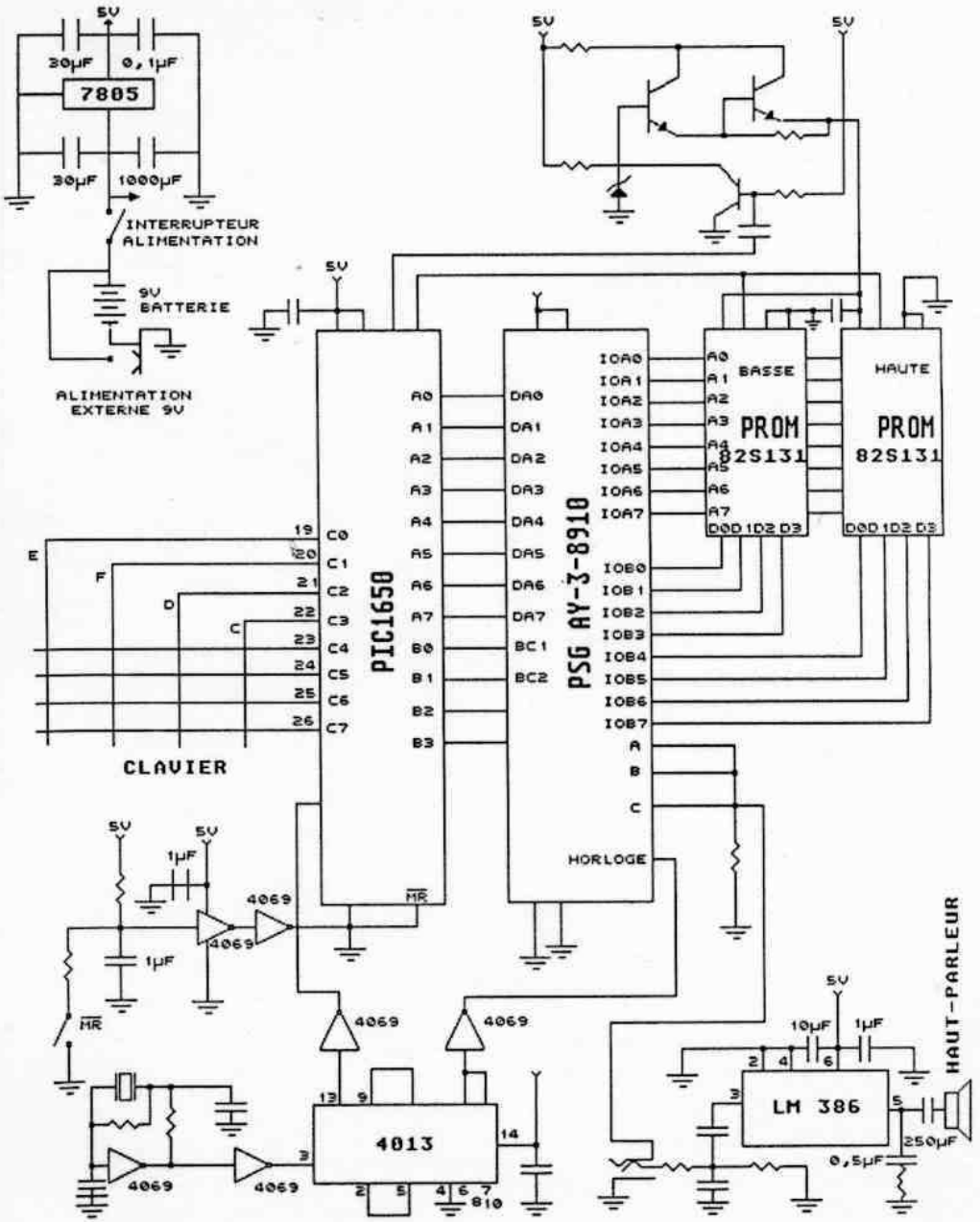
```
51                ADRESSE DE LA DONNEE A LIRE DANS REGISTRE D'ADRESSE
52                APRES LECTURE, ENTREE DE LA DONNEE DANS REGISTRE
53                ENTREE READ1 SUPPOSE QUE LE N° DU REGISTRE EST
54                DANS W
55   000  0066 READ1 MOVWF ADDRES    STOCKAGE DE L'ADRESSE
56   001  1026 READ  MOVF  ADDRES.W SAISIT NUMERO DE REGISTRE
57   002  0045       MOVWF IOA       L'ENVOIE SUR BUS DU 8910
58   003  1006       MOVF  IOB.W     DEMANDE ETAT BDIR,BC1,BC2
59   004  6404       IORLW 4         FIXE CONTENU
60   005  0046       MOVWF IOB       ENVOI CONTENU
61   006  7370       ANDLW 370
62   007  0046       MOVWF IOB       ENVOI NACT (DESACTIVATION)
63   010  6377       MOVLW 377
64   011  0045       MOVWF IOA       FIXE POUR LECTURE
65   012  1006       MOVF  IOB.W
66   013  7370       ANDLW 370
67   014  6403       IORLW 3         FIXE DTB
68   015  0046       MOVWF IOB       ENVOI DTB (LECTURE)
69   016  1005       MOVF  IOA.W
70   017  0067       MOVWF DATA      SAUVE LA DONNEE
71   020  1006       MOVF  IOB.W
72   021  7370       ANDLW 370
73   022  0046       MOVWF IOB       ENVOI NACT (DESACTIVATION)
74   023  4000       RET             RETOUR A LA ROUTINE D'APPEL
```

## 4.6.3. ROUTINE DE LECTURE DE ROM

```
106               ADRESSE EN ROM DANS W EN ENTREE DE NXROM
107               ADRESSE EN ROM DANS ROMAD EN ENTREE DE ROMRD
108
109               INCREMENTE ROMAD APRES LECTURE. SI L'ADRESSE EN
110               ROM DEPASSE 256, SELECTIONNE LA PLAGE DE MEMOIRE
111               SUPERIEURE
112               UTILISE LE REGISTRE 16 DU 8910 POUR L'ADRESSAGE
113               ET LE REGISTRE 17 POUR LA LECTURE DE LA DONNEE
114  046  1033 NXROM MOVF   ROMAD.W
115  047  0067 ROMRD MOVWF  DATA      PLACE L'ADRESSE
116  050  6016       MOVLW  16        ADRESSE REGISTRE PORT A
117  051  0066       MOVWF  ADDRES
118  052  2306       BCF    IOB6      ROM ACTIVE
119  053  4425       CALL   WRITE     ENVOI VERS PORT A
120  054  1266       INCF   ADDRES    ADRESSE VERS PORT B
121  055  4401       CALL   READ      LECTURE DE LA DONNEE
122  056  2706       BSF    IOB6      ROM DESACTIVEE
123  057  1770       INCFSZ ROMAD     INCREMENTE ADRESSE
124  060  4000       RET
125  061  2646       BSF    IOB5      SELECTION PLAGE HAUTE
126  062  4000       RET              RETOUR A LA ROUTINE D'APPEL
```







## 4.7 Interfaçage avec le CP1600/1610

Comme on peut le constater sur le schéma ci-dessous, l'interfaçage entre le circuit AY-38910 et un microprocesseur CP1600 ou CP1610 s'effectue de façon directe. Les niveaux d'entrée et de sortie étant compatibles, aucun convertisseur de niveau n'est utile. Même la nature des signaux de contrôle est identique pour les deux circuits, ce qui autorise une connectique simple.

Le CP1600/CP1610 agit comme un contrôleur dans cette configuration, saisissant les données à partir de ROMs. Le CP1600/1610 agit également comme contrôleur du bus et fournit la base de temps nécessaire au circuit AY-3-8910.

### 4.7.1. ROUTINE D'ECRITURE DE DONNEE

Le sous-programme nécessaire pour écrire dans un registre donné a la forme suivante :

```
     MVI   valeur,R0;  place la valeur à écrire
     MVO   R0,Reg;     écrit dans le registre
```

La routine de chargement d'une même valeur dans tous les registres s'écrira ainsi:

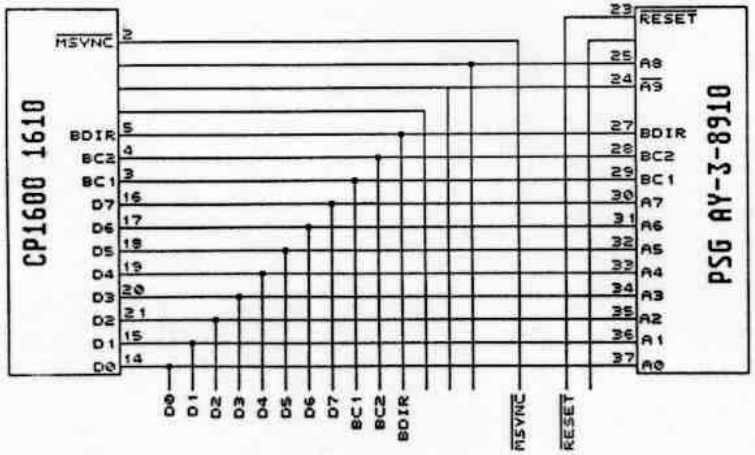
```
     MVII  Reg0,R4
     CLRR  R0
bcle MVO@  R0,R4
     CMPI  Reg0+17,R4
     BLT   bcle
```

### 4.7.2. ROUTINE DE LECTURE DE DONNEE

La routine permettant de lire dans un registre est:

```
     MVI   Reg,R0;     saisit la donnée du registre dans R0
     MVO   R0,valeur;  la stocke en mémoire
```

SCHEMA D'INTERFACAGE DU CP1600/1610 AVEC LE AY-3-8910








## 4.8. Interfaçage au M6800

Un microprocesseur M6800 peut être interfacé avec un circuit AY-3-8910/8912 par l'intermédiaire d'un circuit PIA M6820 ("Peripheral Interface Adapter"). Les huit lignes PA0 à PA7 du port d'entrées-sorties A du PIA servent de bus et les trois lignes PB0 à PB2 du port d'entrées-sorties B du PIA servent de lignes de contrôle du bus. Les routines listées ci-dessous servent à contrôler le verrouillage d'adresse, l'écriture et la lecture de données sur le AY-3-8910/8912.

### 4.8.1. ROUTINE DE VERROUILLAGE D'ADRESSE

```
;EN ENTREE, L'ACCUMULATEUR B A LA VALEUR DE L'ADRESSE
;
LATCH
     CLRA      ;VIDE L'ACCUMULATEUR DE TRAVAIL A
     STAA 8005 ;STOCKAGE DE A(0) DANS PIACRA (ACCES A CRA)
     LDAA #$FF ;CHARGE A AVEC $FF (DECLARATION DE PA0-PA7 EN
     STAA 8004 ;SORTIE), ACCES A PIADDA
     LDAA #4   ;MOT DE CONTROLE A
     STAA 8005 ;STOCKAGE DANS PIACRA (ACCES A CRA)
     STAB 8004 ;STOCKAGE DE L'ADRESSE DANS PIADDA
     STAA 8006 ;STOCKAGE DU MOT DE CONTROLE DANS PIADDB
     CLRA      ;MOT DE CONTROLE NUL POUR ACCES EN ENTREE
     STAA 8006 ;VERROUILLAGE DE L'ADRESSE VIA LE PIADDB
     RTS       ;RETOUR A LA ROUTINE D'APPEL
```

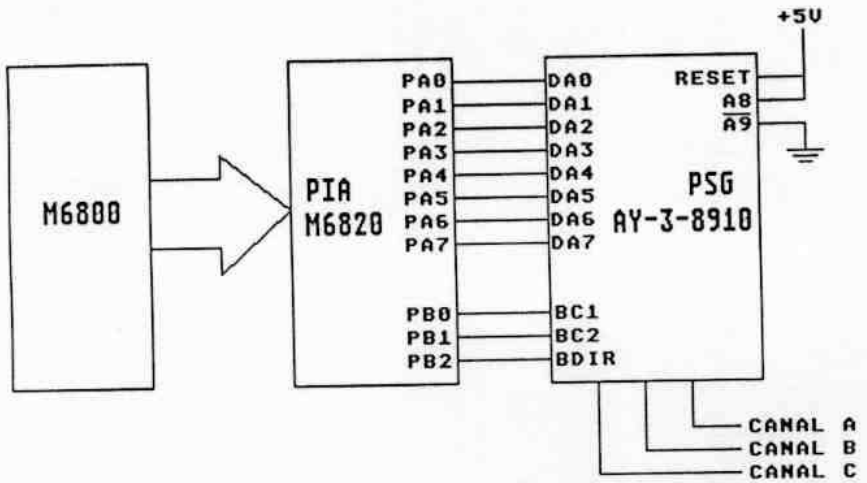
### 4.8.2. ROUTINE D'ECRITURE DE DONNEE

```
;EN ENTREE, L'ACCUMULATEUR B CONTIENT LA DONNEE A ECRIRE
;
WRITE
     STAB 8004 ;STOCKE LA DONNEE DANS PIADDA
     LDAA #6   ;MOT DE CONTROLE (POUR COMMANDE ECRITURE)
     STAA 8006 ;STOCKAGE DANS PIADDB (LIGNES DE COMMANDE)
     CLRA      ;POUR ACCES EN ENTREE
     STAA 8006 ;STOCKAGE DANS PIADDB (ECRITURE DE LA DONNEE)
     RTS       ;RETOUR A LA ROUTINE D'APPEL
```

### 4.8.3. ROUTINE DE LECTURE DE DONNEE

```
;APRES LECTURE, LA DONNEE EST RETOURNEE DANS L'ACCUMULATEUR B
READ
     CLRA      ;POUR ACCES EN ENTREE
     STAA 8005 ;ACCES A DDRA PAR PIACRA
     STAA 8004 ;DECLARATION DE PA0 A PA7 EN ENTREE
     LDAA #4   ;MOT DE CONTROLE (POUR COMMANDE LECTURE)
     STAA 8005 ;STOCKAGE DANS PIACRA (ACCES A ORA)
     DECA      ;A=$FF, POUR ACCES EN SORTIE
     STAA 8006 ;DECLARATION DE PB0 A PB2 EN SORTIE
     LDAB 8004 ;CHARGE B AVEC CONTENU DE PIADDA
     CLRA      ;POUR ACCES EN ENTREE
     STAA 8006 ;DECLARATION DE PB0 A PB2 EN ENTREE
     RTS       ;RETOUR A LA FONCTION D'APPEL
```

INTERFACAGE DU M6800 AVEC LE AY-3-8910




N.D.T.: RAPPELS SUR LE FONCTIONNEMENT DU PIA6820.

Circuit d'interface travaillant en mode parallèle, le PIA 6820 comporte 6 registres adressables par le microprocesseur M6800 en écriture et en lecture:

| | |
|---|---|
| CRA | contient les paramètres de contrôle |
| CRB | contient les paramètres de contrôle |
| DDRA | contient un mot définissant le sens du transfert |
| DDRB | des lignes (entrée ou sortie) |
| ORA | stockage des données en sortie lors de l'écriture. |
| ORB | |

Ces six registres ne sont pas directement adressables. En fait CRA et CRB sont directement adressables mais ORA et DDRA d'une part, ORB et DDRB d'autre part sont sélectionnables en fonction du bit 2 de CRA ou CRB préalablement écrit. Si le bit 2 est à 1 c'est ORA, ou ORB qui est sélectionné. S'il est nul, c'est DDRA, respectivement DDRB, qui est sélectionné.

Chacune des lignes peut être programmée en entrée ou en sortie, le sens du transfert dépendant du contenu de DDRA, DDRB.





## 4.9. Interfaçage avec le bus S100 du 8080

Le bus S100 a été conçu pour autoriser des opérations de lecture ou d'écriture entre le 8080 et un autre circuit par le biais des instructions élémentaires 'IN' et 'OUT'. Un aspect intéressant de cette norme est qu'elle permet le raccordement de plusieurs périphériques PSG sur un simple bus. Le système décrit ci-dessous permet précisément de relier deux PSG, chacun se voyant affecté un canal stéréo.

Comme les routines suivantes de lecture et écriture de donnée le laissent à penser, le protocole de communication entre le 8080 et un PSG est réduit à sa plus simple expression/

### 4.9.1. ROUTINE DE VERROUILLAGE D'ADRESSE

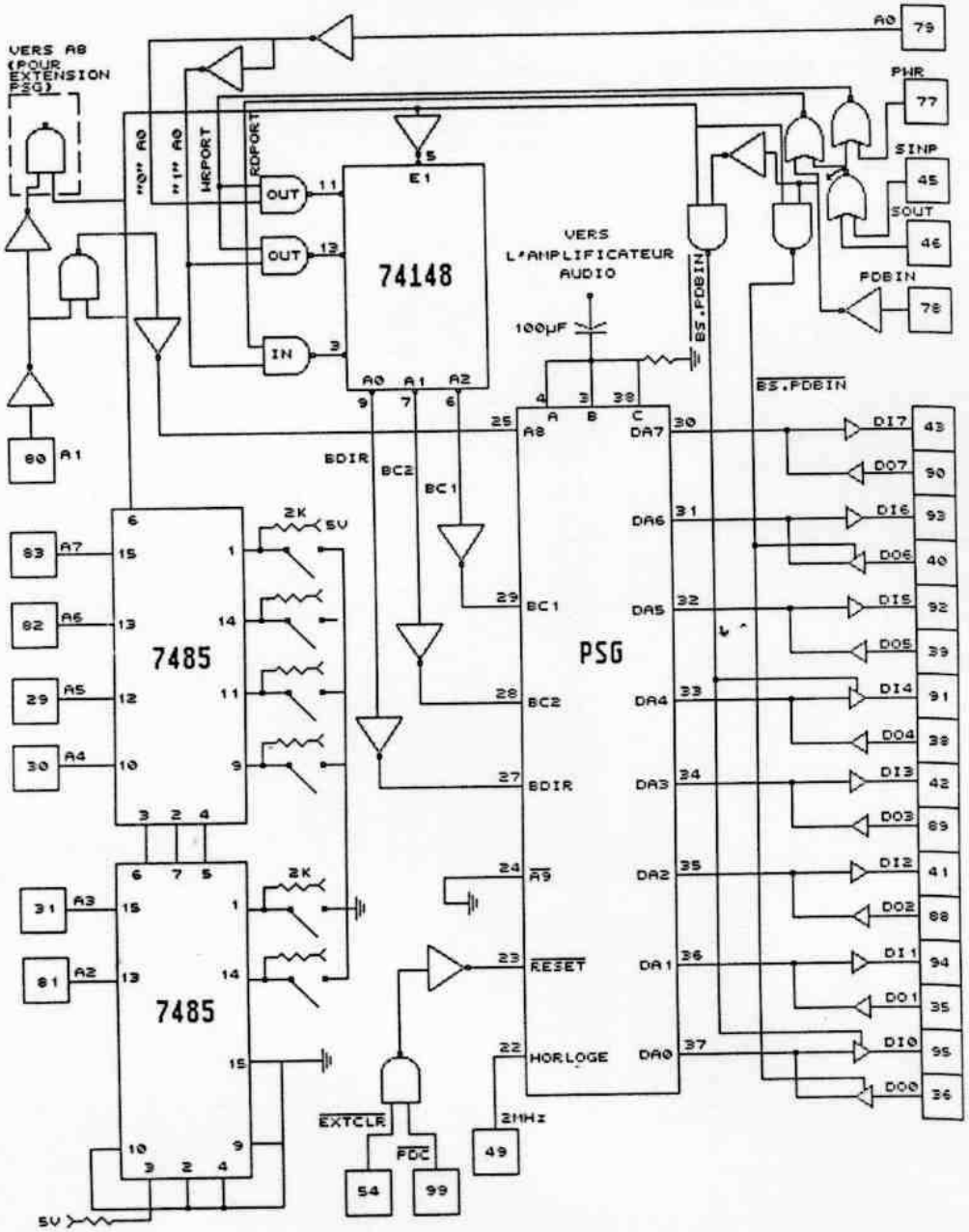
```
PORTADR   EQU  80H    ;ADRESSE TRANSFEREE A L'ADRESSE DU PORT
PORTDONN  EQU  81H    ;DONNEE TRANSFEREE A L'ADRESSE DU PORT
;
;CETTE ROUTINE TRANSFERE LE CONTENU DU REGISTRE C DU 8080
;VERS LE REGISTRE D'ADRESSE DU PSG
PSGBAR
     MOV  A,C         ;PLACE C DANS A POUR SORTIE SUR LE PORT
     OUT  PORTADR     ;ENVOI A L'ADRESSE DU PORT
     RET              ;RETOUR A LA ROUTINE D'APPEL
```

### 4.9.2. ROUTINE D'ECRITURE DE DONNEE

```
;
;ROUTINE ECRIVANT LE CONTENU DU REGISTRE B DU 8080
;DANS LE REGISTRE DU PSG SPECIFIE PAR LE REGISTRE C DU 8080
;
PSGWRITE
     CALL PSGBAR      ;VERROUILLAGE DE L'ADRESSE
     MOV  A,B         ;PLACE CONTENU DE B DANS A POUR TRANSFERT
     OUT  PORTDONN    ;PLACE DONNEE A L'ADRESSE DU PORT
```

### 4.9.3. ROUTINE DE LECTURE DE DONNEE

```
;
;ROUTINE PERMETTANT DE LIRE LE REGISTRE DU PSG SPECIFIE PAR LE
;REGISTRE C DU 8080 ET RETOURNANT LA DONNEE DANS LE REGISTRE B
;DU 8080
;
PSGREAD
     CALL PSGBAR      ;VERROUILLAGE DE L'ADRESSE
     IN   PORTDONN    ;SAISIT DONNEE DANS LE REGISTRE
     MOV  B,A         ;PLACE A DANS B, RESULTAT DU TRANSFERT
     RET
```








## 5. Production musicale

La production musicale suppose la création de séries de fréquences qui soient agréables à l'oreille humaine (sans aborder ici de critère qualitatif). Cela implique essentiellement des relations mathématiques, domaine privilégié pour un périphérique numérique. Ainsi, le passage à l'octave inférieure ou supérieure se fait par division ou multiplication par 2, autrement dit un simple décalage pour la plupart des microprocesseurs.

Un autre aspect de la production musicale est l'aspect "communication". Le compositeur doit pouvoir mettre en oeuvre ses idées de mélodies de la même façon qu'un musicien ou un orchestre peut reproduire les idées du compositeur - souvent sur des instruments extrêmement différents. Cela suppose la possibilité d'accorder les instruments autour d'un jeu standard de fréquences et selon un tempo donné. La fréquence d'accord la plus communément utilisée est basée sur le LA de la troisième octave (fréquence de 440Hz) dans la gamme chromatique.

Bien qu'il soit possible de composer des airs reconnaissables en utilisant une seule note à la fois, la faculté de pouvoir produire plusieurs sons simultanés (musique polyphonique, choeurs et contrepoints) contribue à accroître fortement le niveau de qualité sonore. Cette possibilité est offerte par le PSG puisque trois canaux sonores peuvent être programmés simultanément.

### 5.1. Génération de notes

Les notes correspondant à des sons de fréquences données maintenus durant un temps donné, le PSG les reproduit par une série de chargement de registres. Généralement la méthode utilisée consiste à calculer les valeurs à charger dans les registres pour les notes de la première octave, les valeurs correspondant aux octaves suivantes s'en déduisant par décalages (multiplications).

La table fournie dans les deux pages qui suivent constitue la table de conversion pour 8 octaves de notes du DO de la première octave (32,703Hz) au SI de la huitième octave(7902,080Hz). Cette table a été élaborée sur la base d'un signal d'horloge à 2MHz et en appliquant les formules fournies en section 3.1 de ce chapitre pour le calcul du contenu des registres de période sonore. De par la nature du diviseur interne du PSG, les fréquences basses sont reproduites particulièrement fidélement par le PSG, les fréquences hautes étant plus approximatives. Vous pouvez le constater en comparant dans la table la "fréquence idéale" à la "fréquence réelle", les fréquences idéales correspondant à la gamme chromatique et les fréquences réelles étant les valeurs les plus proches obtenues par le PSG.

N.D.T.: La table de conversion d'origine,fournie par le constructeur, correspondait à une fréquence de 1,78977MHz. Le ST appliquant au circuit son une fréquence d'horloge de 2MHz, les éléments de cette table ont été recalculés par les traducteurs. Les valeurs des registres sont fournies en HEXADECIMAL.

| NOTE | OCTAVE | FREQ.IDEALE | FREQ.REELLE | REG. 4 bits | REG. 8 bits |
|---|---|---|---|---|---|
| DO | 1 | 32 703 | 32 705 | 0xE | 0xEE |
| DO# | 1 | 34 648 | 34 645 | 0xE | 0x17 |
| RE | 1 | 36 708 | 36 710 | 0xD | 0x4D |
| RE# | 1 | 38 891 | 38 892 | 0xC | 0x8E |
| MI | 1 | 41 203 | 41 200 | 0xB | 0xDA |
| FA | 1 | 43 654 | 43 660 | 0xB | 0x2F |
| FA# | 1 | 46 249 | 46 245 | 0xA | 0x8F |
| SOL | 1 | 48 999 | 49 000 | 0x9 | 0xF7 |
| SOL# | 1 | 51 913 | 51 910 | 0x9 | 0x68 |
| LA | 1 | 55 000 | 54 993 | 0x8 | 0xE1 |
| LA# | 1 | 58 270 | 58 275 | 0x8 | 0x61 |
| SI | 1 | 61 735 | 61 728 | 0x7 | 0xE9 |
| DO | 2 | 65 406 | 65 410 | 0x7 | 0x77 |
| DO# | 2 | 69 296 | 69 290 | 0x7 | 0x0C |
| RE | 2 | 73 416 | 73 400 | 0x6 | 0xA7 |
| RE# | 2 | 77 782 | 77 785 | 0x6 | 0x47 |
| MI | 2 | 82 406 | 82 399 | 0x5 | 0xED |
| FA | 2 | 87 308 | 87 291 | 0x5 | 0x98 |
| FA# | 2 | 92 498 | 92 524 | 0x5 | 0x47 |
| SOL | 2 | 97 998 | 97 962 | 0x4 | 0xFC |
| SOL# | 2 | 103 826 | 103 821 | 0x4 | 0xB4 |
| LA | 2 | 110 000 | 110 035 | 0x4 | 0x70 |
| LA# | 2 | 116 540 | 116 496 | 0x4 | 0x31 |
| SI | 2 | 123 470 | 123 518 | 0x3 | 0xF4 |
| DO | 3 | 130 812 | 130 753 | 0x3 | 0xBC |
| DO# | 3 | 138 592 | 138 581 | 0x3 | 0x86 |
| RE | 3 | 146 832 | 146 886 | 0x3 | 0x53 |
| RE# | 3 | 155 564 | 155 473 | 0x3 | 0x24 |
| MI | 3 | 164 812 | 164 908 | 0x2 | 0xF6 |
| FA | 3 | 174 616 | 174 581 | 0x2 | 0xCC |
| FA# | 3 | 184 996 | 184 911 | 0x2 | 0xA4 |
| SOL | 3 | 195 996 | 195 925 | 0x2 | 0x7E |
| SOL# | 3 | 207 652 | 207 641 | 0x2 | 0x5A |
| LA | 3 | 220 000 | 220 070 | 0x2 | 0x38 |
| LA# | 3 | 233 080 | 233 209 | 0x2 | 0x18 |
| SI | 3 | 246 940 | 247 036 | 0x1 | 0xFA |
| DO | 4 | 261 624 | 261 506 | 0x1 | 0xDE |
| DO# | 4 | 277 184 | 277 162 | 0x1 | 0xC3 |
| RE | 4 | 293 664 | 293 427 | 0x1 | 0xAA |
| RE# | 4 | 311 128 | 310 945 | 0x1 | 0x92 |
| MI | 4 | 329 624 | 329 815 | 0x1 | 0x7B |
| FA | 4 | 349 232 | 349 162 | 0x1 | 0x66 |
| FA# | 4 | 369 992 | 369 822 | 0x1 | 0x52 |
| SOL | 4 | 391 992 | 391 850 | 0x1 | 0x3F |
| SOL# | 4 | 415 304 | 415 282 | 0x1 | 0x2D |
| LA | 4 | 440 000 | 440 141 | 0x1 | 0x1C |
| LA# | 4 | 466 160 | 466 418 | 0x1 | 0x0C |
| SI | 4 | 493 880 | 494 071 | 0x0 | 0xFD |





| NOTE | OCTAVE | FREQ.IDEALE | FREQ.REELLE | REG.4 bits | REG.8 bits |
|---|---|---|---|---|---|
| DO | 5 | 523 248 | 523 013 | 0x0 | 0xEF |
| DO# | 5 | 554 368 | 555 556 | 0x0 | 0xE1 |
| RE | 5 | 587 328 | 586 856 | 0x0 | 0xD5 |
| RE# | 5 | 622 256 | 621 891 | 0x0 | 0xC9 |
| MI | 5 | 659 248 | 657 895 | 0x0 | 0xBE |
| FA | 5 | 698 464 | 698 324 | 0x0 | 0xB3 |
| FA# | 5 | 739 984 | 739 645 | 0x0 | 0xA9 |
| SOL | 5 | 783 984 | 786 164 | 0x0 | 0x9F |
| SOL# | 5 | 830 608 | 833 333 | 0x0 | 0x96 |
| LA | 5 | 880 000 | 880 281 | 0x0 | 0x8E |
| LA# | 5 | 932 320 | 932 836 | 0x0 | 0x86 |
| SI | 5 | 987 760 | 984 252 | 0x0 | 0x7F |
| DO | 6 | 1046 496 | 1050 420 | 0x0 | 0x77 |
| DO# | 6 | 1108 736 | 1106 195 | 0x0 | 0x71 |
| RE | 6 | 1174 656 | 1179 245 | 0x0 | 0x6A |
| RE# | 6 | 1244 512 | 1250 000 | 0x0 | 0x64 |
| MI | 6 | 1318 496 | 1315 789 | 0x0 | 0x5F |
| FA | 6 | 1396 928 | 1404 494 | 0x0 | 0x59 |
| FA# | 6 | 1479 968 | 1488 095 | 0x0 | 0x54 |
| SOL | 6 | 1567 968 | 1562 500 | 0x0 | 0x50 |
| SOL# | 6 | 1661 216 | 1666 667 | 0x0 | 0x4B |
| LA | 6 | 1760 000 | 1760 563 | 0x0 | 0x47 |
| LA# | 6 | 1864 640 | 1865 672 | 0x0 | 0x43 |
| SI | 6 | 1975 520 | 1984 127 | 0x0 | 0x3F |
| DO | 7 | 2092 992 | 2083 333 | 0x0 | 0x3C |
| DO# | 7 | 2217 472 | 2232 143 | 0x0 | 0x38 |
| RE | 7 | 2349 312 | 2358 491 | 0x0 | 0x35 |
| RE# | 7 | 2489 024 | 2500 000 | 0x0 | 0x32 |
| MI | 7 | 2636 992 | 2659 574 | 0x0 | 0x2F |
| FA | 7 | 2793 856 | 2777 778 | 0x0 | 0x2D |
| FA# | 7 | 2959 936 | 2976 190 | 0x0 | 0x2A |
| SOL | 7 | 3135 936 | 3125 000 | 0x0 | 0x28 |
| SOL# | 7 | 3322 432 | 3289 474 | 0x0 | 0x26 |
| LA | 7 | 3520 000 | 3472 222 | 0x0 | 0x24 |
| LA# | 7 | 3729 280 | 3676 471 | 0x0 | 0x22 |
| SI | 7 | 3951 040 | 3906 250 | 0x0 | 0x20 |
| DO | 8 | 4185 984 | 4166 667 | 0x0 | 0x1E |
| DO# | 8 | 4434 944 | 4464 286 | 0x0 | 0x1C |
| RE | 8 | 4698 624 | 4629 630 | 0x0 | 0x1B |
| RE# | 8 | 4978 048 | 5000 000 | 0x0 | 0x19 |
| MI | 8 | 5273 984 | 5208 333 | 0x0 | 0x18 |
| FA | 8 | 5587 712 | 5681 818 | 0x0 | 0x16 |
| FA# | 8 | 5919 872 | 5952 381 | 0x0 | 0x15 |
| SOL | 8 | 6271 872 | 6250 000 | 0x0 | 0x14 |
| SOL# | 8 | 6644 864 | 6578 947 | 0x0 | 0x13 |
| LA | 8 | 7040 000 | 6944 444 | 0x0 | 0x12 |
| LA# | 8 | 7458 560 | 7352 941 | 0x0 | 0x11 |
| SI | 8 | 7902 080 | 7812 500 | 0x0 | 0x10 |

## 5.2. Fluctuations de tons

L'une des plus intéressantes possibilités offertes par la création musicale sur micro-ordinateur est la faculté de pouvoir faire varier le ton après avoir composé une mélodie. Les modifications les plus élémentaires sont les suivantes:

### 5.2.1. CHANGEMENT D'OCTAVE

Si un intervalle constant d'une octave est ajouté à la note enregistrée avant son envoi dans un des registres du PSG, on peut obtenir un changement de portée simple et rapide. La programmation de cet effet pourra s'effectuer par un décalage d'un cran à gauche des valeurs à placer dans les registres pour des notes d'octaves inférieures et par un décalage d'un cran à droite pour des notes d'octaves supérieures. Ainsi il est par exemple possible qu'une mesure écrite pour un piano résonne comme si elle avait été écrite pour des cloches si un changement d'octave supérieure est effectué.

### 5.2.2. TOUCHER

La virtuosité d'un musicien se juge essentiellement à son "toucher", à des décalages légers dans la portée. L'équivalent logique du toucher consiste à décaler vers le haut ou vers le bas chaque note d'un nombre pré-déterminé de notes. Ainsi un morceau écrit en DO et joué en DO dièse aura tous les DO convertis en DO dièse, tous les DO dièse convertis en RE, etc. (Notez qu'il faut traiter le cas particulier du SI d'un octave transformé en DO de l'octave supérieure). Toutes ces opérations supposent qu'un des douze demi-tons composant la gamme soit conservé dans la représentation enregistrée.

### 5.2.3. TEMPO

La durée de toute note enregistrée peut être exprimée en termes d'impulsions de métronome. Lorsqu'un morceau est joué, le tempo peut être modifié par une augmentation ou une réduction du temps affecté à chaque note, sans que soit pour cela modifiées les relations entre les notes. Ceci s'effectue en fixant le nombre de boucles durant lesquelles la note est maintenue.





## 5.3. Fluctuations de sons

### 5.3.1. VOLUME RELATIF D'UN CANAL

L'amplitude de chaque canal étant programmable séparément, vous disposez de 16 niveaux d'amplitude si vous avez choisi le mode d'amplitude variable (bit 4 des registres 10,11 ou 12 à 0). Dans le cas d'une note décroissante ou stable, lorsqu'une note est jouée ou 'lancée', une fréquence doit être placée dans les registres de période (réglage rudimentaire et fin), puis une valeur doit être placée dans le registre d'amplitude correspondant au canal choisi. Il est ainsi possible de produire des sons sur trois canaux disposant d'amplitudes variables et différentes.

### 5.3.2. DECLIN

La différence essentielle entre le son d'un piano et le son d'un orgue tient à la vitesse à laquelle le son perd de son intensité. Si toutes les notes peuvent décroître de la même façon, le générateur automatique d'enveloppe peut servir à reproduire cette onde décroissante. Chacun des trois canaux peut avoir la même constante d'affaiblissement mais des tempos différents afin de simuler le même instrument avec différentes variations de tempo.

### 5.3.3. AUTRES EFFETS

En ajoutant la génération de bruit à un ou plusieurs canaux, il est possible de produire des effets semblables au "souffle" d'un instrument à vent. Il est également possible de générer un rythme semblable à celui d'une batterie. Le fait que l'on puisse faire varier les fréquences du bruit permet de produire des effets semblables à ceux d'un synthétiseur avec une programmation très élémentaire.

D'autres effets sonores agréables comme le vibrato ou le tremolo, peuvent résulter d'une variation périodique du volume et de la fréquence. Etant donné qu'un microprocesseur intelligent contrôle ces effets, ceux-ci peuvent être associés au son lui-même ou à une autre source de synchronisation externe.

## 5.5. Applications

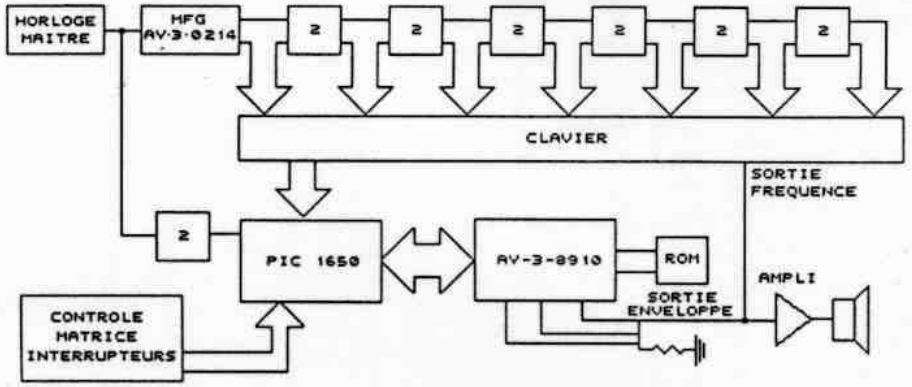
Même si le PSG trouve ses champs d'application musicaux essentiellement dans des domaines ludiques sinon frivoles, d'autres applications sont possibles, y compris dans les domaines pointus comme la reproduction musicale d'instruments sophistiqués ou la synthèse vocale. Par l'intermédiaire de fréquences sonores produites par une source externe afin que soit reproduit le plus fidèlement possible un orgue électronique, le PSG peut être utilisé comme un générateur d'enveloppe complexe. Le PSG a aussi la possibilité de générer des notes correspondant à des basses, une batterie ou des instruments à percussion, dans la mesure où il reproduit très fidèlement les sons à basse fréquence. Vous trouverez ci-dessous quelques exemples de ces possibilités.

### 5.5.1. GENERATION D'ENVELOPPE POUR SYNTHESE VOCALE

Le diagramme de génération d'enveloppe présenté ci-dessous montre comment un circuit AY-3-8910 peut être configuré pour produire des enveloppes de synthèse vocale. Toutes les fonctions sont contrôlées par le microprocesseur.

La base du système est constituée d'un générateur de fréquence associé à une série de diviseurs. Cela permet d'associer une fréquence à chaque touche du clavier. Le microprocesseur et le AY-3-8910 servent ici à remplacer les composants habituels de filtrage de voix, normalement indispensables dans ce type d'application électronique.

Le microprocesseur utilisé est un PIC1650(General Instrument) connecté en entrée avec le clavier par le biais d'une matrice d'interrupteurs. Au clavier sont entrées les informations relatives aux octaves, à l'amplitude et à la durée des sons. La matrice d'interrupteurs peut servir au contrôle du maintien du niveau sonore ou à divers effets spéciaux. La ROM figurant sur le schéma est optionnelle, elle peut servir à fournir des données externes utiles.







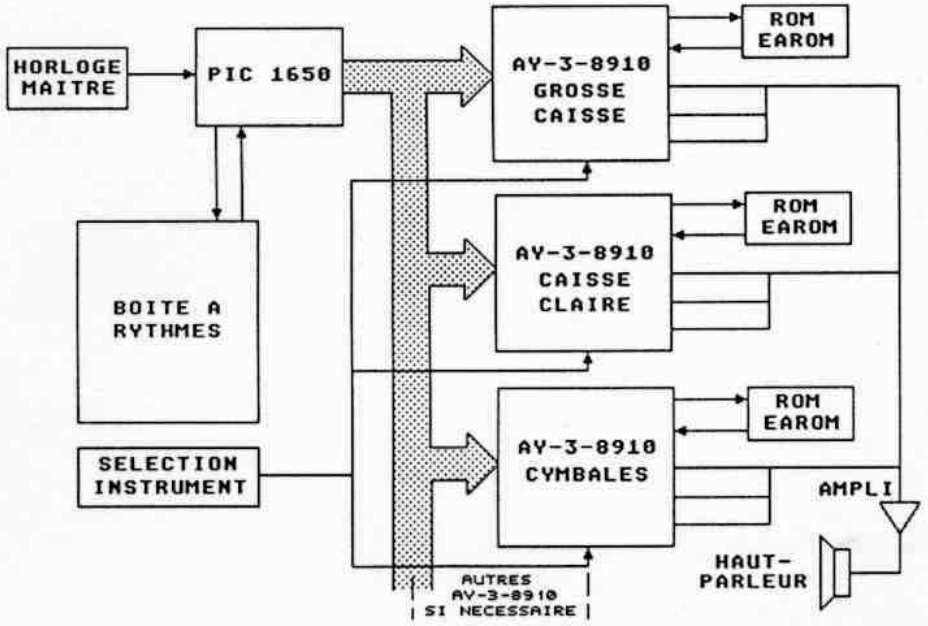

Ce système pourrait également être constitué de plusieurs AY-3-8910, tous contrôlés par un même microprocesseur. Cela fournit une solution peu onéreuse au développement de la synthèse vocale avec un minimum de composants.

### 5.5.2. PRODUCTION DE RYTHMES

Le schéma ci-dessous fournit un exemple simple de production de rythmes à l'aide de AY-3-8910. Le microprocesseur utilisé est un PIC 1650 qui peut être programmé pour piloter plusieurs AY-3-8910, tous étant connectés à un port d'entrée-sortie. Chaque AY-3-8910 fournit une enveloppe et une fréquence correspondant à l'instrument qu'il synthétise.

La 'boite à rythmes' sert à sélectionner un tempo précis et à l'envoyer au PIC. Les interrupteurs de sélection d'instrument permettent une sélection manuelle du ou des AY-3-8910 utilisé(s), à travers les lignes A8 et A9.

En option, des ROMs peuvent être couplées aux PSG, afin de produire des sons pré-enregistrés. Ces ROMs peuvent être remplacées par des EAROMs afin de fournir un rythme programmé à partir d'une boite à rythmes.
GENERATION DE RYTHME



## 6. PRODUCTION D'EFFETS SONORES

L'une des caractéristiques importantes du PSG est la possibilité qu'il offre de produire des effets sonores non musicaux pour accompagner une scène visuelle ou en tant que tels. Les sections suivantes traitent des techniques à utiliser pour réaliser ces effets et fournissent quelques exemples d'effets sonores classiques.

NDT: Tous les exemples fournis par le constructeur étaient basés sur un signal d'horloge à 1,78977Mhz. Le ST fournissant une base de temps de 2MHz au circuit son, les exemples ont été modifiés par les traducteurs pour correspondre à cette fréquence d'horloge.

### 6.1. Effets de tonalités seules

Beaucoup d'effets sonores sont possibles en utilisant uniquement la génération de sons sans ajouter de génération de bruit et sans utiliser la génération d'enveloppe. Parmi les possibilités offertes, signalons la sonnerie de téléphone (deux fréquences émises simultanément) ou les effets de sirénes d'usines comme dans l'exemple ci-dessous (deux fréquences distinctes émises successivement).

EFFET DE SIRENE D'USINE

| N° registre | Valeur à charger (hexadécimal) | Explication |
|---|---|---|
| tous registres | 00 | -- |
| R0 | 1C | Fréquence de 440Hz envoyée |
| R1 | 01 | sur le canal A. |
| R7 | 3E | Autorise le son sur A seulement. |
| R10 | 0F | Amplitude sonore maximale sur A. |
| | (Attendre environ | 350ms avant de continuer) |
| R0 | 9C | Fréquence de 187Hz environ. |
| R1 | 02 | envoyée sur le canal A. |
| | (Attendre environ | 350ms avant de continuer) |
| R10 | 00 | Désactive le canal A. |





### 6.2. Effets de bruit seul

Certains effets sonores ne nécessitent que l'utilisation du générateur de bruit et du générateur d'enveloppe, pour contrôler l'enveloppe du canal si d'autres canaux utilisent le générateur d'enveloppe.

Des exemples de tels effets sont fournis ci-dessous, ce sont une détonation et une explosion. Dans les deux cas un bruit pur est utilisé avec une enveloppe descendante. Entre les deux exemples cités, les seules différences tiennent à la longueur de la période de l'enveloppe. On notera qu'une explosion nettement plus sourde peut être obtenue en utilisant les trois canaux en même temps.

EFFET DE DETONATION

| N° Registre | Valeur à charger (hexadécimale) | Explication |
|---|---|---|
| tous registres | 00 | -- |
| R6 | 0F | valeur moyenne de période bruit |
| R7 | 07 | active le bruit sur canaux A,B,C |
| R10 | 0F | sélectionne l'amplitude maximale |
| R11 | 0F | sous le contrôle direct |
| R12 | 0F | du générateur d'enveloppe |
| R14 | 10 | période d'enveloppe à 0,5s |
| R15 | 00 | forme enveloppe descendante |

EFFET D'EXPLOSION

| N° Registre | Valeur à charger (hexadécimale) | Explication |
|---|---|---|
| tous registres | 00 | -- |
| R6 | 00 | Fixe période de bruit au maximum |
| R7 | 07 | active le bruit sur canaux A,B,C |
| R10 | 0F | sélectionne l'amplitude maximale |
| R11 | 0F | sous le contrôle direct du |
| R12 | 0F | générateur d'enveloppe |
| R14 | 32 | période d'enveloppe à 2s |
| R15 | 00 | forme enveloppe descendante |

## 6.3. Effets de balayage de fréquences

Les effets de laser, de bombe sifflante, de hurlement de loup et de voiture de course décrits ci-dessous utilisent tous des effets de balayage de fréquences. Dans tous les cas, ils sont constitués de boucles avec décrémentation ou incrémentation de valeurs de fréquences avec des valeurs de début, de fin et de pas variables entre les changements de fréquences. Ainsi l'effet de son laser est beaucoup plus rapide que le bruit d'accélération de la voiture de course, ces deux effets utilisant la même routine avec des paramètres différents.

D'autres effets sonores sont productibles aisément. Le balayage du registre de période du générateur de bruit(R6) produit un effet "doppler" qui parait tout à fait adapté à des jeux style "guerre des étoiles".

EFFET DE SON LASER

| N° Registre | Valeur à charger (hexadécimale) | Explication |
|---|---|---|
| tous registres | 00 | -- |
| R7 | 3E | active le bruit sur canaux A,B,C |
| R10 | 0F \| | sélectionne l'amplitude maximale |
| | | balayage de la fréquence du |
| R0 | 35 | canal A à travers une boucle du |
| R0 | 79 | processeur, de 35(2358Hz, 0,424 |
| | | msec.) à 79(1000Hz, 1,0ms). |
| R10 | 00 | désactive le canal A. |

EFFET DE BOMBE SIFFLANTE

| N° Registre | Valeur à charger (hexadécimale) | Explication |
|---|---|---|
| tous registres | 00 | -- |
| R7 | 3E | active le bruit sur canaux A,B,C |
| R10 | 0F \| | sélectionne l'amplitude maximale |
| | | balayage de la fréquence du |
| R0 | 35 | canal A à travers une boucle du |
| R0 | D5 | processeur, de 35(2358Hz, 0,424 |
| | | msec.) à D5(586Hz, 1,706ms). |
| R10 | 00 | désactive le canal A. |





## 6.4. Effets multi-canaux

Du fait de l'architecture particulière du PSG, beaucoup d'autres effets plus complexes sont possibles sans surcharge du processeur. Ainsi l'effet de hurlement de loup utilise deux canaux pour combiner en permanence le halètement sifflant avec les trois fréquences balayées du hurlement. Une fois le bruit défini sur le canal, le processeur ne s'occupe que de l'opération de balayage de fréquences.

EFFET DE HURLEMENT DE LOUP

| | | |
|---|---|---|
| tous registres | 00 | -- |
| R6 | 01 | période du bruit à son minimum. |
| R7 | 3E | active son sur A, bruit sur B. |
| R10 | 0F \| | sélectionne l'amplitude maximale |
| | | balayage de la fréquence du |
| R0 | 47 | canal A à travers une boucle du |
| R0 | 24 | processeur, de 47(1760Hz, 0,568 |
| | | msec.) à 24(3472Hz, 0,288ms). |
| | attente d'environ | 12ms entre changement fréquence |
| | attente de 150ms | environ en fin de boucle |
| R0 | 47 | canal A à travers une boucle du |
| R0 | 35 | processeur, de 47(1760Hz, 0,568 |
| | | msec.) à 35(2358Hz, 0,424ms). |
| | attente d'environ | 25ms entre changement fréquence |
| R0 | 35 | canal A à travers une boucle du |
| R0 | 74 | processeur, de 35(1760Hz, 0,424 |
| | | msec.) à 74(1073Hz, 0,932ms). |
| | attente d'environ | 6ms entre changement fréquence |
| R10 | 00 | désactive le canal A. |
| R11 | 00 | désactive le canal B |

EFFET DE VOITURE DE COURSE

| | | |
|---|---|---|
| tous registres | 00 | -- |
| R3 | 0F | amplitude canal B = 30ms (33Hz) |
| R7 | 3C | active son sur A et sur B. |
| R10 | 0F | sélectionne amplitude maxi A. |
| R11 | 0A | sélectionne amplitude moy. B. |
| | | balayage de la fréquence du |
| R1/R0 | C/40 | canal A à travers une boucle du |
| R1/R0 | 4/70 | processeur, de C40(39Hz;25,6ms) |
| | | à 470(100Hz, 10ms). |
| | attente d'environ | 3ms entre changement fréquence |
| R1/R0 | 9/F7 | canal A à travers une boucle du |
| R1/R0 | 3/53 | processeur, de 9F7(49Hz;20,4ms) |
| | | à 353(146,9Hz, 6,81ms). |
| | attente d'environ | 3ms entre changement fréquence |
| R1/R0 | 6/A7 | canal A à travers une boucle du |
| R1/R0 | 1/1C | processeur, de 6A7(73,4Hz, soit |
| | | 13,62ms) à 11C(436,1Hz;2,29 ms) |
| | attente d'environ | 6ms entre changement fréquence |
| R10 | 00 | désactive le canal A. |
| R11 | 00 | désactive le canal B |

## 7. CARACTERISTIQUES ELECTRIQUES

### 7.1. Intervalles d'utilisation

Température de stockage ..... -55°C à +150°C
Température de fonctionnement ..... 0°C à +40°C
Alimentation et tous voltages
relativement à la masse ..... -0,3V à +8,0V

Un dépassement de ces valeurs minima ou maxima peut endommager ce circuit. Le bon fonctionnement du circuit n'est garanti qu'aux conditions standard ci-dessous.

### 7.2. Conditions standard d'utilisation

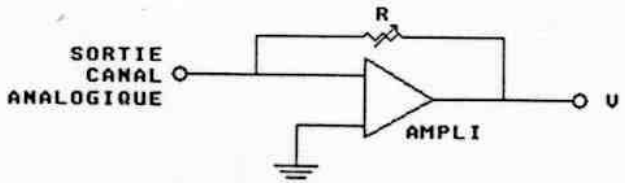
Alimentation ..... +5V + 5%
Tension de référence ..... MASSE
Température de fonctionnement .. 0°C à +40°C

### 7.3. Caractéristiques électriques statiques

| Caractéristique | Sym | Min | Typ* | Max | Unité | Conditions |
|---|---|---|---|---|---|---|
| Toutes entrées | | | | | | |
| Logique "0" | VIL | 0 | - | 0,6 | V | |
| Logique "1" | VIH | 2,4 | - | VCC | V | |
| Toutes sorties (sauf canaux analogiques) | | | | | | |
| Logique "0" | VOL | 0 | - | 0,5 | V | IOL=1,6mA,20pF |
| Logique "1" | VOH | 2,4 | - | VCC | V | IOH=100 A,20pF |
| Sorties canaux analog. | VO | 0 | - | 60 | dB | Test circuit** |
| Alimentation | ICC | - | 45 | 75 | mA | |

* Les valeurs typiques sont données pour 25°C.
** voir schéma ci-dessous.

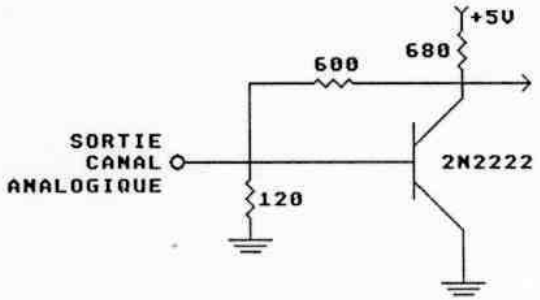
CIRCUIT DE TEST DE SORTIE ANALOGIQUE








CONVERTISSEUR EN VOLTAGE

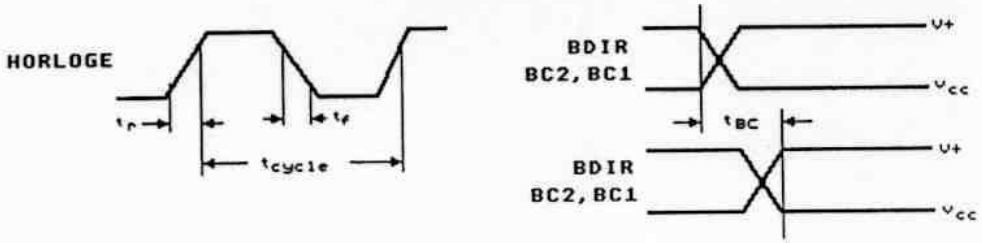



### 7.4. Caractéristiques électriques dynamiques

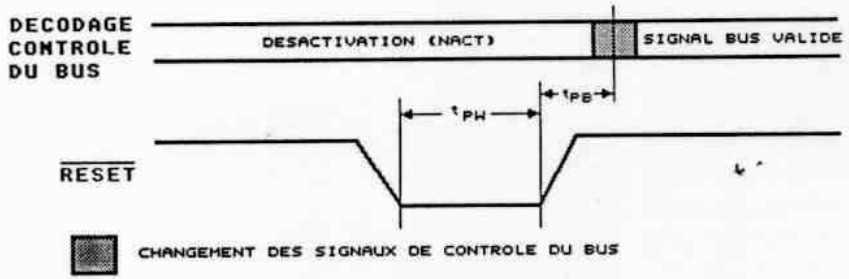
| Caractéristique | Sym | Min | Typ* | Max | Unité | Schéma |
|---|---|---|---|---|---|---|
| Entrée Horloge | | | | | | SCHEMA 1 |
| Fréquence | | 1,0 | - | 2,0 | MHz | |
| Temps de montée | tr | - | - | 50 | ns | |
| Temps de descente | tf | - | - | 50 | ns | |
| Largeur impulsion | tCYCLE | 25 | 50 | 75 | % | |
| Signaux Bus BDIR,BC1,BC2 | | | | | | SCHEMA 1 |
| Temps de retard | tBC | - | - | 50 | ns | |
| RESET | | | | | | SCHEMA 2 |
| Largeur impulsion | tPH | 500 | - | - | ns | |
| Délai controle bus | tPB | 100 | - | - | ns | |
| A9,A8,DA7-DA0 (Adresse) | | | | | | SCHEMA 3 |
| Temps pré-établissement | tAS | 400 | - | - | ns | |
| Temps de maintien | tAH | 100 | - | - | ns | |
| DA7-DA0 (Mode Ecriture) | | | | | | SCHEMA 4 |
| Largeur d'impulsion | tDW | 500 | - | 10000 | ns | |
| Temps pré-établissement | tDS | 50 | - | - | ns | |
| Temps de maintien | tDH | 100 | - | - | ns | |
| DA7-DA0 (Mode Lecture) | | | | | | SCHEMA 5 |
| Temps accès en lecture | tDA | - | 250 | - | ns | |
| DA7-DA0 (Mode désactivé) | | | | | | |
| Délai trois-états | tTS | - | 100 | - | ns | |

* Les valeurs typiques sont données pour 25°C.

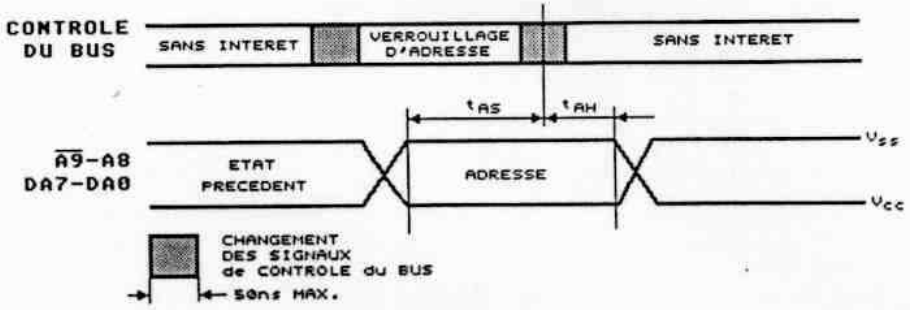
## 1. CHRONOGRAMMES HORLOGE ET SIGNAUX DU BUS




## 2. CHRONOGRAMME DU RESET



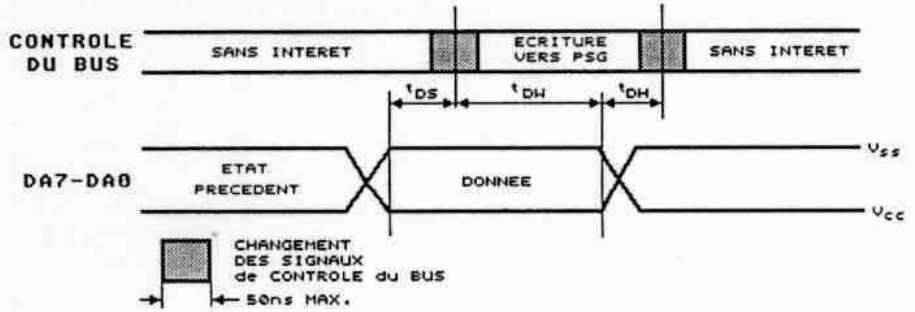

## 3. CHRONOGRAMME DU VERROUILLAGE D'ADRESSE



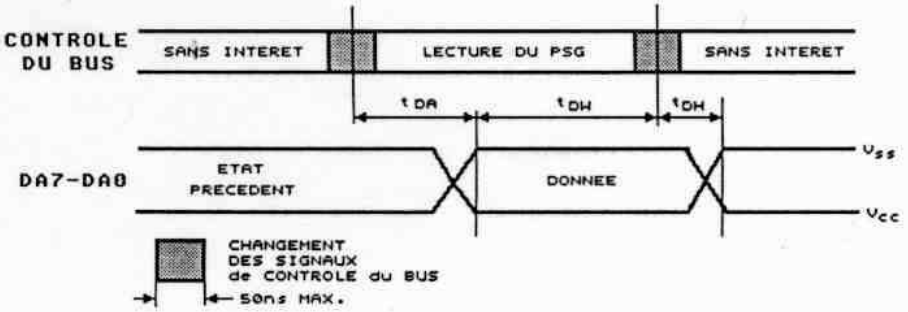





## 4. CHRONOGRAMME D'ECRITURE DE DONNEE




## 5. CHRONOGRAMME DE LECTURE DE DONNEE



# LE WD1770/1772

# LE WD1770/1772

## 1. CARACTERISTIQUES

* 28 broches
* Alimentation unique 5 Volts
* Séparateur de données intégré
* Précompensation en écriture intégrée
* Ecriture en simple et double densité
* Contrôle du moteur
* Longueur de secteur de 128, 256, 512, OU 1024 octets
* Compatibilité TTL
* Bus de données bidirectionnel 8 bits
* Deux versions disponibles:

  WD1770 standard
  WD1772 'taux de pas' plus rapide

## 2. DESCRIPTION

Le WD1770 est un circuit réalisé en technologie MOS/LSI qui permet les opérations de contrôle et de formatage d'une disquette. Identique à son prédécesseur, le WD179X, il possède en plus un circuit de séparateur de données et de précompensation en écriture. L'interface avec le lecteur n'a pas besoin de circuits logiques additionnels, excepté pour les tampons récepteurs nécessaires aux opérations. Le périphérique contient un signal programmable d' activation du moteur (Motor On), lui permettant le démarrage du moteur avant toute opération sur la disquette.

Le WD1770 est la version bas prix du Formateur/Contrôleur de disquette FD179X, compatible avec le 179X. Une simple ligne de lecture (RD, broche 19) est la seule entrée nécessaire à la réception série FM (simple densité) ou MFM (double densité) d'une donnée à partir du lecteur du disquette. Ce périphérique a été spécialement conçu pour contrôler des lecteurs de disquettes avec un flux de données de 125 Kbits/Sec en simple densité et de 250 Kbits/Sec en double densité. De plus, une précompensation en écriture de 125 ns par rapport à la valeur théorique est possible par commande logicielle.

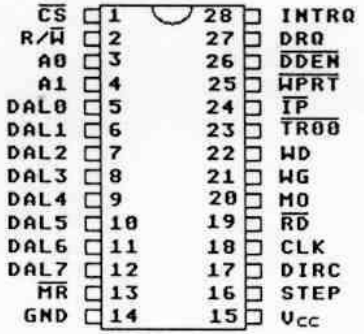
Deux versions du WD1770 sont disponibles. La version standard est compatible avec le taux de transfert de données du 179X alors que le WD1772 offre des 'taux de pas' (délai de déplacement de tête de lecture) de 2, 3, 5 et 6 ms.
N.D.T: Le ST est équipé d'un WD1772 fonctionnant en écriture double densité.





L'interface processeur se compose d'un bus 8 bits, bidirectionnel, pour le transfert des données, d'états et de commandes. Toutes les communications avec le lecteur passent par ces lignes de données.

SCHEMA DU BROCHAGE
(Vue du dessus)




## DESCRIPTION DES BROCHES

1 CS — Sélection de Circuit
Un signal bas sur cette broche sélectionne le circuit et valide la communication entre le WD1772 et le processeur maître.

2 R/W — Lecture/Ecriture
Contrôle du sens des données. Un signal haut sur cette broche contrôle le placement de données sur les lignes D0-D7 à partir du registre sélectionné, alors qu'un signal logique bas provoque l'écriture dans le registre sélectionné.

3.4 A0-A1 — Adresses 0-1
Ces deux adresses servent à sélectionner un registre du WD1772.

| CS | A1 | A0 | R/W=1 (Lecture) | R/W=0 (Ecriture) |
|---|---|---|---|---|
| 0 | 0 | 0 | registre d'état | registre de commande |
| 0 | 0 | 1 | registre de piste | registre de piste |
| 0 | 1 | 0 | registre de secteur | registre de secteur |
| 0 | 1 | 1 | registre de données | registre de données |

5-12 DAL0-DAL7 — Lignes de Données 0 à 7
Bus bidirectionnel 8 bits utilisé pour les transferts de données. Ce bus est validé par CS et R/W.

13 MR — Reset Maitre
Un impulsion basse sur cette ligne remet dans un état défini le WD1772 et réinitialise les registres.

14 GND — Masse

15 Vcc — Alimentation +5 V (±5%)

16 STEP — Pas
Cette broche de sortie envoie une impulsion pour pour déplacer d'un 'pas' de la tête de lecture sur la disquette. Différents pas sont possibles.

17 DIRC — Direction
Direction de déplacement de la tête. Le signal est haut lorsque la tête doit aller vers le centre, et bas lorsqu'elle doit se déplacer vers l'extérieur.

18 CLK — Horloge
Impulsion du signal d'horloge, servant comme base de temps interne, sur une fréquence de 8 MHz.

19 RD — Lecture de Données
Cette entrée, active à l'état bas, reçoit les impulsions de données et de fréquence à partir de la tête de lecture.

20 MO — Démarrage du Moteur
Actif à l'état haut, MO sert à lancer le moteur avant une opération de lecture, d'écriture ou de positionnement.

21 WG — Ouverture de Porte
Validé, à l'état haut, avant une opération d'écriture. Inhibé le reste du temps (lecture, déplacement de la tête), empèche l'écriture de parasites.

22 WD — Ecriture de Données
Lignes où sont placées les impulsions de données et de fréquences à écrire sur la disquette.

23 TR00 — Piste 00
Cette entrée active à l'état bas indique au WD1772 que le tête de lecture/écriture est positionnée en piste 0.

24 IP — Index d'Impulsions
Cette entrée active à l'état bas est validée à chaque tour complet de la disquette.





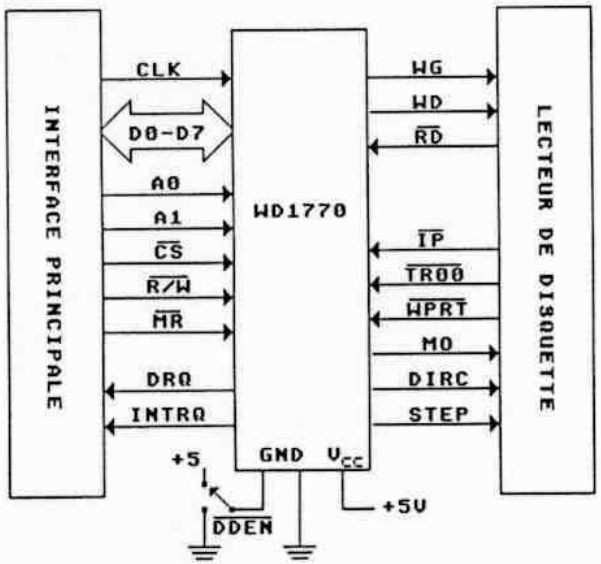
25 WPRT — Protégé en Ecriture
Cette entrée est testée à chaque fois qu'une commande d'écriture est reçue. Un état bas signifie que la disquette est protégée en écriture. Si c'est le cas la commande est interrompue.

26 DDEN — Validation de Double Densité
Cette broche spécifie si l'on est en mode double densité (état bas) ou simple densité (état haut).

27 DRQ — Demande de Données
Cette sortie, active à l'état haut, est validée lorsque le Registre de Données est plein en lecture ou lorsqu'il est vide en écriture.

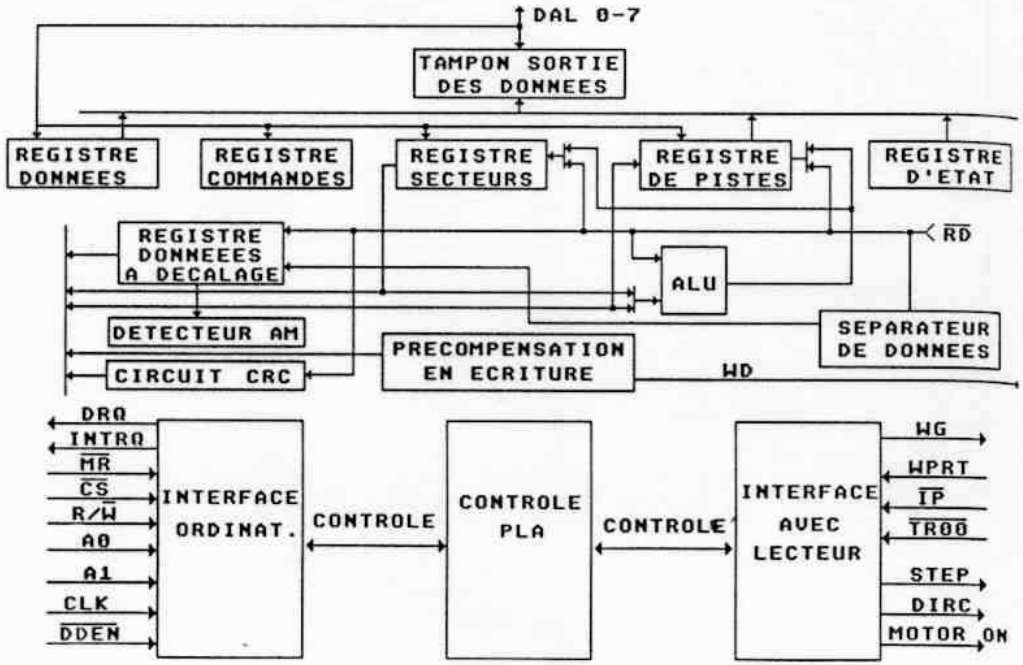
28 INTRQ — Demande d'Interruption
Sortie active à l'état haut, positionnée à la fin de l'exécution de chaque commande, et remise à zéro par la lecture du Registre d'état.

DIAGRAMME DU BLOC SYSTEME DU WD1770



## 3. ARCHITECTURE

DIAGRAMME GENERAL DU WD1770




Registre de Données à Décalage 'DSR'

Registre de 8 bits qui sert à regrouper les données série provenant de l'entrée RD (Lecture de donnée, broche 19) lors d'une opération de lecture, et à envoyer les données série sur la sortie WD (Ecriture de données, broche 22) lors d'une opération d'écriture.

**Registre de données 'DR'**

Ce registre sert, lors d'une opération d'écriture ou de lecture, de mémoire tampon. En lecture, l'octet assemblé dans le Registre de Données à Décalage est transféré en parallèle dans le Registre de Données et inversement en écriture.

Lors d'une recherche de piste, le Registre de Données contient l'adresse de la piste désirée.





**Registre de Piste (TR)**

Registre 8 bits qui contient le numéro de la piste sur laquelle se trouve la tête de lecture/écriture. Il est incrémenté à chaque fois que la tête avance d'un pas vers le centre, et décrémenté à chaque déplacement d'un pas vers la piste 00. Durant les opérations de lecture ou d'écriture le contenu du Registre de Piste est comparé avec le numéro de piste enregistré dans le Champs d'Identification (ID) des secteurs.

Ce registre peut être lu ou écrit, mais l'écriture ne doit pas être faite durant une opération.

**Registre de Secteur (SR)**

Ce registre 8 bits contient le numéro du secteur désiré. Le contenu du registre est également comparé avec le numéro de secteur enregistré dans le Champ d'Identification (ID), lors d'une opération de lecture ou d'écriture.

Il peut être lu ou écrit, mais l'écriture ne doit pas se faire durant une opération.

**Registre de Commande (CR)**

Ce registre 8 bits contient la commande en cours d'exécution. Il n'est pas possible de le lire et il ne doit pas être écrit durant une opération, sauf pour l'envoi d'une commande d'Interruption Forcée.

**Registre d'état (STR)**

Registre 8 bits, ne pouvant qu'être lu, il contient des renseignements sur l'état du contrôleur. La signification des bits de ce registre dépend du type de commande exécutée:

bit 7
Moteur en Marche
Reflète l'état de la broche numéro 20.

bit 6
Protégé en Ecriture
Uniquement utilisé en écriture. Ce bit indique qu'une disquette est protégée en écriture. Il est remis à 0 lors d'une mise à jour.

bit 5
Type de Marque/Lancement du Moteur
Lors d'une commande de type 1 (positionnement de la tête de lecture), ce bit indique que le moteur a atteint sa vitesse optimale de rotation.
Lors d'une commande de type 2 (lecture/écriture de secteur), il indique le type de Marque de Donnée:
0 = Marque normale,
1 = Marque effaçée.

bit 4
Enregistrement Introuvable
Indique que la face, la piste, ou le secteur recherché n'a pu être trouvé.

bit 3
Erreur CRC
Le contenu du mot de CRC (checksum) sur la disquette est différent de celui réalisé lors de la lecture.

bit 2
Perte de données/Piste 00
Indique qu'aucune réponse n'a été faite à une Demande de Donnée (DRQ).

Lors d'une commande de type 1, ce bit reflète l'état de la broche 00. Il est mis à 1 lorsque la tête de lecture se trouve sur la piste 00.

bit 1
Demande de Donnée/Index
Copie de la broche DRQ. Lorsqu'il est à 1, ce bit indique que le Registre de Donnée est plein, lors d'une opération de lecture, et vide, lors d'une opération d'écriture.
Lors d'une commande de type 1, reflète l'état de la broche d'index (PI). Il est mis à un lors de la réception d'une impulsion d'index.

bit 0
Travail (Busy)
Indique lorsqu'il est à 1, qu'une commande est en train d'être exécutée.

**Circuit Logique CRC**

Ce circuit génère et contrôle le mot de 16 bits appelé CRC (Cyclic Redundancy Check) qui permet de tester la validité des données lues à partir d'un test de somme.
Le polynôme est :

$$G(x) = x16 + x12 + x5 + 1.$$

Le CRC est réalisé sur toutes les informations depuis la Marque d'Adresse (AM) jusqu'au mot de CRC (non compris). Le registre CRC est rempli avant que les données soient envoyées.

**Unité Arithmétique et logique (ALU)**

L'unité arithmétique et logique réalise des comparaisons, des incrémentations et des décrémentations. Il sert aux modifications des registres et aux comparaisons de ceux-ci avec le champ d'ID enregistré sur la disquette.





**Base de temps et Contrôle**

Tous les contrôles d'interface avec le processeur maître ou avec le lecteur passent par ce circuit logique. La base de temps interne est générée à partir d'une horloge externe (à 8 MHz). Le WD1770 a deux modes de travail, selon l'état de DDEN. Lorsque DDEN est à 0, la double densité est validée, lorque DDEN est à 1, la simple densité est alors validée. Seule la double densité est traitée par le ST.

**Détecteur de Marques d'Adresse (AM)**

Le détecteur de Marques d'Adresse permet de détecter et de différencier les Marques d'Adresse (AM) qui sont écrites avant le Champ d'Identification et avant le champ des données. Les Marques d'Adresse sont de octets spéciaux, supérieurs à $F7 et qui ont été écrits sans impulsions de fréquence, contrairement aux autres données se trouvant sur la disquette. Elles servent à délimiter les différent champs se trouvant sur la disquette.

**Séparateur de Données**

Le Séparateur de Données permet de séparer les données des impulsions de fréquence.

## 4. INTERFACAGE AVEC LE SYSTEME

L'interfaçage avec le système est réalisé par les 8 lignes de données (DAL) et les signaux de contrôle associés. Les lignes DAL servent au transfert des données, des bits d'état et des caractères de contrôle depuis ou vers le WD1772. Les lignes de données sont utilisées en sortie lorsque CS est actif et R/W à 1. Elles servent en entrée lorsque CS est actif et que R/W est à 0.

Lorsque le processeur maître fait une demande de transfert de données avec le contrôleur du lecteur de disquette, l'adresse est décodée et CS est mis à l'état bas. Les bits d'adresse A0 et A1, combinés avec le signal R/W servent à définir les registres suivants:

| A1 | A0 | (Lecture) R/W=1 | (Ecriture) R/W=0 |
|---|---|---|---|
| 0 | 0 | registre d'état | registre de commande |
| 0 | 1 | registre de piste | registre de piste |
| 1 | 0 | registre de secteur | registre de secteur |
| 1 | 1 | registre de données | registre de données |

La sortie Demande de Données (DRQ) est employée pour contrôler les transferts de données en mode DMA entre le WD1772 et le processeur maître. Le bit 1 du Registre d'état est le reflét de cette broche durant les opérations de lecture et d'écriture.

Durant une opération de lecture de disquette, DRQ est activé (état haut) lorsque l'octet qui a été reconstitué dans le Registre de Données à Décalage est transféré en parallèle dans le Registre de données.

DRQ est mis à 0 lors de la lecture du Registre de Données par le processeur. Si de nouvelles données sont chargées dans le Registre de Données avant que celles qui s'y trouvaient aient été transférées, les données précédentes sont perdues et le bit de Perte de Données est positionné dans le Registre d'Etat. La lecture se poursuit, cependant, jusqu'à la fin du secteur.

En écriture, DRQ est activé lorsque le Registre de Données passe son contenu au Registre de Données à Décalage et demande ainsi un nouvel octet de donnée. DRQ est remis à 0 lorsque le processeur écrit la nouvelle donnée dans le Registre de Données. Si la donnée n'est pas chargée dans les temps, l'octet suivant est réclamé par le lecteur et un octet à zéro est écrit sur la disquette. Le bit Perte de Données est alors mis à 1 dans le Registre d'Etat.

A la fin de toute commande INTRQ est positionné à 1, provoquant ainsi une demande d'interruption. INTRQ est remis à 0 soit par la lecture du Registre d'Etat soit par l'écriture d'une nouvelle commande. De plus INTRQ est positionné lorsque la commande 'Interruption Forcée' est rencontrée.

## 5. OPERATIONS GENERALES DE LECTURE

La taille d'un secteur peut être de 128, 256, 512 ou 1024 octets, en simple ou en double densité. En simple densité DDEN doit être à 1. Il doit être à 0 pour la double densité. La taille des secteurs est déterminée au formatage par le quatrième octet du Champ d'Identification (ID):

| Champ ID de longueur | nombre d'octets du secteur |
|---|---|
| $00 | 128 |
| $01 | 256 |
| $02 | 512 |
| $03 | 1024 |

Le WD1772 peut traiter les secteurs allant de 1 à 255 et les pistes de 0 à 255.





## 6. OPERATIONS GENERALES D'ECRITURE

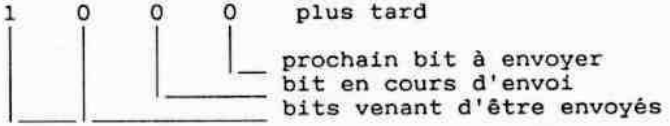
Lors d'une opération d'écriture sur une disquette, la sortie 'Porte d'Ecriture' (WG) est activée, permettant ainsi le passage de données dans la tête de lecture/écriture. Afin de prévenir les erreurs d'écriture dues à des parasites, le premier octet de donnée doit être écrit dans le Registre de Données en réponse à une Demande de Données avant l'activation de la Porte d'Ecriture.

L'écriture est empêchée lorsque l'entrée 'Protégé en Ecriture' (WPRT) est à l'état bas. La commande d'écriture prend fin sur le champ, une interruption est générée et le bit 'Protégé en écriture' est mis à 1 dans le Registre d'état.

L'opération d'écriture consiste pour le WD1772 à s'assurer que WPRT n'est pas à l'état bas puis à envoyer une série d'impulsions comprenant des données et des impulsions de fréquence.

Le bit de précompensation en écriture permet, lorsqu'il est validé, une précompensation d'écriture automatique: Le flot de données à écrire est envoyé 125ns plus tôt ou plus tard (par rapport à la valeur normale) selon la table suivante:

| | | | | |
|---|---|---|---|---|
| x | 1 | 1 | 0 | plus tôt |
| x | 0 | 1 | 1 | plus tard |
| 0 | 0 | 0 | 1 | plus tôt |
| 1 | 0 | 0 | 0 | plus tard |




La précompensation en écriture est activée lorsque la densité d'écriture est forte permettant ainsi de limiter les erreurs d'écriture. Généralement elle est activée pour l'écriture des pistes se trouvant sur la moitié la plus prés du centre d'une disquette.

## 7. DESCRIPTION DES COMMANDES

Le WD1772 accepte onze commandes. Celles-ci ne doivent être placée dans le registre de commande que lorsque le bit de Travail (Busy, bit 0 du Registre d'Etat) est à 0. La seule exception concerne la commande 'Interruption Forcée'. Lorsqu'une commande va être exécutée, le bit de Travail est mis à 1. A la fin de l'exécution de la commande une interruption est générée et le bit de Travail est remis à 0. Le Registre d'Etat permet de savoir si une erreur s'est produite durant l'exécution de la commande. Pour plus de clarté, les commandes sont réparties en quatre groupes:

Le type 1 concerne les commandes de déplacement de tête,
le type 2 comprend les commandes de lecture et d'écriture de secteurs,
le type 3 englobe les commande de lecture brute de piste et d'écriture brute de piste (formatage).
Le type 4 ne concerne que la commande d'interruption forcée.

| TYPE | COMMANDE | BIT 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Restore | 0 | 0 | 0 | 0 | h | V | r1 | r1 |
| 1 | Seek | 0 | 0 | 0 | 1 | h | V | r1 | r0 |
| 1 | Step | 0 | 0 | 1 | u | h | V | r1 | r0 |
| 1 | Step-in | 0 | 1 | 0 | u | h | V | r1 | r0 |
| 1 | Step-out | 0 | 1 | 1 | u | h | V | r1 | r0 |
| 2 | Read sector | 1 | 0 | 0 | m | h | E | 0 | 0 |
| 2 | Write sector | 1 | 0 | 1 | m | h | E | P | a0 |
| 3 | Read address | 1 | 1 | 0 | 0 | h | E | 0 | 0 |
| 3 | Read track | 1 | 1 | 1 | 0 | h | E | 0 | 0 |
| 3 | Write track | 1 | 1 | 1 | 1 | h | E | P | 0 |
| 4 | Force Interrupt | 1 | 1 | 0 | 1 | I3 | I2 | I1 | I0 |

Drapeaux des commandes de type 1

h drapeau Motor On (bit 3)
- h = 0 activation du moteur
- h = 1 désactivation du moteur

V drapeau de vérification (bit 2)
- V = 0 pas de vérification
- V = 1 vérification de piste

r1,r0 taux de pas (bits 1,0)

| r1 | r0 | WD1770 | WD1772 |
|---|---|---|---|
| 0 | 0 | 6 ms | 2 ms |
| 0 | 1 | 12 ms | 3 ms |
| 1 | 0 | 20 ms | 5 ms |
| 1 | 1 | 30 ms | 6 ms |

u drapeau de mise à jour (bit 4)
- u = 0 pas de mise à jour
- u = 1 mise à jour du registre de piste

Drapeaux de commande de type 2 et 3

m drapeau de secteurs multiples (bit 4)
- m = 0 opération sur un seul secteur
- m = 1 opération sur plusieurs secteurs





a0 donnée marque d'adresse (bit 0)
- a0 = 0 écriture d'une marque d'adresse normale
- a1 = 1 écriture d'une marque d'adresse effacée

E délai d'attente (bit 2)
- E = 0 pas de délai
- E = 1 ajoute un délai de 30 ms

P drapeau d'écriture précompensée (bit 1)
- P = 0 valide la précompensation
- P = 1 inhibe la précompensation

Drapeaux de commandes de type 4

I3-I0 conditions d'interruption (bits 3-0)
- I0 = 1 sans importance
- I1 = 1 sans importance
- I2 = 1 interruption sur une impulsion d'index
- I3 = 1 interruption immédiate
- I3-I0 = 0 fin sans interruption

## 7.1. Commandes de type 1

Les commandes de type 1 comprennent les commandes 'Recherche de piste 00' (Restore), 'Positionnement sur une piste' (Seek), 'Déplacement de la tête de lecture' (Step), 'Déplacement de la tête vers le centre' (Step in), et 'Déplacement de la tête vers l'extérieur' (Step out). Chaque commande est fonction du 'taux de pas' du moteur (r0,r1).

Une impulsion de 4 s (double densité) ou de 8 s (simple densité) est générée sur la sortie vers le lecteur. A chaque impulsion de pas générée, la tête avance d'un pas selon la direction déterminée par le Signal de Direction (DIRC). Celle-ci reste la même tant qu'une commande ne spécifie pas un changement de direction.

Le Signal de Direction est mis, à l'état haut pour un déplacement de la tête vers le centre, à l'état bas pour un déplacement vers l'extérieur. Il est validé 24 s avant la première impulsion.

Après l'exécution d'un pas dans une direction, une période de 30 ms est nécessaire au positionnement de la tête de lecture, si le drapeau de vérification est positionné, dans une commande de type 1. Cette temporisation de 30 ms est effectuée également pour une commande de type 2 ou 3, si le drapeau 'E' est positionné.

Lors d'une opération de 'positionnement de piste', de 'recherche de piste 00' ou de 'déplacement de tête' une vérification de la tête de lecture/écriture peut être faite en positionnant le bit 2 à 1 (V = 1). La vérification commence après les 30 ms nécessaires au positionnement de la tête sur la disquette. Le numéro de la piste lu sur le premier champ d'ID rencontré est comparé avec le contenu du Registre de Piste. Si les nombres sont identiques et si le CRC de l'ID est correct, la phase de vérification est achevée et une interruption sans erreur est générée via INTRQ. Si la comparaison est correcte mais que le CRC n'est pas valide, le bit 'Erreur CRC'(bit 3) du Registre d'Etat est mis à 1, et le prochain champ d'ID rencontré est lu dans le but de vérification.
Si le WD1772 n'a pas trouvé un champ d'ID avec un numéro de piste et un CRC valides en 5 rotations de disquette, une Erreur de Positionnement est placée dans le Registre d'Etat et une interruption INTRQ est générée.
Si V = 0, aucune vérification n'est faite.

Toutes les commandes exceptée 'Interruption Forcée' doivent tenir compte du drapeau de moteur. Si le moteur est arrêté lors de la réception d'une commande, le drapeau Motor On sera mis à 1 et la commande ne sera exécutée qu'après 6 rotations afin que la vitesse optimale de 300 TPM soit atteinte, ce qui représente approximativement une seconde. Après l'exécution de la commande, les moteurs restent en marche (MO = 1) pendant 10 tours, soit 2 secondes. Si une commande est reçue pendant ce laps de temps, celle-ci sera exécutée instantanément, sans effectuer les six rotations de démarrage. Cette configuration permet la succession d'opérations de lecture ou d'écriture sans avoir à redémarrer le moteur à chaque fois, le WD1772 supposant que la vitesse optimale est acquise.

Recherche de la piste 00 (Restore)

A la réception de cette commande, la broche TR00 est testée. Si elle se trouve à l'état bas, la tête de lecture/écriture se trouve déjà sur la piste 00. Le Registre de Piste est alors chargé avec des zéros et une interruption est générée. Si TR00 est à l'état haut, indiquant que la tête ne se trouve pas sur la piste 00, une impulsion, fonction du taux de pas r1-r0, est envoyée sur la broche de pas (broche 16) tant que TR00 n'est pas à l'état bas. Lorsque la piste est trouvée, le Registre de Piste est mis à 0 et une interruption est générée. Si TR00 n'est pas passé à l'état bas après 255 impulsions, la commande est terminée, une interruption est générée et le bit d'Erreur de Positionnement est mis à 1 dans le Registre d'Etat. Une opération de vérification est effectuée si le drapeau 'V' est positionné. Le bit 'h' permet de faire démarrer le moteur au début de la commande.





Positionnement sur une piste (Seek)

Cette fonction permet de positionner la tête de lecture sur une piste donnée. Son bon fonctionnement suppose que le Registre de Piste est correctement chargé avec le numéro de la piste courante sur laquelle se trouve la tête de lecture. Le numéro de piste sur laquelle le positionnement est désiré est mis dans le Registre de Données. Le Registre de Piste est mis à jour au fur et à mesure que des impulsions de déplacement de tête sont envoyées, ceci tant que le Registre de Piste est différent du Registre de Données. Une vérification est également faite si le drapeau 'V' est à 1. Le bit 'h' permet le démarrage du moteur au début de l'exécution de la commande et une interruption est générée à l'accomplissement de celle-ci.

Si l'on utilise plusieurs lecteurs, le Registre de Piste doit être mis à jour pour le lecteur concerné avant l'opération de positionnement.

Déplacement de la tête de lecture d'une piste (Step)

A la réception de cette commande, une impulsion de pas est envoyée au lecteur. La direction du pas n'étant pas modifiée, est celle qui a servie dans une précédente opération de déplacement de tête. Une vérification est effectuée après un délai fonction de r1-r0 si le drapeau 'V' est positionné. Si le drapeau 'u' est à 1, le registre de piste est mis à jour. Le bit 'h' permet le démarrage du moteur avant le lancement de la commande et une interruption est générée à l'accomplissement de celle-ci.

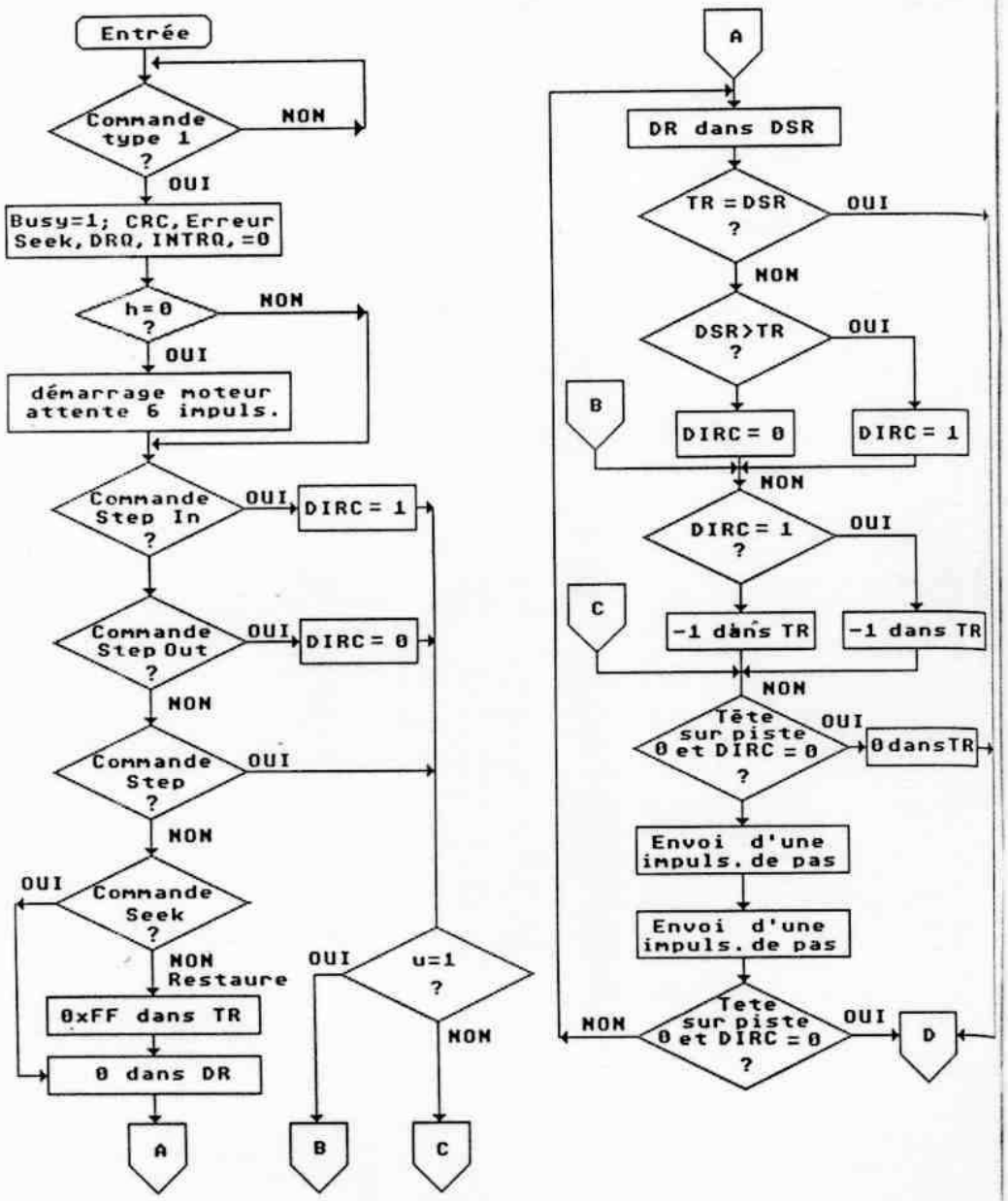
Déplacement de la tête d'une piste vers le centre (Step in)

La tête de lecture est déplacée d'une piste vers le centre de la disquette à la réception de cette commande, quelque soit l'état dans lequel se trouvait la broche DIRC. Celle-ci est alors mise à l'état haut. Si le drapeau 'u' est à 1, le Registre de Piste est incrémenté. Une vérification est faite si le drapeau 'V' est positionné dans un délai fonction de r1-r2. Le bit 'h' permet la mise en route du moteur avant le début d'exécution de la commande. Une interruption est générée à l'achèvement de cette fonction.

Déplacement de la tête d'une piste vers l'extérieur (Step out)

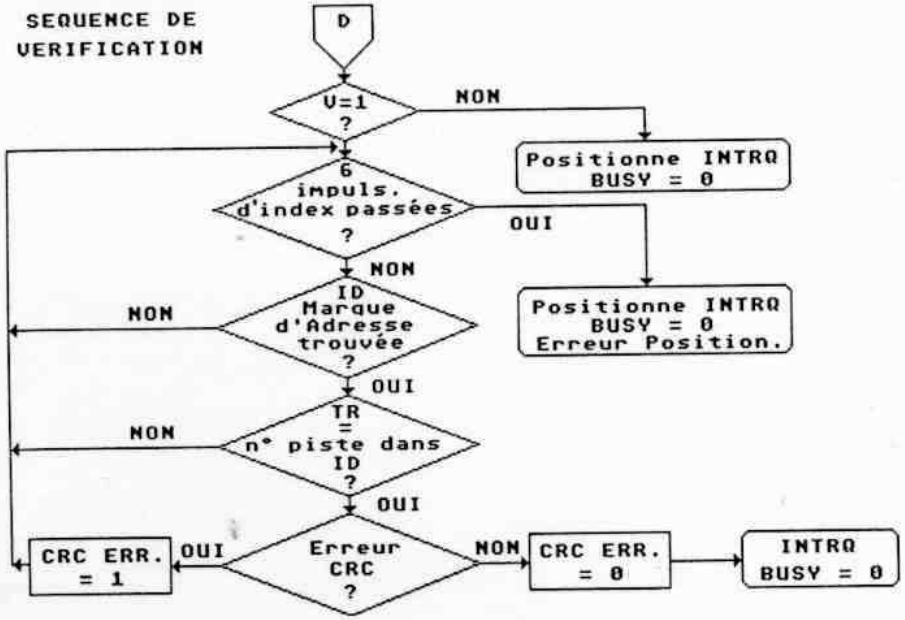
Commande au fonctionnement indentique à la fonction précédente (Step in), excepté que le déplacement se fait vers l'extérieur de la disquette.

L'organigramme des fonctions du type 1 est présenté ci-après.








## 7.2. Commandes du type 2

Les commandes de type 2 correspondent aux commandes de lecture et d'écriture de secteur. Avant tout chargement de commande de type 2 dans le Registre de Commandes, le numéro du secteur désiré doit être chargé dans le Registre de Secteur.

A la réception de la commande le bit de Travail (Busy) du Registre d'Etat est positionné. La commande sera exécutée avec un délai de 30 ms si le drapeau 'E' est à 1. Lorsque le Champ d'Identification est localisé sur la disquette, le WD1772 compare le numéro de piste se trouvant dans ce champ avec le nombre qui est dans le Registre de Piste. Si les deux nombres ne s'avèrent pas identiques, une nouvelle comparaison est faite avec le prochain champ d'identification rencontré. Si les deux nombres sont identiques, le numéro du secteur se trouvant dans le champ d'identification est comparé avec le Registre de Secteur. De même que pour la piste, si les deux nombres ne sont pas les mêmes, une nouvelle comparaison est faite avec le numéro de secteur se trouvant dans le prochain Champ d'indentification trouvé. Si le CRC est correct, l'emplacement des données est localisé et, selon la commande, une lecture ou une écriture est réalisée.

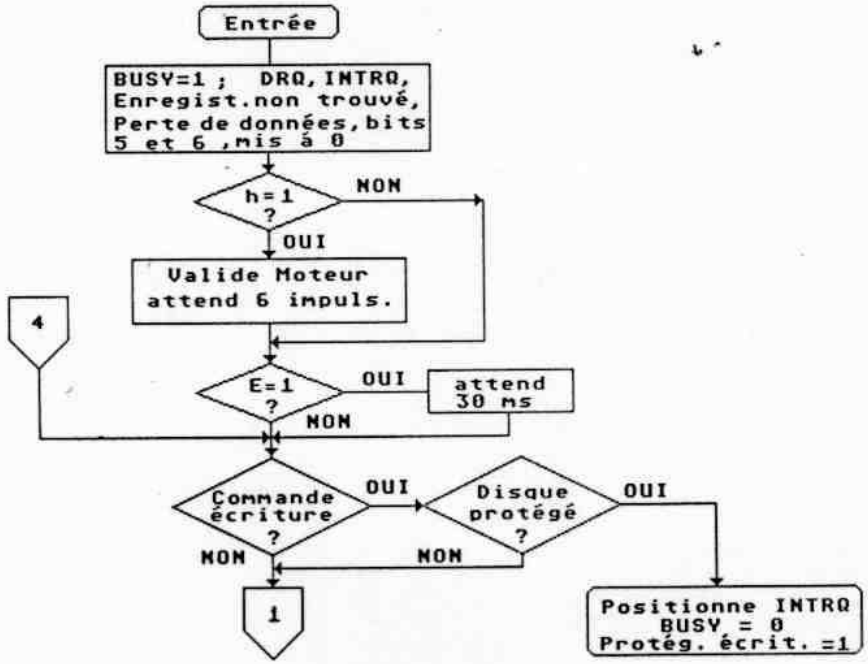
Un numéro de piste, un numéro de secteur et un compte CRC exact doivent être trouvés en quatre tours de disquette, sinon le bit 'Enregistrement non Trouvé' (bit 4) du Registre d'Etat est mis à 1 et la commande se termine par une demande d'interruption (INTRQ).

Chaque commande du Type 2 contient un drapeau, noté 'm', qui spécifie si la commande de lecture ou d'écriture concerne un seul ou plusieurs secteurs.

Si m = 0, la commande ne concerne qu'un seul secteur et une interruption est générée à la fin de l'exécution de la commande.
Si m = 1, la commande sera exécutée sur plusieurs secteurs consécutifs.

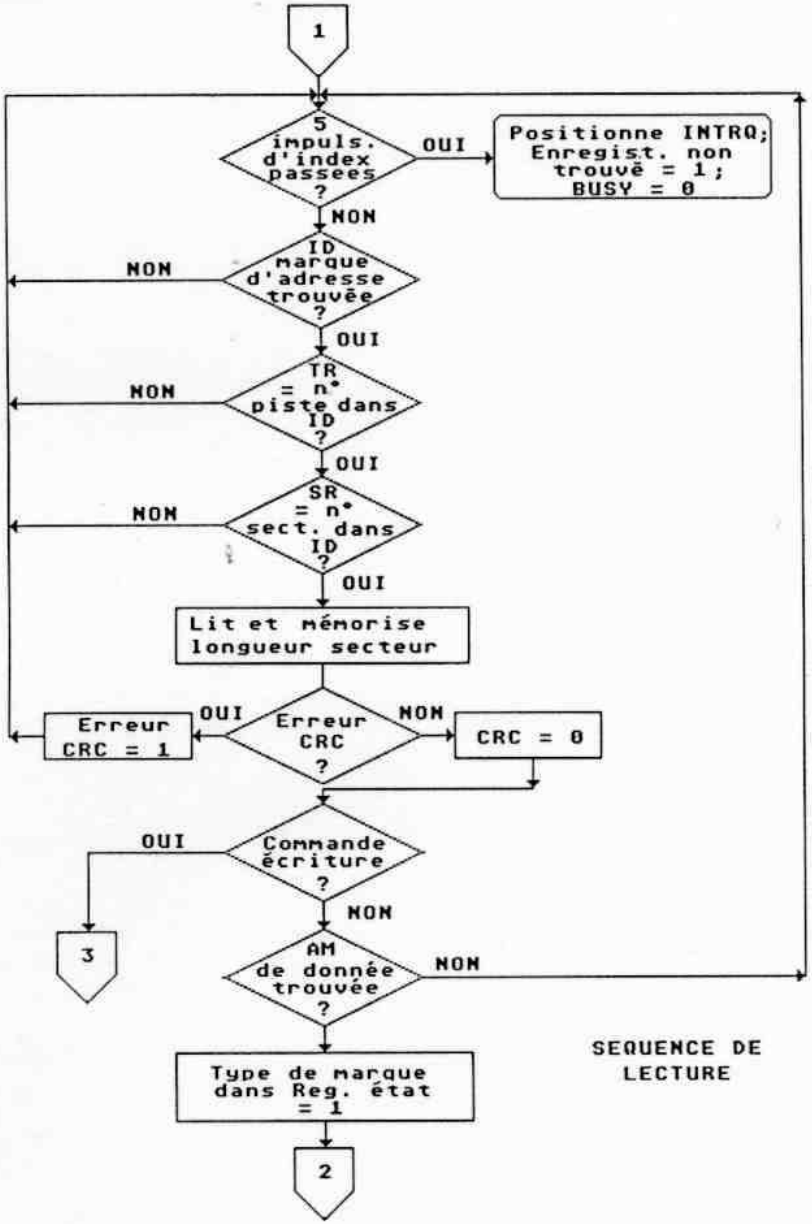
Le Registre de Secteur sera remis à jour par incrémentation de son contenu à chaque opération sur un secteur. La commande sera exécutée jusqu'à ce que le numéro dans le Registre de Secteur soit égal au nombre de secteurs par piste ou qu'une commande 'Interruption Forcée' soit chargée dans le Registre de Commande, ce qui arrêtera l'exécution et provoquera une interruption.

Organigramme des commandes de Type 2



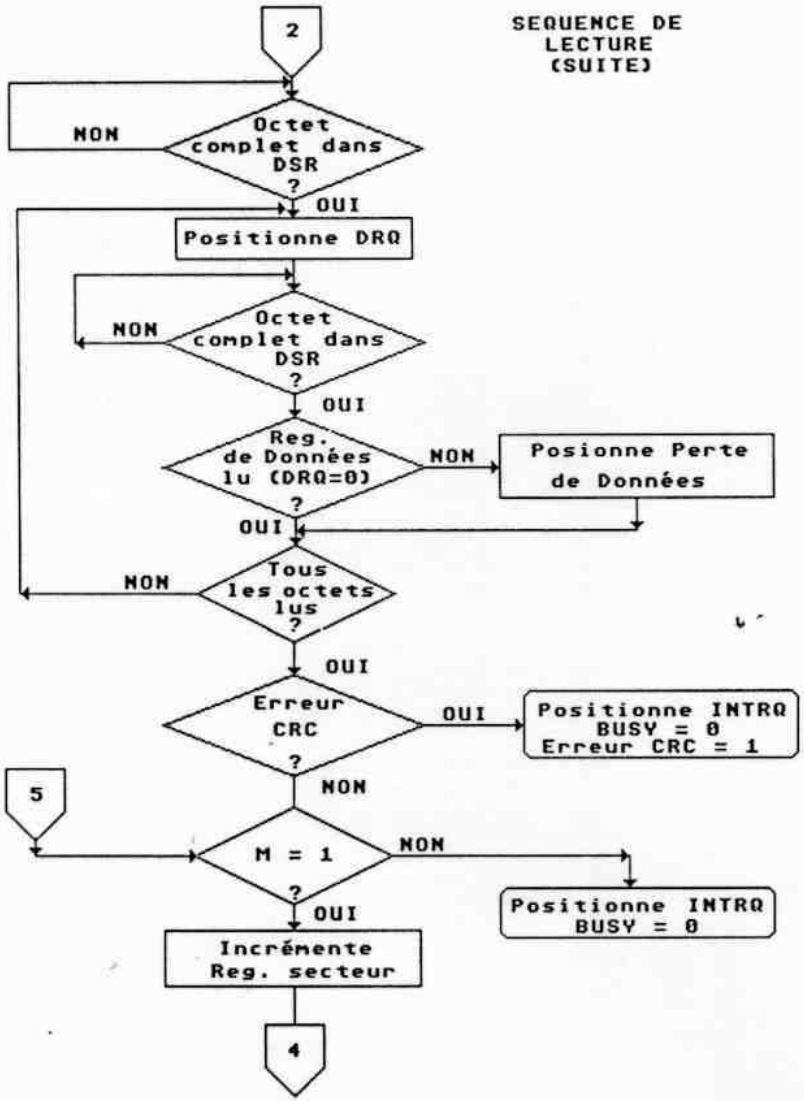





SEQUENCE DE LECTURE

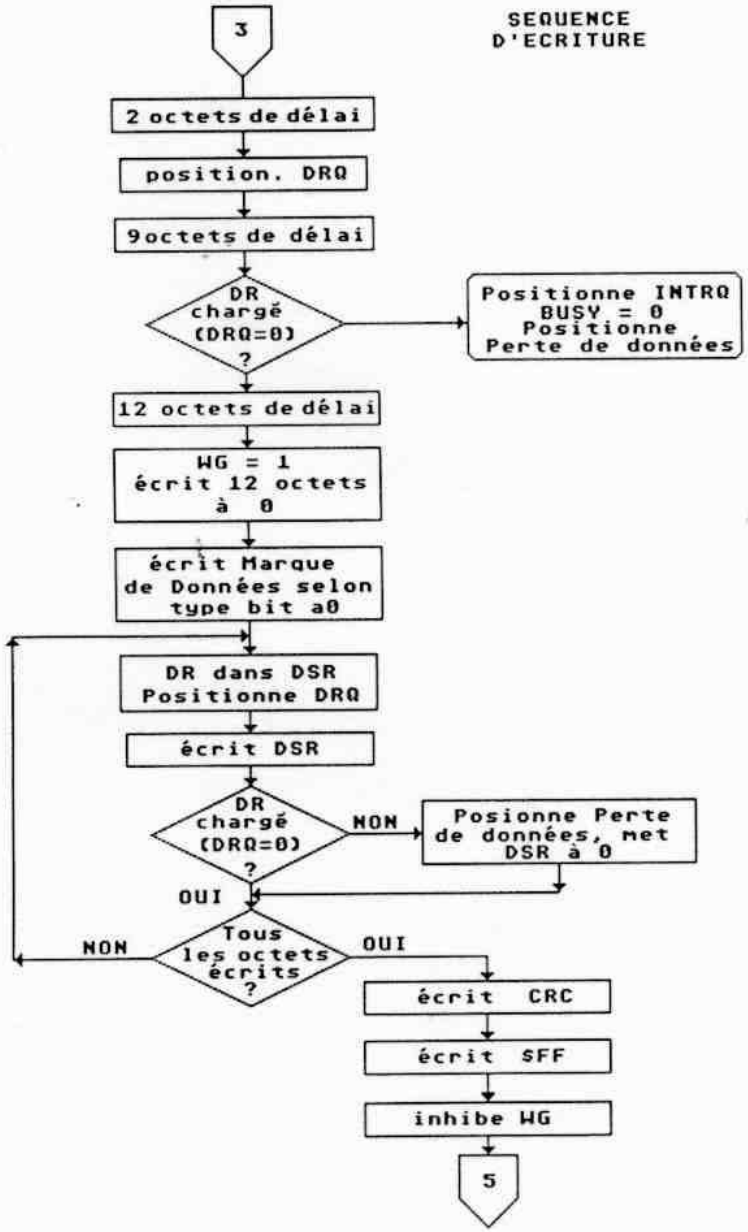
SEQUENCE DE LECTURE (SUITE)








SEQUENCE D'ECRITURE



Lecture de secteur (Read Sector)

Après les opérations préliminaires présentées ci-dessus les données sont envoyées vers le processeur. Une Marque d'Adresse de Données (DAM) doit être trouvée dans le champ de données dans les 30 premiers octets, en simple densité, ou dans les 43 premiers octets en double densité (format ST). Si la DAM n'est pas trouvée, le Champ d'Identification suivant est localisé et une nouvelle recherche est faite. Si, au bout de 5 tours de disquette, la DAM ne peut être localisée, le bit de 'Secteur Non Trouvé' est positionné et la commande prend fin. Lorsque le premier octet a été placé dans le Registre de Données à Décalage (DSR) il est transféré dans le Registre de Données et une demande de données est faite en validant DRQ. Si l'ordinateur n'a pas eu le temps de lire le contenu de DR avant qu'un autre caractère y soit placé, ce caractère est perdu et le bit 'Donnée perdue' est mis à 1 dans le Registre d'Etat. La commande est exécutée jusqu'à lecture complète du champ de données. Si une erreur CRC se produit à la fin d'un champ de données, le bit de 'Erreur CRC' du Registre d'Etat est mis à 1 et la commande prend fin, même si le drapeau de Multiple Secteurs, 'm', est à 1.

A la fin d'une opération de lecture, le type de Marque d'Adresse de Données rencontré dans le champ de données est mis dans le bit 5 du Registre d'Etat comme suit :

bit d'état n°5

| | |
|---|---|
| 1 | Marque de Donnée effacée |
| 0 | Marque de Donnée normale |

Ecriture de secteur

A la réception de cette commande, lorsque le numéro de piste, le numéro de secteur et le CRC sont exacts, une Demande de Donnée est faite par le biais de DRQ. Onze octets en simple densité et vingt deux en double densité sont passés à partir du champ CRC et la sortie 'Ouverture de Porte'(WG) est activée si une réponse a été apportée à la Demande de Données (c'est à dire si la donnée a été chargée dans DR). Si aucune réponse n'a été donnée, la commande est terminée et le bit 'Perte de Donnée' est mis à 1. Si une réponse a été convenablement faite, WG est validé et 6 octets (en simple densité) ou 12 octets (en double densité) à zéro sont écrits sur la disquette puis la Marque d'Adresse de Données est écrite dans le format déterminé par le champ a0 de la commande comme décrit ci-après :

a0 Marque d'Adresse de Donnée (Bit 0)

1 Marque de Donnée effacée
0 Marque de Donnée normale





Le WD1772 écrit ensuite le champ de données et envoie un signal sur DRQ au processeur maître. Si la donnée n'est pas envoyée assez rapidement le bit 'Perte de donnée' est positionné et un octet à zéro est écrit sur la disquette sans que cela mette un terme à la commande en cours. Après l'écriture du dernier octet de donnée, les deux octets CRC sont constitués d'une manière interne et écrits sur la disquette suivis d'un octet à 0xFF. L'Ouverture de Porte est dévalidée et INTRQ est positionné 24 secondes après l'écriture du dernier octet de CRC. La meilleure méthode pour l'écriture partielle de secteur consiste à écrire les données puis à remplir avec des zéros.

## 7.3. Commandes du type 3

Les commandes de type 3 sont au nombre de trois. Ce sont des fonctions de lecture et d'écriture brute, c'est à dire qui permettent la lecture et l'écriture des octets de contrôle nécessaires à la synchronisation et au repérage des pistes et des données de la disquette en plus de octets de données eux-mêmes.

Lecture d'un Champ d'Identification (Read Adress)

A la réception de cette commande le bit de Travail est mis à 1 et le premier Champ d'Identification rencontré est placé dans le registre de donnée. DRQ est alors mis à 1.

Le Champ d'Identification est structuré ainsi:

| Octet | signification |
|---|---|
| 1 | numéro de piste |
| 2 | numéro de face |
| 3 | numéro de secteur |
| 4 | taille du secteur en octets |
| 5 et 6 | CRC du Champ d'Identification |

Bien que le CRC soit transféré dans le Registre de données, il n'en est pas moins testé à la lecture et le bit Erreur de CRC est positionné si celui-ci n'est pas valide. Le numéro de piste est écrit dans le Registre de Secteur, permettant ainsi de le tester. A la fin de l'opération une interruption est générée et le bit de Travail est remis à 0.

## Lecture brute de piste

La lecture débute au début de la première impulsion d'index reçue et se poursuit jusqu'à la suivante (ce qui représente un tour complet de la disquette). Tous les Caractères de Remplissage (GAP), les en-têtes, et les données sont lus et tranférés dans le registre de données. A chaque octet, DRQ est positionné. La lecture des octets est synchronisée avec toutes les Marques d'Adresse rencontrées (Marque d'Adresse d'ID et Marque d'Adresse de Données de chaque secteur). Le bit Perte de données est mis à 1 si aucune réponse n'est faite dans les temps à la demande DRQ. Une interruption est générée à la fin de l'exécution de la commande.

Cette commande possède plusieurs caractéristiques qui la rendent utile pour les opérations de diagnostic : Aucun test de CRC n'est fait, les Caractères de Données (GAP) sont inclus dans les données, et le détecteur de Marques d'Adresse est actif tout au long de la commande. Les Marques d'Adresse du Champ d'Identification, le Champ d'Identification, Les Marques d'Adresse des Données, les données et le CRC de chaque secteur sont lus correctement. Certaines erreurs peuvent apparaître dans les GAP et les octets de synchronisation, dues à la synchronisation.

## Ecriture et formatage de piste

Formater une disquette est une opération assez simple lorsqu'on dispose d'une bonne taille mémoire. Les données et les GAP doivent être fournis par l'interface maître (ordinateur). Le formatage se fait en positionnant la tête de lecture/écriture sur la piste désirée et en exécutant la commande d'écriture de piste.

L'écriture débute dés la première impulsion d'index reçue et se poursuit jusqu'à ce qu'une autre impulsion d'index soit rencontrée. Une interruption est alors générée. La broche Demande de Données (DR) est immédiatement activée à la réception de la commande mais l'écriture ne débute qu'après le chargement du premier octet dans le Registre de Données. Si le Registre de Données n'est pas chargé dans le temps nécessaire au chargement de trois octets, le bit de Travail est remis à 0, le bit de Perte de données est mis à 1 et une interruption est générée. De plus un octet à zéro est écrit.

Cette séquence se poursuit d'une marque d'index à une autre. Théoriquement toute valeur placée dans le Registre de données est écrite telle quelle avec une impulsion de fréquence. Cependant si une valeur comprise entre $F5 et $FE est écrite dans le Registre de Données, celle-ci est interprétée comme étant une Marque d'Adresse écrite sans impulsion de fréquence et non prise en compte dans le CRC.



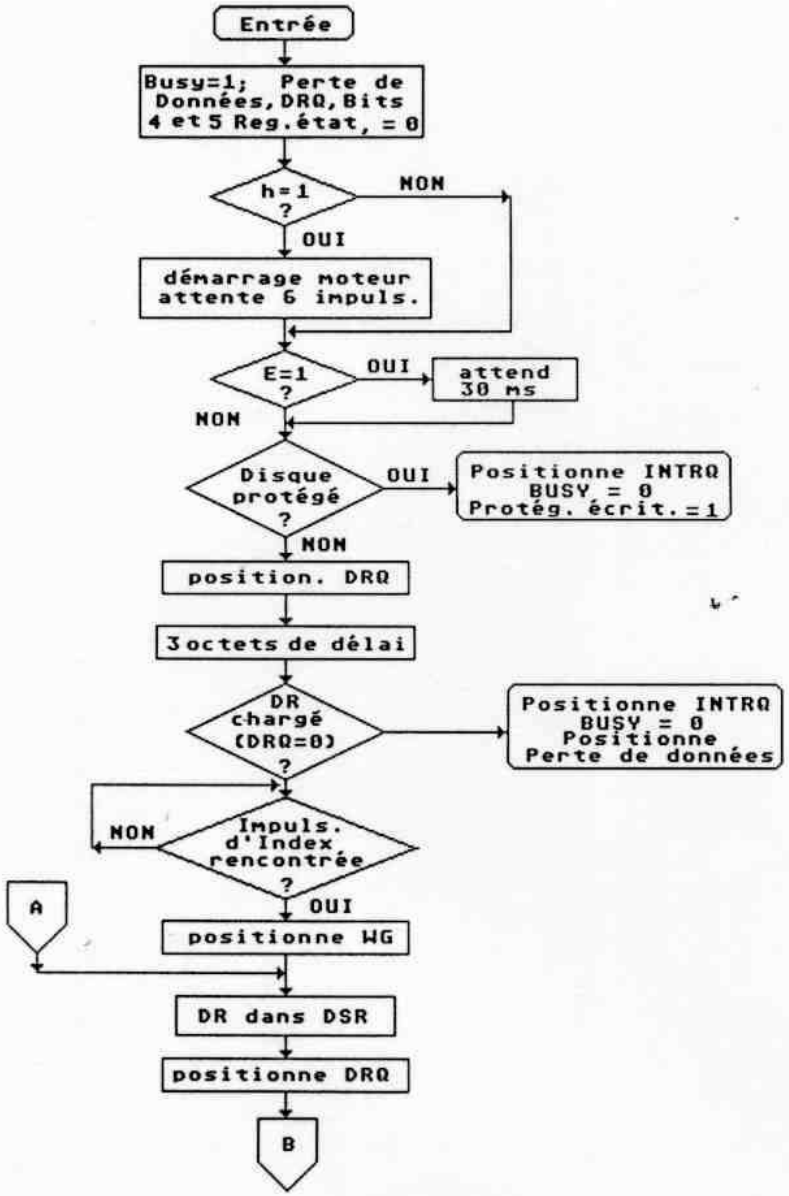


Le générateur de CRC est initialisé à chaque transfert d'un octet à $F5, en double densité, ($F8 à $FE en simple densité), depuis le Registre de Données vers le Registre de Données à Décalage. Un octet à $F7 provoquera l'écriture de deux caractères de CRC. En conséquence les caractères allant de $F5 à $FE ne doivent pas apparaître dans les Caractères de Remplissage GAP, les données et les Champs d'Identifications. De plus le CRC doit être provoqué par l'écriture de $F7 (Voir tableau).

Les disquettes peuvent être formatées selon différents formats: IBM ou System 34 avec des secteurs de taille de 128, 256, 512, ou 1024 octets.

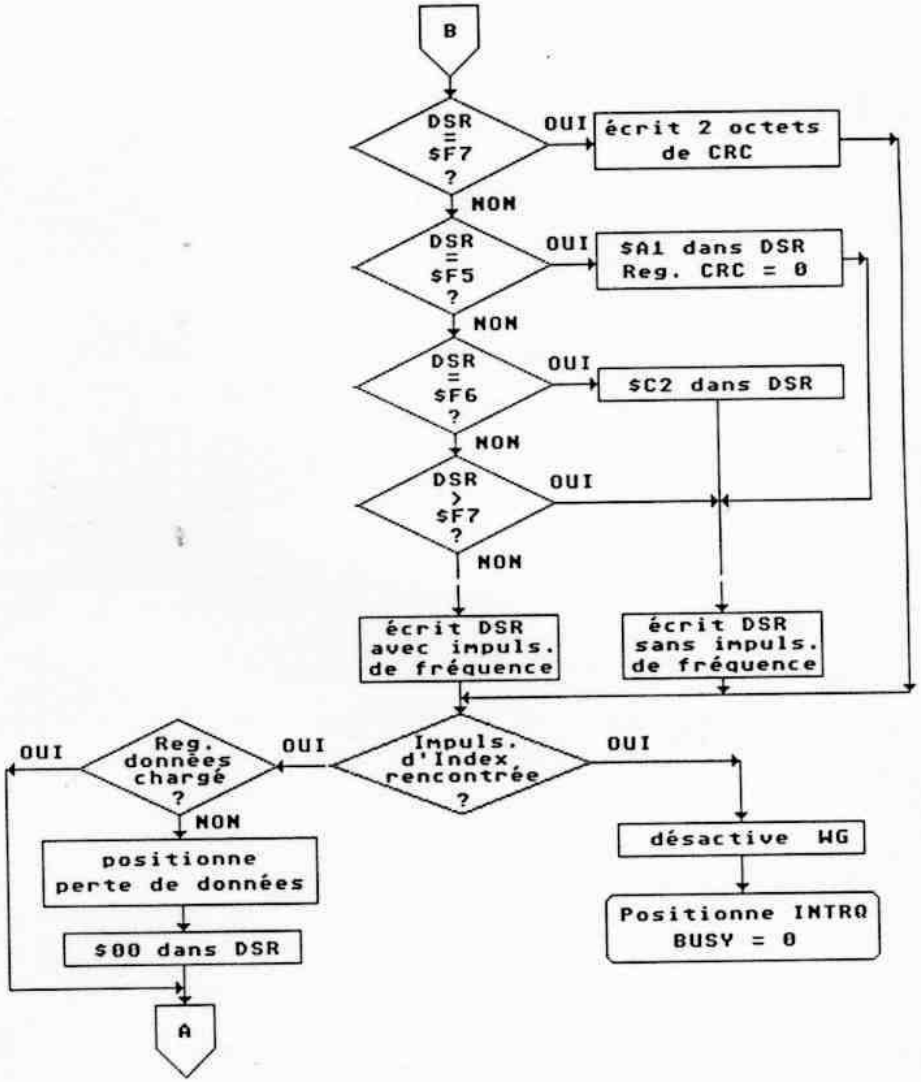
| Donnée dans Reg. Donnée | simple densité | double densité |
|---|---|---|
| 00 à $F4 | écrit 00 à $F4 impuls. de fréq.=$FF | écrit 00 à $F4 |
| $F5 | interdit | écrit $A1 sans impuls. de freq. (entre bit 4 et 5, octet de synchro.).remise à 0 de CCR |
| $F6 | interdit | écriture de $C2 sans impuls. de fréquence entre le bit 3 et 4. |
| $F7 | génération de 2 octets de CRC | |
| $F8 à $FB | écrit de $F8 à $FB impuls. de fréq.=$C7 mise à 0 de CRC | écrit de $F8 à $FB |
| $FC | écrit $FC impuls. de freq.=$D7 | écrit $FC |
| $FD | écrit $FD impuls. de freq.=$FF | écrit $FD |
| $FE | écrit $FE, impuls.de freq.=$C7, CRC = 0 | écrit $FE |
| $FF | écrit $FF, impuls. de freq. = $FF | écrit $FF |

Organigramme de la commande Ecriture brute de piste







## 7.4 Commande de type 4

La commande Interruption Forcée sert essentiellement à mettre fin à une commande d'écriture ou de lecture sur plusieurs secteurs consécutifs (bit 'm' positionné). Elle peut également servir à s'assurer de l'état des bits lors d'une commande de Type 1, du Registre d'Etat.

Cette commande est la seule à pouvoir être chargée à n'importe quel moment dans le Registre de Commande, quelque soit l'état du bit de Travail. Si une commande est en cours lors du chargement d'Interruption Forcée, la commande prendra fin lorsque l'instruction en cours dans l'ALU (génération de CRC, comparaison) sera finie. Le bit de Travail sera remis à 0.

Le quartet faible de la commande détermine les conditions d'interruption comme suit:

I0 non pris en compte
I1 non pris en compte
I2 Interruption à chaque Impulsion d'Index (tour de disquette)
I3 Interruption immédiate

Les conditions d'interruption sont prises en compte lorsque leur bit correspondant dans I0-I3 est à 1. Lorsque ces conditions sont rencontrées la ligne INTRQ est placée à l'état haut. Si tous les bits sont à zéro (commande $D0), aucune interruption ne sera générée mais la commande en cours sera interrompue instantanément. Si le bit I3 est à 1 (commande $D8), la commande sera également arrêtée mais de plus une interruption sera générée. La lecture du Registre d'Etat ou l'écriture d'une nouvelle commande ne remet pas automatiquement à 0 l'interruption. Pour ce faire il est nécessaire d'envoyer une commande $D0 après la commande $D8.

Une attente de 32 s en simple densité et de 16 s en double densité doit être faite avant l'envoi d'une nouvelle commande suite à une Interruption Forcée. Une commande chargée trop tôt empêcherait la prise en compte d'Interruption Forcée.

## 8. Le Registre d'Etat

Lors de la réception d'une commande, exceptée Interruption Forcée, le bit d'occupation est mis à 1 et le reste du Registre d'Etat est remis à jour ou nettoyé. Si une Interruption est reçue durant l'exécution d'une fonction, le bit d'Occupation est remis à 0, mais les autres registres ne sont pas modifiés. Si une Interruption Forcée est reçue alors que le bit de travail est à 0 le Registre d'Etat est mis à jour. Dans ce cas le Registre d'Etat reflète le résultat d'une commande de Type 1.





Le programmeur peut lire le Registre d'Etat par programme ou en se servant de la ligne DRQ ou par l'emploi d'Interruption. Lorsque le Registre de données est lu (ou écrit), le bit DRQ du Registre d'Etat et la broche DRQ sont remis à 0.

Il est possible par programme de tester le bit de Travail pour déterminer si une commande est finie au lieu de se servir de la ligne INTRQ. Lors de l'emploi de la ligne INTRQ, il est recommandé de ne pas tester le bit de Travail du Registre d'Etat car la lecture du registre d'état remet à 0 INTRQ.

## 9. Formats d'une piste

### 9.1. Format normal d'une piste sur le ST

Une piste appelée normale est une piste qui a été formatée par le système du ST à partir de l'option formatage du bureau.

Une piste ainsi formatée possède 9 secteurs de 512 octets chacun.

Elle se décompose ainsi:

| | | |
|---|---|---|
| GAP 1 | 60 x $4E | octets de remplissage de début de piste |
| (à chaque secteur, c'est à dire 9 fois) | | |
| GAP 2 | 12 x $00 | octets de remplissage |
| SYNC | 3 x $F5 | octets de synchronisation |
| ID AM | 1 x $FE | Marque d'Adresse du Champ d'Identification |
| NPISTE | 1 octet | numéro de piste |
| NFACE | 1 octet | numéro de face |
| NSECT | 1 octet | numéro de secteur |
| TAILSEC | 1 octet | Taille secteur codée |
| ID CRC | 2 octets | Somme Test du Champ d'Identification |
| GAP 3 | 22 x $4E<br>13 x $00 | octets de remplissage |
| SYNC | 3 x $F5 | octets de synchronisation |
| DAM | 1 x $FB | Marque d'Adresse des Données |
| DONNEES | 512 octets | Données |
| CRC | 2 octets | Somme Test des données |
| GAP 4 | 40 x $4E | octets de remplissage |
| (fin des secteurs) | | |
| GAP 5 | entre 664 et 1400 octets | octets de remplissage |

Le début de la piste commence avec une zone de 60 octets de $4E appelé GAP 1, puis suit une structure qui se retrouve pour les 9 secteurs de la piste :

12 octets à 0 appelés GAP 2, suivis de 3 octets de $F5 (qui seront lus $A1 après tranformation) servant de repérage au WD1772. Suit la Marque d'Adresse ($FE) du Champ d'Identification qui signale que l'octet suivant est le premier octet du Champ d'Identification (numéro de piste). Le GAP 3, formé de 22 octets à $4E suivis de 13 octets à 0, vient ensuite. Puis les 3 octets de Synchronisation ($F5, lus $A1), suivis de la Marque d'Adresse Données signalent le début des données qui occupent 512 octets et se terminent par le mot de test de somme. 40 octets à $4E suivent les données (GAP 4).

A la fin de la piste un dernier bloc de Caractères de remplissage (GAP 5) est écrit. Le système essaie d'y écrire 1400 octets de $4E. Il n'y réussit jamais car il reçoit l'impulsion d'index suivante lui indiquant que la piste est complètement remplie avant d'avoir tout écrit. Le nombre des octets écrits peut ne pas être identique à chaque piste (variation de 2%).

## 9.2. Formats spéciaux

Les variations de formats sont possibles dans la limites de contraintes ci-dessous:

1. La taille des secteurs doit être de 128, 256, 512, ou 1024 octets.

2. Le GAP 3 ne peut varier du format recommandé (voir tableau).

3. 3 octets de $F5 ($A1) doivent être écrits en double densité.

Ces contraintes respectées, les pistes peuvent avoir d'autres structures. Il est ainsi possible de faire varier le nombre de pistes formatées jusqu'à 82 (sans garantie qu'elles puissent être lues par tous les lecteurs, Atari Corp. ne garantissant que les 80 premières pistes), de changer le nombre de secteurs et la taille de ces secteurs sur une piste, de modifier le nombre de Caractères de Remplissage, etc.





## LE MONITEUR MONOCHROME SM 124

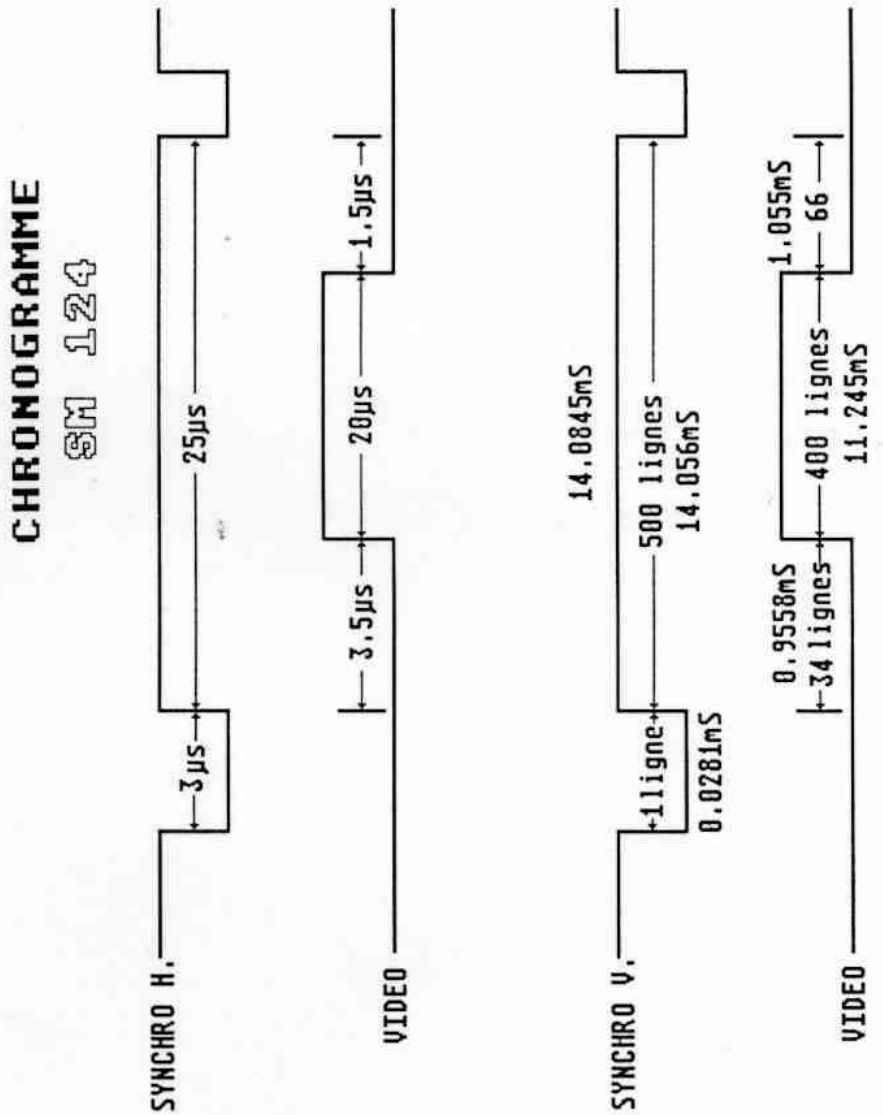
## LE MONITEUR COULEUR SC 1224

| | DESCRIPTION | SM 124 | SC1224 |
|---|---|---|---|
| TUBE | TAILLE | 12" (11" V) | 12" (11" V) |
| | PHOSPHORE | | POINT |
| | LARGEUR DE POINT | | 0.38mm |
| INTERFACE | CONNECTEUR | DIN MALE 13 BROCHES | DIN MALE 13 BROCHE |
| | AUDIO | 1 VPP/1.0K | 1 VPP/1.0K |
| | ENTREE VIDEO | DIGITALE | ANALOGIQUE RVB |
| | IMPEDANCE EN ENTREE | 75 /1.0VPP | 75 /0-1.0VPP |
| | "DOT RATE" | 32 Mhz | 16 Mhz |
| | SYNCHRO HORIZONTALE | TTL/3.3K /NEG | TTL/3.3K /NEG |
| | SYNCHRO VERTICALE | TTL/3.3K NEG | TTL/3.3K /NEG |
| | ZONE D'AFFICHAGE | 210mm X 131mm | 210mm X 150mm |
| | RESOLUTION | 640 X 400 | 640 X 200 |
| DEFLECTION | FREQUENCE HORIZONTALE | 35.7 Khz | 15.75 Khz |
| | " VERTICALE | 71.2 Khz | 60 Hz |
| | RETRACAGE HORIZONTAL | 4 usec | 10 usec |
| | " VERTICAL | 700 usec | 1 msec |
| | TEMPS DE MONTEE | 10 nsec | 15 nsec |
| | TEMPS DE CHUTE | 10 nsec | 15 nsec |
| | DISTORTION GEO | 1.5% | 1.5% |
| | NON-LINEARITE | 9% | 7% |
| | DISCONVERGENCE (CENTRE) | N/A | 0.6 mm |
| | DISCONVERGENCE (COINS) | N/A | 1.0 mm |
| CONTROLE | INTERRUPTEUR MARCHE/ARRET | INTERRUPTEUR/MOLETTE | INTERRUPTEUR/MOLETTE |
| | VOLUME AUDIO | MOLETTE | MOLETTE |
| | LUMINOSITE | MOLETTE | MOLETTE |
| | CONTRASTE | MOLETTE | MOLETTE |

1. L'intérrupteur marche/arrêt et le bouton de volume peuvent être combinés.
2. Sur le SM 124, luminosité et contraste peuvent être combinés.
3. La réponse de la fréquence audio s'étalonne de 100 Hz à 15 Khz, 3db.

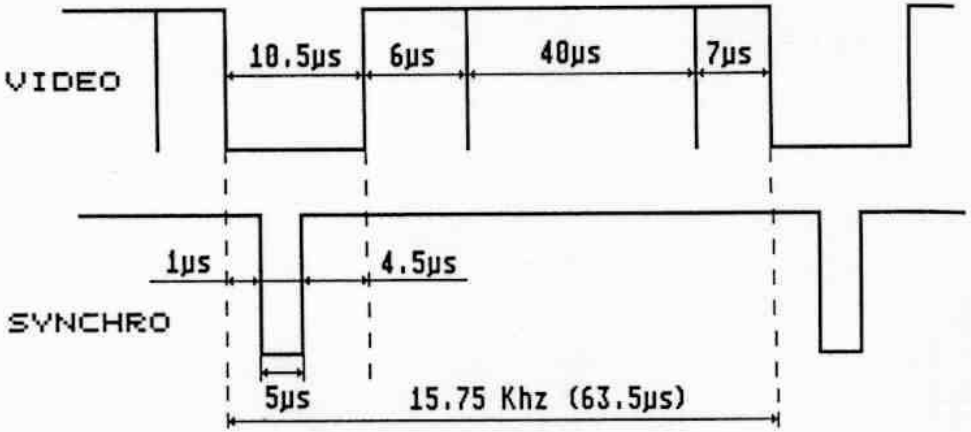




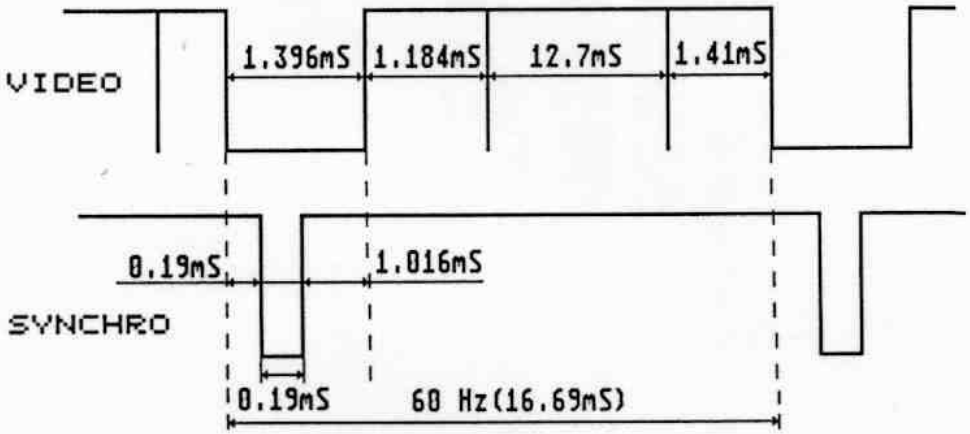


# SC 1224

## SYNCHRONISATION HORIZONTALE




## SYNCHRONISATION VERTICALE



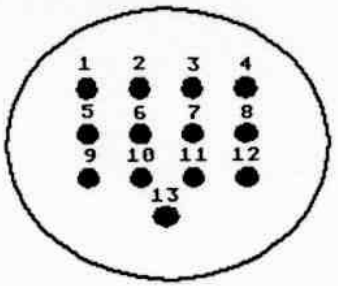




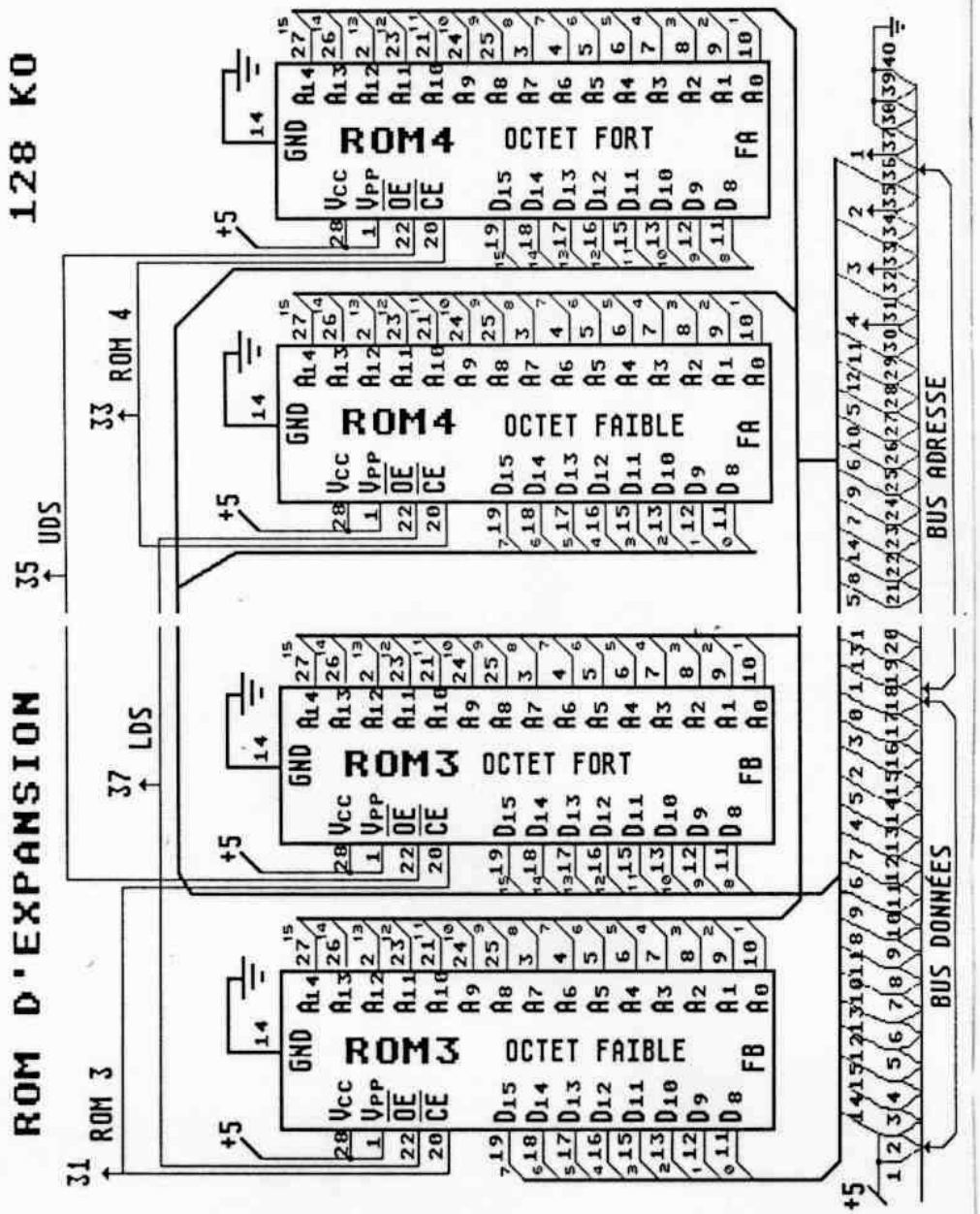

**BROCHAGE DU CONNECTEUR VIDEO**

**DIN FEMELLE 13 BROCHES**

Le schéma ci-dessous représente le connecteur femelle vu côté unité centrale, et non le connecteur mâle du moniteur.




1. AUDIO
2. NON CONNECTE
3. NON CONNECTE
4. "MASSE" POUR MONO, NON CONNECTE POUR RVB
5. NON CONNECTE
6. VERT
7. ROUGE
8. 12 VOLTS
9. SYNCHRONISATION HORIZONTALE
10. BLEU
11. VIDEO
12. SYNCHRONISATION VERTICALE
13. MASSE








LE MANUEL DE REFERENCE

DU

GEMDOS ATARI

## 1. INTRODUCTION

> CECI EST UN DOCUMENT PRELIMINAIRE.
> IL SE PEUT QU'IL CONTIENNE DES
> IMPRECISIONS ET DES INEXACTITUDES.
> PRIERE DE RENDRE COMPTE DE TOUTES
> LES 'BUGS' DE CE DOCUMENT A ATARI.

Voici le Manuel de l'Utilisateur du GEMDOS d'Atari. Il décrit le contenu et l'emploi du GEMDOS sur l'Atari ST. Ce manuel se divise en trois parties : un cours d'introduction pour les débutants, un manuel de référence pour les auteurs d'applications, et des appendices pour les cracks du GEMDOS.

L'introduction au GEMDOS est une préparation en douceur aux bases du GEMDOS. Son but est de permettre aux débutants un démarrage le plus rapide possible . Elle fournit des exemples de programmes destinés à se familiariser avec la plupart des fonctions GEMDOS, réunies dans de simples commandes ,appelées "shell". Ce cours traite aussi des pièges les plus communs et des raccourcis utiles.

Le Manuel de référence du GEMDOS est la bible du développeur. Il englobe les conventions d'appel, la manipulation de fichiers et d'identificateurs, l'exécution de programmes, et tous les appels du GEMDOS.

Les appendices contiennent les détails les plus secrets et s'adressent à ceux qui veulent pousser le GEMDOS à ses limites ; développeurs (et plus généralement curieux) qui veulent 'tout savoir' des aspects les plus obscurs du système.

Pour un emploi efficace de ce manuel, le lecteur devra être familiarisé avec le langage C et l'assembleur. Une connaissance du MSDOS , d'UNIX, et des routines de la bibliothèque C standard serait utile.

## 2. L'APPEL DU GEMDOS

Le GEMDOS emploiE les conventions d'appel C de l'Alcyon (Digital Research). Il est à noter que ces conventions peuvent être différentes sur d'autres compilateurs C 68000. Si vous vous êtes servi d'un autre compilateur il se peut que vous ne puissiez appeler directement le GEMDOS. Vérifiez la compatibilité sur la documentation de votre compilateur.

Les paramètres sont déposés dans la pile en ordre inverse de leur déclaration. Le numéro de la fonction GEMDOS (un mot) est déposé en dernier. L'appel est réalisé par l'exécution d'une instruction "TRAP #1" du 68000. Le trap peut être fait soit en mode utilisateur soit en mode superviseur.

ATTENTION:

Une application tournant en mode superviseur peut se retrouver en mode utilisateur après un appel de l'AES.

Photo de la pile
(juste avant l'appel d'un trap GEMDOS)

| pile | contenu |
|---|---|
| (sp) | MOT numéro de la fonction |
| 2(sp) | 1er paramètre |
| X(sp) | 2éme paramètre |
| Y(sp) | 3éme paramètre |
| . | ..... |
| . | ... etc ... |
| . | ..... |

Les résultats sont renvoyés dans D0. Les registres D0 à D2 et A0 à A2 peuvent être modifiés. Les registres D3 à D7 et A3 à A7 sont toujours préservés. C'est à la fonction appelante de retirer les paramètres (le numéro de fonction compris) de la pile après l'appel.

Le compilateur C d'Alcyon ne génère pas d'instruction trap, de telle sorte que la plupart des applications emploient un court lien en assembleur. Il ressemble typiquement à ceci :

```
    text
*
*   lien GEMDOS pour le C d'Alcyon
*
*   NB.
*        Ce lien n'est pas réentrant et ne peut être
*        partagé par le programme principal et une
*        routine  sous interruption.
*
*
    .globl    _gemdos

    _gemdos:
         move.l    (sp)+,sauvereg     * sauve l'adr. de retour
         trap      #1                 * appel du GEMDOS
         move.l    sauvereg,-(sp)     * restitue  adr. retour
         rts                          * revient

    bss
    sauvereg:  ds.l      1           * adr. de retour sauvée
```





## 3. NOMS DE FICHIERS

Un nom de fichier se compose d' un identificateur de disque optionnel , suivi par un chemin et un simple nom de fichier. Un identificateur de disque est formé d'une simple lettre ( de A à P) suivie par ':'. Si l'identificateur de disque manque, le disque par défaut est employé. Un chemin se compose d'une liste de simple noms de fichiers séparés par des slash arrières ( \ ). Si le chemin commence par un slash arrière, il est rattaché au catalogue racine, sinon il est rattaché au catalogue courant. Un simple nom de fichier se compose d'un à huit caractères, pouvant être suivi d'un '.' et de zéro à trois autres caractères.

Les caractères autorisés dans un nom de fichier et dans un chemin comprennent l'alphabet (A-Z), les chiffres (0-9), et la plupart des ponctuations. Les points, deux_points, slashs, slashs arrières, points d'interrogation, astérisque, caractéres de controle (englobant les NULs), et les caractères plus grands que 0x7F ne doivent jamais apparaître dans un nom de fichier. Les minuscules sont converties en majuscules.

L'ensemble d'une spécification de fichier ne peut excéder 125 caractères.

| Caractères Autorisés dans un nom de fichier |
|---|
| Lettres A-Z , a-z |
| Nombres 0_9 |
| ! ¨ @ # $ % ^ & ( ) + - = |
| ~ ` ; ' " , < > \| [ ] { } |

Dans un chemin , '.' désigne le catalogue courant et '..' désigne le catalogue parent du catalogue courant.

Ainsi les chemins :

'..\..\foo'

et

'.\.\.\..\.\.\..\.\foo'

désignent le même fichier ,deux catalogues plus haut que le catalogue courant. (Il n'y a pas de catalogue parent d'une racine.)

Il existe trois périphériques 'à caractères'. Ils ne fonctionnent qu'avec les fonctions Fread(), Fwrite(), Fopen(), Fcreate(), Fclose(), et les fonctions standards d'entrée/sortie:

| noms | identificateur | périphérique |
|---|---|---|
| CON: ou con: | 0xffff (-1) | console système |
| AUX: ou aux: | 0xfffe (-2) | port RS232 |
| PRN: ou prn: | 0xfffd (-3) | port parallèle |

Un appel Fopen() ou Fcreate() sur un périphérique 'à caractères' retournera un identificateur de périphérique 'à caractères'. L'identificateur est un MOT négatif ,mais non un LONG négatif (lequel serait un symbole d'erreur).





## 4. OPERATIONS SUR LES FICHIERS

Le GEMDOS n'impose pas de restrictions sur ce que doit contenir un fichier. La plupart des applications adoptent le fait qu'un fichier de texte se compose de lignes séparées par des retours de chariot, 'Ctrl z' (0x1A) indiquant la fin du fichier. Le format d'un fichier exécutable est documenté dans les Appendices.

Les appels GEMDOS Fcreate() et Fopen() retournent un entier positif court sur 16 bits, appelé identificateur, qui fait référence au fichier ouvert. Un fichier peut être ouvert en écriture, en lecture, ou en écriture et lecture. Fermer un fichier libère l'identificateur, autorisant ainsi son réemploi.

Il existe trois sortes d'identificateurs: Les identificateurs standards numérotés de 0 à 5, qui peuvent s'appliquer à un périphérique 'à caractères' ou à un fichier; les identificateurs non standards débutant à 6, et qui ne font référence qu'à des fichiers; les identificateurs 'à caractères' qui ne font référence qu'à des périphériques 'à caractères'; les numéro des ces identificateurs vont de 0xfffd à 0xffff. Ce sont des MOTS négatifs et non des LONGS MOTS négatifs.

Quand un programe exécute un appel de Pexec(), l'enfant du processus hérite des identificateurs standards du parent. L'identificateur '0' désigne souvent "l'entrée standard" ou la "sortie standard", normalement connectée à la console, CON:. Il est possible de rediriger l'entrée/sortie standard d'un programme grace aux fonctions Fdup() et Fforce(), de ou vers un fichier ou un autre périphérique 'à caractères'.

Quand un changement de disquette (média change) se produit, tous les fichiers ouverts sur le disque retiré sont fermés par le GEMDOS.

ERREURS

Le concept d'"erreur standard" sortie n'exite pas.

## 5. LES PROCESSUS DU GEMDOS

Bien que le GEMDOS ne permette pas le multi-tâche, il est possible d'exécuter un processus en sous-routine. Un programme peut en appeler un autre avec Pexec(). Le processus fils se terminera avec un code de retour sur un MOT.

Un processus possède toute les fichiers qu'il ouvre et toute la mémoire qu'il alloue. Les fichiers sont fermés et la mémoire désallouée à la fin du processus.

Avant la fin réelle du processus le GEMDOS appelle le vecteur d'extension 0x102. Ceci permet à l'application d'essayer de retomber sur ses pieds après une erreur éventuelle.

Le modèle de mémoire employé par le GEMDOS est le même que celui du MSDOS. Un programme se déroule dans la 'TPA'( zone de programme transitoire ). Les 256 premiers octets de la TPA sont appelés page de base du programme. Ils contiennent des informations spécifiques au programme.

Structure d'une page de Base

| offset | nom | description |
|---|---|---|
| 0x00 | p_lowtpa | base de la TPA |
| 0x04 | p_hitpa | haut de la TPA |
| 0x08 | p_tbase | base des codes (texte) |
| 0x0C | p_tlen | taille de la partie texte |
| 0x10 | p_dbase | base des données initialisées |
| 0x14 | p_dlen | taille des données initialisées |
| 0x18 | p_bbase | base des données non initialisées |
| 0x1C | p_blen | taille des données non initial. |
| 0x20 | p_dta | adresse du DTA |
| 0x24 | p_parent | adr. de la page de base du parent |
| 0x28 | (réservé) | |
| 0x2C | p_env | adr. de la chaine d'environnement |
| 0x80 | p_cmdlin | image de la ligne de commande |

'p_lowtpa' pointe vers la page de base (vers lui_même).
'p_hitpa' pointe vers vers le haut de la TPA, le premier octet libre.
'p_tbase', 'p_tlen', etc.. contiennent respectivement l'adresse et la taille du texte, des données initialisées ,des données non initialisées.
'p_parent' pointe vers la page de base du processus parent.
'p_env' pointe vers la chaine d'environnement (voir Pexec()).





Le premier octet de l'image de la chaîne de commande contient le nombre de caractères de la commande, l'image va du second au dernier octet. Il n'est pas obligatoire que la chaîne se finisse par un caractère nul.

L'application reçoit le contrôle à l'adresse de départ de sa zone de texte. Le second long-mot de la pile , 4(sp), contient un pointeur sur la page de base du programme. Normalement toute la mémoire libre est allouée au nouveau processus. Si l'application se sert de Malloc() ou de Pexec() alors elle doit reloger sa pile et appeler Mshrink() afin de libérer de la mémoire pour le système. La pile commence au sommet de la TPA et s'étend vers la BSS (zone de données non initialisées).

## 6. LES VECTEURS D'EXTENSION

Le 68000 utilise les vecteurs 0x02 à 0xFF, ce qui correspond aux adresses 0x0000 à 0x03FC. Le GEMDOS ajoute huit vecteurs logiques de numéro 0x100 à 0x107. L'adresse d'un vecteur logique est indéterminée. C'est le BIOS qui les fournit.

Attribution des vecteurs logiques

| vecteur | usage |
|---|---|
| 0x100 | Timer |
| 0x101 | Gestion des erreurs critiques |
| 0x102 | Gestion de fin de programme (^C) |
| 0x103 - 0x107 | réservé pour un emploi ultérieur |

### 0x100 Timer

Ce vecteur est appelé sur une période de 50hz par le BIOS pour maintenir l'heure et la date du système et pour des tâches internes. 4(sp), contient le nombre de millisecondes depuis la dernière interruption timer.

Pour intercepter ce vecteur utilisez l'appel BIOS qui place et retire un vecteur. Chaque module devra d'abord effectuer ses propres routines puis rendre la main à l'ancien vecteur. Les modules devront être le plus court possibles : flaner ici affecterait les performances du système.

Tous les registres (sauf SP et USP) sont modifiés par le GEMDOS. Le BIOS s'occupe de la sauvegarde des registres D0 à D7, A0 à A6; aussi les modules chaînés à cette interruption n'ont pas à sauver et à restituer les registres.

### 0x101 Module d'erreur critique

Le module d'erreur critique est appelé par le BIOS pour gérer certaines erreurs (erreur de disque et demande de changement de disque de la fonction rwabs() ).Il permet à l'application de gérer ces erreurs et de faire ce qui convient.





le numéro d'erreur se trouve dans 4(sp). Selon l'erreur, d'autres arguments peuvent être dans la pile. Le module d'erreur critique doit sauver les registres D3 à D7,A3 à A6.

Au retour du module, D0.L contient un code de retour:

| valeur dans D0.L | signification |
|---|---|
| 0x00010000 | nouvel essai à tenter |
| 0x00000000 | feindre l'absence d'erreur |
| 0xffffffXX | abandonner avec une erreur |

Le module d'erreur critique système retourne simplement -1.

### 0x102 Module de fin (^C)

Avant la fin réelle d'un processus, le GEMDOS appelle le vecteur de fin. Si le vecteur de fin pointe sur un 'RTS' (le cas par défaut), le processus est terminé. Si l'application ne souhaite pas être terminée elle doit faire un long-saut à une routine appropriée.

## 7. DESCRIPTIF DES ERREURS

Tous les numéros d'erreur sont négatifs. Il existe deux plages d'erreurs : Les erreurs BIOS rangées de -1 à -31 et les erreurs GEMDOS rangées de -32 à -127.

### Codes Erreurs BIOS

| nom | numéro | description |
|---|---|---|
| E_OK | 0 | OK (pas d'erreur) |
| ERROR | -1 | Erreur générale |
| EDRVNR | -2 | Lecteur non près |
| EUNCMD | -3 | Commande inconnue |
| E_CRC | -4 | Erreur CRC (somme de contrôle) |
| EBADRQ | -5 | Mauvaise demande |
| E_SEEK | -6 | Erreur de positionnement |
| EMEDIA | -7 | Media inconnu |
| ESECNF | -8 | Secteur introuvable |
| EPAPER | -9 | Manque de papier |
| EWRITF | -10 | Erreur d'écriture |
| EREADF | -11 | Erreur de lecture |
| | -12 | (inutilisé) |
| EWRPRO | -13 | Tentative d'écriture sur une disquette protégée en écriture |
| E_CHNG | -14 | Changement de disquette détecté |
| EUNDEV | -15 | Périphérique inconnu |
| EBADSF | -16 | Mauvais formatage de secteur |
| EOTHER | -17 | Insérer une autre disquette (demande) |

'EOTHER' est vraiment une requête du BIOS à insérer une autre disquette dans le lecteur A:. Le numéro du disque 'virtuel' (0 ou 1 ) se trouve dans 6(sp). Cette caractéristique est employée pour leurrer le système en lui faisant croire qu'un système avec un simple disque en possède en fait deux.





### Codes Erreur GEMDOS
(les nombres entre parenthèses sont les numéros d'erreur équivalents MSDOS)

| nom | numéro | description |
|---|---|---|
| EINVFN | -32 (1) | Numéro de fonction invalide |
| EFILNF | -33 (2) | Fichier introuvable |
| EPTHNF | -34 (3) | Chemin introuvable |
| ENHNDL | -35 (4) | Plus d'identificateur disponible |
| EACCDN | -36 (5) | Accès interdit |
| EIHNDL | -37 (6) | Identificateur invalide |
| ENSMEM | -39 (8) | Mémoire insuffisante |
| EIMBA | -40 (9) | Adresse de bloc mémoire invalide |
| EDRIVE | -46 (15) | Spécification de disque invalide |
| ENMFIL | -49 (18) | Plus de fichier |
| ERANGE | -64 | Erreur de champ |
| EINTRN | -65 | Erreur interne GEMDOS |
| EPLFMT | -66 | Format de fichier exécutable invalide |
| EGSBF | -67 | Echec d'accroissement de bloc mémoire |

## 8. LES FONCTIONS GEMDOS PAR NUMERO

| | | |
|---|---|---|
| 0x00 | Pterm0 | Fin de processus |
| 0x01 | Cconin | Lit un caractère sur l'entrée standard |
| 0x02 | Cconout | Écrit un caractère sur la sortie standard |
| 0x03 | Cauxin | Lit un caractère sur l'entrée standard AUX: |
| 0x04 | Cauxout | Écrit un caractère sur la sortie standard AUX: |
| 0x05 | Cprnout | Écrit un caractère sur la sortie standard PRN: |
| 0x06 | Crawio | Lecture/écriture brute sur l'entrée/sortie standard |
| 0x07 | Crawcin | Lecture brute sur l'entrée standard |
| 0x08 | Cnecin | Lit un caractère sur l'entrée standard, sans écho |
| 0x09 | Cconws | Écrit une chaine sur la sortie standard |
| 0x0A | Cconrs | Lit une chaine formatée sur l'entrée standard |
| 0x0B | Cconis | Teste l'état de l'entrée standard |
| 0x0E | Dsetdrv | Fixe le lecteur de disque par défaut |
| 0x10 | Cconos | Teste l'état de la sortie standard |
| 0x11 | Cprnos | Teste l'état du périphérique standard PRN: |
| 0x12 | Cauxis | Teste l'état du standard AUX: en entrée |
| 0x13 | Cauxos | Teste l'état du standard AUX: en sortie |
| 0x19 | Dgetdrv | Demande le disque par défaut |
| 0x1A | Fsetdta | Fixe l'adresse du DTA (Disk Transfer Adress) |
| 0x20 | Super | Entre/Sort/Demande du mode superviseur |
| 0x2A | Tgetdate | Demande la date |
| 0x2B | Tsetdate | Fixe la date |
| 0x2C | Tgettime | Demande l'heure |
| 0x2D | Tsettime | Fixe l'heure |
| 0x2F | Fgetdta | Demande l'adresse du DTA (Disk Transfer Adress) |
| 0x30 | Sversion | Demande le numéro de version du GEMDOS |
| 0x31 | Ptermres | Finit un programme qui reste résident |
| 0x36 | Dfree | Demande d'informations sur un disque |
| 0x39 | Dcreate | Création d'un sous-répertoire |
| 0x3A | Ddelete | Efface un sous-répertoire |
| 0x3B | Dsetpath | Fixe le catalogue courant |
| 0x3C | Fcreate | Création d'un fichier |
| 0x3D | Fopen | Ouvre un fichier |
| 0x3E | Fclose | Ferme un fichier |
| 0x3F | Fread | Lit un fichier |
| 0x40 | Fwrite | Écrit un fichier |
| 0x41 | Fdelete | Efface un fichier |
| 0x42 | Fseek | Positionnement dans un fichier |
| 0x43 | Fattrib | Retourne/fixe les attributs de fichier |
| 0x45 | Fdup | Recopie un identificateur de fichier standard |
| 0x46 | Fforce | Force un identificateur de fichier |
| 0x47 | Dgetpath | Demande le répertoire courant |
| 0x48 | Malloc | Demande d'allocation mémoire |
| 0x49 | Mfree | Libère de la mémoire |
| 0x4A | Mshrink | Rétrécit un bloc de mémoire allouée |
| 0x4B | Pexec | Charge/Exécute un programme |
| 0x4C | Pterm | Termine un programme |
| 0x4E | Fsfirst | Recherche la première occurence d'un fichier |
| 0x4F | Fnext | Recherche l'occurence suivante |
| 0x56 | Frename | Renomme un fichier |
| 0x57 | Fdatime | Demande ou fixe la date de création d'un fichier |





## FONCTIONS GEMDOS PAR NOM

| | | |
|---|---|---|
| 0x03 | Cauxin | Lit un caractère sur l'entrée standard AUX: |
| 0x12 | Cauxis | Teste l'état du standard AUX: en entrée |
| 0x13 | Cauxos | Teste l'état du standard AUX: en sortie |
| 0x04 | Cauxout | Écrit un caractère sur la sortie standard AUX: |
| 0x01 | Cconin | Lit un caractère sur l'entrée standard |
| 0x0B | Cconis | Teste l'état de l'entrée standard |
| 0x10 | Cconos | Teste l'état de la sortie standard |
| 0x02 | Cconout | Écrit un caractère sur la sortie standard |
| 0x0A | Cconrs | Lit une chaine formatée sur l'entrée standard |
| 0x09 | Cconws | Écrit une chaine sur la sortie standard |
| 0x08 | Cnecin | Lit un caractère sur l'entrée standard, sans écho |
| 0x11 | Cprnos | Teste l'état du périphérique standard PRN: |
| 0x05 | Cprnout | Écrit un caractère sur la sortie standard PRN: |
| 0x07 | Crawcin | Lecture brute sur l'entrée standard |
| 0x06 | Crawio | Lecture/écriture brute sur l'entrée/sortie standard |
| 0x39 | Dcreate | Création d'un sous-répertoire |
| 0x3A | Ddelete | Efface un sous-répertoire |
| 0x36 | Dfree | Demande d'informations sur un disque |
| 0x19 | Dgetdrv | Demande le disque par défaut |
| 0x47 | Dgetpath | Demande le répertoire courant |
| 0x0E | Dsetdrv | Fixe le lecteur de disque par défaut |
| 0x3B | Dsetpath | Fixe le catalogue courant |
| 0x43 | Fattrib | Retourne/fixe les attributs de fichier |
| 0x3E | Fclose | Ferme un fichier |
| 0x3C | Fcreate | Création d'un fichier |
| 0x57 | Fdatime | Demande ou fixe la date de création d'un fichier |
| 0x41 | Fdelete | Efface un fichier |
| 0x45 | Fdup | Recopie un identificateur de fichier standard |
| 0x46 | Fforce | Force un identificateur de fichier |
| 0x2F | Fgetdta | Demande l'adresse du DTA (Disk Transfer Adress) |
| 0x3D | Fopen | Ouvre un fichier |
| 0x3F | Fread | Lit un fichier |
| 0x56 | Frename | Renomme un fichier |
| 0x42 | Fseek | Positionnement dans un fichier |
| 0x1A | Fsetdta | Fixe l'adresse du DTA (Disk Transfer Adress) |
| 0x4E | Fsfirst | Recherche la première occurence d'un fichier |
| 0x4F | Fnext | Recherche l'occurence suivante |
| 0x40 | Fwrite | Écrit dans un fichier |
| 0x48 | Malloc | Demande d'allocation mémoire |
| 0x49 | Mfree | Libère de la mémoire |
| 0x4A | Mshrink | Rétrécit un bloc de mémoire allouée |
| 0x4B | Pexec | Charge/Exécute un programme |
| 0x4C | Pterm | Termine un programme |
| 0x00 | Pterm0 | Termine un progamme (code de retour 0) |
| 0x31 | Ptermres | Termine un programme qui reste résident |
| 0x20 | Super | Entre/Sort/Demande du mode superviseur |
| 0x30 | Sversion | Demande le numéro de version du GEMDOS |
| 0x2A | Tgetdate | Demande la date |
| 0x2C | Tgettime | Demande l'heure |
| 0x2B | Tsetdate | Fixe la date |
| 0x2D | Tsettime | Fixe l'heure |

## 0x00 Pterm0 Finit un programmme

```
void Pterm0()
```

Termine un processus, fermant tous les fichiers qu'il a ouvert et libérant la mémoire qu'il a allouée. Retourne 0x0000 comme code de sortie au programme parent.

## 0x01 Cconin Lit un carctère sur l'entrée standard

```
LONG Cconin()
```

Lit un caractère sur l'entrée standard (identificateur 0). Si l'entrée standard est la console, le long_mot retourné dans D0 contient le code ASCII et le code clavier de la touche appuyée (code de scrutation) :

| 31..24 | 23.. 16 | 15..8 | 7..0 |
|---|---|---|---|
| 0x00 ou bits de shift | code de scrutation ou 0x00 | 0x00 | caractère ASCII |

Les touches de fonction (F1 à F10, HELP, UNDO, etc...) retournent le code ASCII 0x00, avec le code clavier approprié (cf. le manuel sur le clavier intelligent pour la valeur des codes clavier). Le ST BIOS place l'état des touches spéciales dans les bits 24 à 31 (voir le Guide du programmeur du BIOS pour plus de détails).

ERREURS

Ne retourne aucune indication de fin de fichier.
Ne reconnaît pas 'Control_C'
Impossible de savoir si l'entrée est un périphérique à caractère ou un fichier.
Il devrait exister un moyen de taper les 256 codes possibles à partir du clavier.

## 0x02 Cconout Ecrit un caractère sur la sortie standard

```
void Cconout(c)
WORD c;
```

Ecrit le caractère 'c' sur la sortie standard (identificateur 0). Les huits bits les plus significatifs de 'c' sont réservés et doivent être à zéro. Les tabulations ne sont pas interprétées.





## 0x03 Cauxin Lit un caractère sur l'entrée standard AUX:

```
WORD Cauxin()
```

Lit un caractère à partir de l'identificateur 1 (le port série AUX:, normalement).

REMARQUE

Le contrôle de flux de la sortie RS232 ne marche pas avec cette fonction. Les programmes devraient utiliser l'appel de périphérique de caractère BIOS afin d'éviter de perdre des caractères reçus.

## 0x04 Cauxout Ecrit un caractère sur la sortie standard AUX:

```
void Cauxout(c)
WORD c;
```

Ecrit 'c' sur l'identificateur 1 (le port série AUX:, normalement). Les huits bits hauts de 'c' sont réservés et doivent être à zéro. Les tabulations ne sont pas interprétées.

REMARQUE

Le contrôle de flux RS232 ne marche pas avec cette fonction. Les programmes devraient employer l'appel de périphérique à caractère BIOS afin d'éviter de perdre des caractères envoyés.

## 0x05 Cprnout Ecrit un caractère sur la sortie standard PRN:

```
void Cprnout(c)
WORD c;
```

Ecrit 'c' sur l'identificateur 2 (le port imprimante PRN:, normalement). Les huits bits hauts de 'c' sont réservés et doivent être à zéro. Les tabulations ne sont pas interprétées.

## 0x06 Crawio  Lecture/Ecriture brute sur l'entrée/sortie standard

```
LONG Crawio(w)
WORD w;
```

Si 'w' n'est pas 0x00FF, 'w' est écrit sur la sortie standard. Autrement si 'w' est 0x00FF, un caractère est lu sur l'entrée standard. Si aucun caractère n'est disponible, 0x0000 est retourné. Les tabulations ne sont pas interprétées.

REMARQUES

Par le fait même de sa conception cette fonction ne peut écrire '0xff' sur la sortie standard.
0x00 ne peut être différencié d'une fin de fichier.

## 0x07 Crawcin  Entrée brute sur l'entrée standard

```
LONG Crawcin()
```

Lit un caractère sur l'entrée standard (identificateur 0). Si le périphérique d'entrée est 'CON:' aucun contrôle de caractère n'est fait et il n'y a pas d'écho.

REMARQUE

Pas d'indication de fin de fichier.

## 0x08 Cnecin  Lit un caractère sur l'entrée standard, sans écho

```
LONG Cnecin()
```

Lit un caractère sur l'entrée standard. Si le périphérique d'entrée est 'CON:' aucun n'écho n'est fait. Les caractères de contrôle sont interprétrés.

## 0x09 Cconws  Ecrit une chaîne sur la sortie standard

```
void Cconws(chaine)
char *chaine;
```

Ecrit une chaine débutant à 'chaine' et finissant par 0, sur la sortie standard.





## 0x0A Cconrs  Lit une chaine sur l'entrée standard

```
void Cconrs(buf)
char *buf;
```

Lit une chaîne sur l'entrée standard. Les caractères de contrôle habituels sont interprétés :

| Caractère | Fonction |
|---|---|
| <return>, ^J | Fin de ligne |
| ^H, <rub> | Efface le dernier caractère |
| ^U, ^X | Efface la ligne entière |
| ^R | Recopie la ligne |
| ^C | Fini le programme |

Le premier caractère de 'buf' contient le nombre maximum d'octets à lire (taille du tampon moins deux). En sortie, le deuxième octet contient le nombre de caractères lus. La chaîne se trouve entre 'buf+2' et 'buf+2+buf[1]'.
Il n'est pas garanti que la chaîne se finisse par 0.

REMARQUE

Plante sur les fins de fichiers.

## 0x0B Cconis  Teste l'état de l'entrée standard

```
WORD Cconis()
```

Retourne 0xFFFF si un caractère est disponible sur l'entrée standard, 0x0000 sinon.

## 0x0E Dsetdrv  Fixe le lecteur de disque par défaut

```
LONG Dsetdrv(drv)
WORD drv;
```

Fixe le disque 'drv' par défaut. Les disques sont rangés de 0 à 15 (A: à P:). Retourne dans D0.L la carte des disques actifs (bit 0 = A, bit 1 = B, etc..).

Un 'disque actif' est un disque à partir duquel un catalogue a été fait.

REMARQUE

Le GEMDOS ne supporte que 16 disques (bits 0 à 15). Les systèmes ultérieurs en supporteront 32 .

## 0x10 Cconos  Teste l'état de la sortie standard

```
WORD Cconos()
```

Renvoie 0xFFFF si la console est prête à recevoir un caractère, 0x0000 si la console n'est PAS prête.

## 0x11 Cprnos  Teste l'état de la sortie standard PRN:

```
WORD Cprnos()
```

Retourne 0xFFFF si 'PRN:' est prêt à recevoir un caractère, 0x0000 sinon.

## 0x12 Cauxis  Teste l'état de l'entrée standard AUX:

```
WORD Cauxis()
```

Retourne 0xFFFF si un caractère est disponible sur l'entrée 'AUX:' (identificateur 1), 0x0000 sinon.

## 0x13 Cauxos  Teste l'état de la sortie standard AUX:

```
WORD Cauxos()
```

Renvoie 0xFFFF si 'AUX:' (identificateur 1) est prêt à recevoir un caractère, 0x0000 sinon.

## 0x19 Dgetdrv  Recherche le lecteur de disque par défaut

```
WORD Dgetdrv()
```

Retourne le numéro du lecteur courant, compris entre 0 et 15.





## 0x1A Fsetdta  Fixe l'adresse du DTA (Disk Transfer Adress)

```
void Fsetdta(adr)
char adr;
```

Fixe l'adresse du DTA à 'adr'. Le tampon de stockage des données sur un fichier (DTA) ne sert que pour les fonctions Fsfirst() et Fsnext().

## 0x20 Super  Change/teste le mode (Utilisateur ou Superviseur)

```
LONG Super(pile)
WORD *pile;
```

'pile' est égal à -1L (0xFFFFFFFF):

la fonction retourne 0x0000 si le processeur est en mode utilisateur, 0x0001 s'il est en mode superviseur.

'pile' est différent de -1L:

si le processeur est en mode utilisateur, la fonction revient avec le processeur en mode superviseur. Si 'pile' est NULL (0x00000000), la pile superviseur sera la même que la pile utilisateur avant l'appel. Sinon la pile superviseur sera placée à 'pile'.

Si le processeur était en mode superviseur, il se retrouve en mode utilisateur au retour de la fonction. 'pile' devra être la valeur de la pile utilisateur qui a été retournée par le premier appel de la fonction.

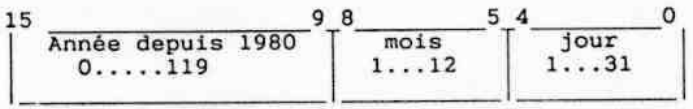
ATTENTION

La pile superviseur originale doit être replacée avant la fin du programme sinon le système plantera à la sortie du programme.

0x2A Tgetdate  Demande la date

```
WORD Tgetdate()
```

Retourne la date en format DOS :




RETOUR

les bits 0 à 4 contiennent le jour allant de 1 à 31.
les bits 5 à 8 contiennent le mois allant de 1 à 12.
les bits 9 à 15 contiennent l'année (à partir de 1980) allant de 0 à 119.

0x2B Tsetdate  Fixe la date

```
WORD Tsetdate(date)
WORD date;
```

Fixe 'date' comme date courante dans le format décrit dans Tgetdate().

RETOUR

0 si la date est valide.
ERROR si le format de la date est incorrect.

REMARQUES

Le GEMDOS n'est pas difficile sur la date; par exemple le 31 février ne lui déplait pas.
Le GEMDOS ne prévient PAS le BIOS que la date a changée.

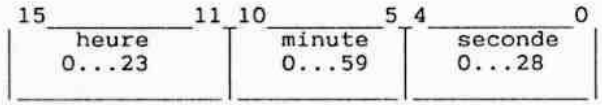




0x2C Tgettime  Demande l'heure

```
WORD Tgettime()
```

Retourne l'heure courante dans le format DOS:




RETOUR

Les bits :
0 à 4 contiennent les secondes divisées par 2 (0 à 28)
5 à 10 contiennent les minutes (0 à 59)
11 à 15 contiennent les heures (0 à 23)

0x2D Tsettime  Fixe l'heure

```
WORD Tsettime(heure)
WORD heure;
```

Fixe 'heure' comme heure courante dans le format décrit dans Tgettime().

RETOUR

0 si le format de l'heure fournie est valide;
ERROR si le format de l'heure n'est pas valide.

REMARQUE

Le GEMDOS ne prévient pas le BIOS que l'heure a changé.

0x2F Fgetdta  Demande l'adresse du DTA (Disk Transfer Address)

```
LONG Fgetdta()
```

Retourne l'adresse courante du tampon de stockage des données sur un fichier (DTA), employée par les fonctions Fsfirst() et Fsnext().

## 0x30 Sversion  Demande le numéro de version du GEMDOS

```
WORD Sversion( )
```

Retourne le numéro de version du GEMDOS en format inversé. L'octet le plus significatif contient la partie basse du numéro, l'octet le moins significatif, la partie haute.

REMARQUES

La version du GEMDOS sur disquette du 29/5/85 et la première version en ROM du 20/11/85 ont 0x1300 comme numéro.
Les numéros de version du GEMDOS et du TOS ne sont PAS les mêmes. (cf. LE MANUEL DE REFERENCE DU BIOS DU ST pour le numéro de version du TOS).

## 0x31 Ptermres  Finit un programme qui reste résident

```
void Permres( nbrest,coderet)
LONG nbrest;
WORD coderet;
```

Finit le programme courant, en conservant une part de sa mémoire.

'nbrest' le est nombre d'octets appartenant au programme qui doivent être gardés, comprenant et commençant à la page de base.

'coderet' est le code de sortie qui est retourné au programme parent.

La mémoire que le programme a allouée (en plus de sa zone TPA) N'EST PAS libérée.

Il est impossible de rappeler le programme terminé par Ptermres( ).

REMARQUE

Les fichiers ouverts sont fermés lors de la fin de processus.





## 0x36 Dfree  Demande l'espace libre sur un disque

```
void Dfree(buf,nbdisq)
LONG *buf;
WORD nbdisq;
```

Demande des informations sur le disque 'nbdisq' et les place dans quatre longs_mots commençant à 'buf':

| | |
|---|---|
| buf + 0 | nombre de blocs libres sur le disque |
| buf + 4 | nombre total de blocs sur le disque |
| buf + 8 | taille d'un secteur en octets |
| buf + 12 | nombre de secteurs par bloc |

REMARQUE

Incroyablement lent sur un disque dur (5 à 10 secondes).

## 0x39 Dcreate  Création d'un sous répertoire de disque

```
WORD Dcreate(chemin)
char *chemin;
```

'Chemin' pointe sur une chaîne terminée par 0 spécifiant le chemin du nouveau répertoire.

RETOUR

0 en cas de succès;
ERROR ou le numéro d'erreur approprié en cas d'échec.

## 0x3A Ddelete  Efface un sous répertoire

```
WORD Ddelete(chemin)
char *chemin;
```

Efface un répertoire qui doit être vide (il peut toutefois comporter les fichiers spéciaux "." et ".."). 'chemin' pointe sur une chaîne terminée par zéro, spécifiant le chemin du répertoire à effacer.

RETOUR

0 en cas de succès;
ERROR ou le numéro d'erreur approprié en cas d'échec.

## 0x3B Dsetpath  Fixe le répertoire courant

```
WORD Dsetpath(chemin)
char *chemin;
```

Fixe 'chemin', (une chaîne terminée par 0), comme répertoire courant. Si le chemin commence par une lettre identificatrice de disque suivie de deux-points, le répertoire courant est placé sur le lecteur spécifié.

Le système conserve un répertoire courant pour chaque lecteur.

RETOUR

0 en cas de succès;
ERROR ou le numéro d'erreur approprié en cas d'echec.

## 0x3C Fcreate  Création d'un fichier

```
WORD Fcreate(nomfich,attribs)
char *nomfich;
WORD attribs;
```

Crée un fichier 'nomfich' et retourne son identificateur, non standard, d'écriture seule. Le mot d'attributs est stocké dans le répertoire; les bits d'attributs sont :

| masque | description |
|---|---|
| 0x01 | fichier créé en lecture seule |
| 0x02 | fichier transparent au répertoire |
| 0x04 | fichier système |
| 0x08 | fichier contenant un nom de volume sur 11 bits |

RETOUR

Un nombre positif, l'identificateur ou :
ERROR ou le numéro d'erreur approprié.

REMARQUE

Si le bit de lecture seule est positionné, un identificateur d'écriture seule est retourné, et de plus cet identificateur ne permet pas d'écrire.

Théoriquement un seul numéro de volume est permis sur un répertoire racine. Le GEMDOS n'impose rien de tel, ce qui peut causer quelque confusion.





## 0x3D Fopen  Ouverture d'un fichier

```
WORD Fopen(nmfich,mode)
char *nmfich;
WORD mode;
```

Ouvre le fichier 'nmfich' en mode 'mode' et retourne son identificateur non standard. Le mode d'ouverture peut être:

| mode | description |
|---|---|
| 0 | lecture seule |
| 1 | écriture seule |
| 2 | lecture et écriture |

RETOUR

Un nombre positif, l'identificateur
Un numéro d'erreur négatif.

## 0x3E Fclose  Fermeture d'un fichier

```
WORD Fclose(idfich)
WORD idfich;
```

Ferme le fichier dont l'identificateur est 'idfich'.

RETOUR

0 en cas de succès;
ERROR ou un numéro d'erreur approprié en cas d'échec.

## 0x3F Fread  Lecture sur un fichier

```
LONG Fread(idfich,nbre,buffer)
WORD idfich;
LONG nbre;
char *buffer;
```

Lit 'nbre' octets dans le fichier d'identificateur 'idfich', et les place en mémoire à partir de l'adresse 'buffer'.

RETOUR

Le nombre d'octets réellement lus, ou:
0 si code de fin de fichier
Un numéro d'erreur négatif.

## 0x40 Fwrite  Ecriture sur un fichier

```
LONG Fwrite(idfich,nbre,buffer)
WORD idfich;
LONG nbre;
char *buffer;
```

Ecrit 'nbre' octets de la mémoire à partir de l'adresse 'buffer', dans le fichier ayant comme identificateur 'idfich'.

RETOUR

Le nombre d'octets réellement écrits en cas de succès.
Un numéro d'erreur négatif autrement.

## 0x41 Fdelete  Effacement d'un fichier

```
WORD Fdelete(nomfich)
char *nomfich;
```

Efface le fichier ayant comme nom 'nomfich'.

RETOUR

0 en cas de succès
Un numéro d'erreur négatif autrement.

## 0x42 Fseek  Positionne le pointeur de fichier

```
LONG Fseek(offset,idfich,mode)
LONG offset;
WORD idfich;
WORD mode;
```

Positionne le pointeur dans le fichier défini par l'identificateur 'idfich'. 'offset' est un nombre signé; une valeur positive déplace le pointeur vers la fin du fichier, une valeur négative, vers le début. Le 'mode' de positionnement peut être :

| mode | Déplacement d'offset octets... |
|---|---|
| 0 | à partir du début du fichier |
| 1 | à partir de la position courante |
| 2 | à partir de la fin du fichier |

RETOUR

La position courante dans le fichier.
Un numéro d'erreur négatif si erreur.





## 0x43 Fattrib  Fixe ou demande les attributs d'un fichier

```
WORD Fattrib(nmfich,drap,attribs)
char *nmfich;
WORD drap;
WORD attribs;
```

'nmfich' pointe sur un chemin terminé par 0. Si 'drap' a la valeur 1, les attributs de fichier 'attribs' sont fixés (pas de valeur de retour). Si 'drap' est 0, ils sont retournés.

Les bits d'attributs sont :

| masque | description |
|---|---|
| 0x01 | fichier à lecture seule |
| 0x02 | fichier transparent au répertoire |
| 0x04 | fichier 'système' |
| 0x08 | fichier contenant un nom de volume (11 octets) |
| 0x10 | fichier sous-répertoire |
| 0x20 | fichier écrit puis refermé |

REMARQUE

Le bit d'archivage, 0x20, ne semble pas marcher comme prévu.

## 0x45 Fdup  Duplique un identificateur de fichier

```
WORD Fdup(idfich)
WORD idfich;
```

L'identificateur 'idfich' doit être un identificateur standard (0 à 5). Fdup() retourne un identificateur non standard (supérieur ou égal à 6) qui pointe le même fichier.

RETOUR

Un identificateur ou :
EIHNDL  'idfich' n'est pas un identificateur standard
ENHNDL  Plus d'identificateur non standard

## 0x46 Fforce  Force un indentificateur de fichier

```
Fforce(stdh,nonstdh)
WORD stdh;
WORD nonstdh;
```

Force l'identificateur standard 'stdh' à pointer le même fichier ou périphérique que l'identificateur non-standard 'nonstdh'.

RETOUR

```
0 si OK
EIHNDL    identificateur invalide.
```

## 0x47 Dgetpath  Demande le répertoire courant

```
void Dgetpath(buf,driveno)
char *buf;
WORD driveno;
```

Le répertoire courant pour le lecteur 'driveno' est recopié dans 'buf'. Le numéro de lecteur commence à 1 pour le lecteur A:, 2 pour le B:, etc..., 0 spécifiant le disque par défaut.

REMARQUE

La taille maximum d'un chemin n'est pas limitée par le système. C'est à l'application de fournir assez de place pour le tampon. 128 octets semblent suffisantS pour 8 ou 9 sous-répertoires.

## 0x48 Malloc  Demande d'allocation mémoire

```
LONG Malloc(taille)
LONG taille;
```

Si 'taille' est -1L (0xFFFFFFFF) la fonction retourne la taille du plus grand bloc libre du système. Autrement si 'taille' est différent de -1L, la fonction essaie d'allouer 'taille' octets pour le programme en cours. La fonction retourne un pointeur sur le début du bloc alloué si tout s'est bien passé, ou NULL s'il n'existait pas de bloc assez grand pour satisfaire la requête.

REMARQUE

Un programme ne peut avoir, à un instant donné, plus de 20 blocs alloués par 'Malloc()'. Dépasser cette limite peut désemparer le GEMDOS. (Il est cependant possible de faire le nombre de 'Malloc()' que l'on veut à condition de les faire suivre par l'appel de la fonction Mfree(), 20 étant le nombre maximum de fragments qu'un programme peut générer).





## 0x49 Mfree  Libération de mémoire

```
WORD Mfree(adbloc)
LONG adbloc;
```

Libère le bloc mémoire commençant à 'adbloc'. Le bloc doit être un de ceux alloués par Malloc().

RETOUR

```
0 si la libération s'est bien Effectuée.
ERROR ou le numéro d'erreur approprié sinon.
```

## 0x4A Mshrink  Rétrécit la taille d'un bloc alloué

```
WORD Mshrink(0,bloc,nouvtail)
(WORD) 0;
LONG bloc;
LONG nouvtail;
```

Rétrécit la taille d'un bloc mémoire alloué. 'bloc' pointe sur la page de base d'un programme ou sur un bloc de mémoire alloué par Malloc(), 'nouvtail' est la nouvelle taille du bloc.

Le premier argument du bloc doit être un mot nul.

RETOU..

```
0 si l'ajustement de taille à été réussi.
EIMBA     si l'adresse du bloc mémoire était invalide.
EGSBF     si la nouvelle taille demandée était Erronée.
```

REMARQUE

un bloc ne peut être que rétréci; la nouvelle taille du bloc doit forcément être inférieure à la précédente.

N.D.T.

Le compilateur 'C' Alcyon rajoute lors de la compilation le premier paramètre d'appel '0'. Il ne faut donc pas le mettre dans l'appel de la fonction si l'on se sert de ce compilateur.

Appel de la fonction avec le 'C' Alcyon :

```
Mshrink(bloc,nouvtail);
```

## 0x4B Pexec  Charge/Exécute un programme

```
WORD Pexec(mode,ptr1,ptr2,ptr3)
WORD mode;
char *ptr1;
char *ptr2;
char *ptr3;
```

Cette fonction permet différentes utilisations selon le drapeau 'mode':

| mode | ptr1 | ptr2 | ptr3 |
|---|---|---|---|
| 0: charge et exécute | le fichier à exécuter | le jeu de commandes | la chaîne d'environnement |
| 3: charge sans lancer | le fichier à charger | le jeu de commandes | la chaîne d'environnement |
| 4: exécute uniquement | l'adr. de la page de base | (inutilisé) | (inutilisé) |
| 5: crée une page de base | (inutilisé) | le jeu de commandes | la chaîne d'environnement |

Le nom du fichier à charger ou à exécuter, 'ptr1', et la chaîne du jeu de commandes, 'ptr2', sont des chemins, terminés par 0. La chaîne d'environnement 'ptr3', est soit NULL (0L), soit un pointeur sur une structure de chaîne de la forme:

```
"chaine1\0"
"chaine2\0"
  etc...
"chaine3\0"
"\0"
```

La chaîne d'environnement peut être n'importe quel numéro de chaine finie par un 0, suivie par une chaîne nulle (un simple 0). Le programme hérite d'une copie de la chaîne d'environnement parente si 'ptr3' est 'NULL'.

Le mode 0 (charge et exécute) chargera le fichier spécifié, composera sa page de base, et l'exécutera. La valeur retournée par Pexec() sera le code de sortie du processus enfant (voir Pterm0() et Pterm()).

Le mode 3 (charge sans exécuter) chargera le fichier spécifié, composera la page de base, et retournera un pointeur sur cette page de base. Le programme n'est pas exécuté.





Le mode 4 (exécute seulement) reçoit un pointeur sur une page de base. Le programme commence son exécution au début de la zone de texte, comme spécifié dans la page de base.

Le mode 5 (création d'une page de base) allouera le plus grand bloc libre de mémoire et créera la plus grande partie de sa page de base. Quelques entrées comme la taille des zones de texte/données initialisées/données non initialisées et leur adresse de base NE SONT PAS installées. Le programme appelant en a la charge.

Un programme enfant hérite des descripteurs de fichiers standards de son parent. Il emploie en fait un appel de Fdup() et de Fforce() sur les identificateurs 0 à 5.

Puisque les ressources système sont alloués lors de la création de la page de base, le processus engendré DOIT se terminer en les libérant. Ceci est particulièrement important lors de l'emploi d'overlays. (voir 'le livre de cuisine de Pexec' pour plus de détails sur Pexec()).

## 0x4C Pterm  Termine un programme

```
void Pterm(retcode)
WORD retcode;
```

Termine le programme courant, ferme tous les fichiers ouverts et libère la mémoire allouée. Retourne 'retcode' au processus parent.

## 0x4E Fsfirst  Recherche la première occurence d'un fichier

```
WORD Fsfirst(fspec,attribs)
char *fspec;
WORD attribs;
```

Recherche la première occurence du fichier 'fspec'. Le spécificateur de fichier peut contenir des caractères spéciaux ('?' ou '*') dans le nom de fichier mais pas dans le chemin de spécification. 'attribs' contrôle le type de fichier qui sera retourné par Fsfirst(). Son format a été décrit dans la documentation sur 'Fattrib()'.

Si 'attribs' est à zéro, les fichiers normaux seront seuls recherchés (aucun nom de volume, fichier caché, sous-répertoire ou fichier système ne sera retourné). Si 'attribs' est positionné sur les fichiers cachés ou sur les fichiers systèmes, alors ceux_ci seront pris en compte pour la recherche. Si 'attribs' est positionné pour trouver un nom de volume, alors seuls les noms de volume seront recherchés.

Lorsqu'un fichier est trouvé, une structure de 44 octets est écrite à l'emplacement pointé par le DTA.

| offset | taille | contenus |
|---|---|---|
| 0 à 20 | | (réservés) |
| 21 | octet | bits d'attributs |
| 22 | mot | heure de création |
| 24 | mot | date de création |
| 26 | long | taille du fichier |
| 30 | 14 octets | nom et extension du fichier |

Le nom de fichier et son extension se terminent par 0, et ne contiennent pas d'espaces.

RETOUR

0 si le fichier a été trouvé
EFILNF  si le fichier n'a pas été trouvé, ou le numéro d'erreur approprié.





## 0x4F Fsnext  Recherche des occurences suivantes

```
WORD Fsnext()
```

Recherche les occurences suivantes d'un fichier. La première occurence devra être recherchée par Fsfirst(). Les octets de 0 à 20 doivent rester inchangés depuis l'appel de Fsfirst() ou depuis le dernier appel de Fsnext().

RETOUR

0 si le fichier a été trouvé
ENMFIL  s'il n'y a plus de fichiers trouvés, ou le numéro d'erreur approprié.

## 0x56 Frename  Renomme un fichier

```
WORD Frename(0,ancnom,nouvnom)
(WORD) 0;
char *ancnom;
char *nouvnom;
```

Renomme un fichier 'ancnom' en 'nouvnom'. Le nouveau nom ne doit pas déjà exister mais peut être dans un autre répertoire.

Le premier paramètre doit être 0 (mot).

RETOUR

0 si OK
EACCDN  si le nouveau nom existe déjà
EPTHNF  si l'ancien nom n'a pas été trouvé
ENSAME  si l'identificateur de disque (A,B,..) n'est pas le même pour les deux noms.

## 0x57 Fdatime  Fixe ou demande le moment de création d'un fichier

```
void Fdatime(idfich,pttemp,drap)
WORD idfich;
LONG pttemp;
WORD drap;
```

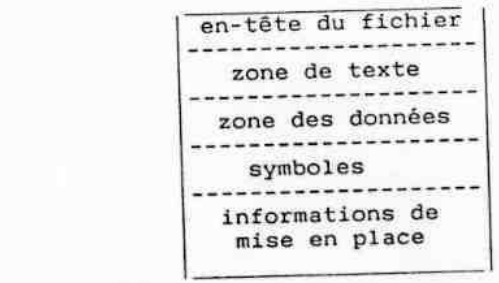
Le fichier est reconnu par son identificateur 'idfich'. 'pttemp' pointe sur deux mots contenant l'heure et la date en format DOS (l'heure se trouve dans le premier mot, la date dans le second).

Si 'drap' est à 1, la date et l'heure sont placées dans le fichier à partir de 'pttemp'.
Si 'drap' est à 0, la date et l'heure sont lues et placées dans 'pttemp'.

## 9. FORMAT D'UN FICHIER EXECUTABLE

Un fichier exécutable se compose d'un en-tête suivi par une image du texte et des données, d'une table des symboles, pouvant être vide, d'une table de décalage pour la relogeabilité et de zéro ou plusieurs enregistrements.

Parties d'un fichier exécutable




L'en-tête de fichier contient un 'nombre magique' (une signature indiquant que le fichier est exécutable) et plusieurs longs-mots contenant des informations de taille :

En-tête de fichier exécutable

| offset | taille | description |
|---|---|---|
| 0x00 | mot | 0x601A (nombre magique) |
| 0x02 | long | taille de la zone texte |
| 0x06 | long | taille des données initialisées |
| 0x0A | long | taille des données non initialisées |
| 0x0E | long | taille de la table des symboles |
| 0x12 | long | (réservé) |
| 0x16 | long | (réservé) |
| 0x1A | long | (réservé) |
| 0x1E | | début de la zone de texte |

La zone du texte et des données suit immédiatement l'en-tête . La table des symboles, s'il en existe une, suit la zone des données.





Le GEMDOS se composera un long-mot dans la zone de texte ou de données en ajoutant l'adresse de base de la zone de texte à la valeur se trouvant déjà dans le long_mot. La liste 'de mise en place' spécifie les long-mots à reloger. Le premier élément de cette liste est un long-mot spécifiant l'offset du premier long_mot à reloger. Si ce long_mot est NULL (0L), il n'y a pas de longs_mots à reloger. Les octets qui suivent le long_mot déterminent l'offset du long_mot suivant à reloger.

Les longs_mots doivent débuter sur une adresse paire, sinon le système plantera.

Octets de relogement

| Octets | Description |
|---|---|
| 0 | fin des informations de relogement |
| 1 | 254 octets, à ajouter à l'octet suivant |
| 2,4,..254 | pointeur sur le long_mot à reloger |
| 3,5,..255 | nombres impairs, pour usage ultérieur |

**TABLE DES SYMBOLES**

La table des symboles se compose d'entrées formatées ainsi:

Entrées dans la table des symboles

| nom du symbole<br>8 octets |
|---|
| type de symbole<br>1 mot |
| valeur du symbole<br>1 long-mot |

Valeur des types de symbole

| Type | valeur |
|---|---|
| défini | 0x8000 |
| égal | 0x4000 |
| global | 0x2000 |
| registre égal | 0x1000 |
| référence externe | 0x0800 |
| donnée relogeable | 0x0400 |
| code relogeable | 0x0200 |
| BSS relogeable | 0x0100 |





## 10. STRUCTURE DES DISQUES

Le GEMDOS se sert des premiers secteurs d'un disque pour indiquer les emplacements de stockage des fichiers. Un volume se décompose habituellement en cinq parties : un secteur de départ, appelé 'boot', deux tables d'allocation (FAT) identiques, un répertoire racine, et une zone de blocs contenant des données ou des catalogues fils.

Lorsque le GEMDOS accède pour la première fois à un disque, ou y accède après un changement de média ( média change), il fait un appel à la fonction BIOS 'GETBPB' (Get BIOS Parameter Block) afin de déterminer la taille de ses différentes zones, et leurs emplacements sur le disque. GETBPB renvoie un pointeur sur une structure de neufs mots. Le GEMDOS peut déduire de cette structure les emplacements des différentes parties du fichier système.

Bloc de Paramère BIOS (BPB)

| nom | valeur | fonction |
|---|---|---|
| recsiz | 512 | nombre d'octets d'un secteur physique |
| clsiz | 2 | nombre de secteurs par bloc |
| clsizb | 1024 | nombre d'octets par bloc |
| rdlen | | nbre. de secteurs du catalogue racine |
| fsiz | | nombre de secteurs de FAT |
| fatrec | | numéro du 1er secteur de la 2ème FAT |
| datrec | | numéro du 1er secteur de données |
| numcl | | nombre de blocs de données |
| bflags | | drapeaux |

RECSIZ indique le nombre d'octets par secteurs physique: 512 pour le GEMDOS actuel. CLSIZ indique le nombre de secteurs par bloc: 2 actuellement. CLSIZB est le nombre d'octets par bloc, qui doit être de 1024.

RDLEN est le nombre de secteurs réservés au catalogue racine. Une entrée dans le catalogue se faisant sur 32 octets, le nombre de fichiers que l'on peut rentrer dans un catalogue racine est de RDLEN * 512 / 32.

FSIZ est le nombre de secteurs pour chaque 'FAT' (table d'allocation). FATREC est le numéro du premier secteur de la deuxième FAT.

DATREC est le numéro du premier secteur de données. NUMCL représente le nombre de secteurs de données sur le disque.

BFLAGS était supposé être un vecteur de bits de drapeau. En fait, actuellement, seul le bit 0 est employé; Son positionnement signifie que l'on emploie 16 bits pour chaque entrée sur les FAT (disque dur),au lieu de 12 (disquette).

S'il existe des secteurs 'boots', ils occupent les secteurs logiques 0 à FATREC - FSIZ - 1. La deuxième 'FAT' débute à FATREC, et la première débute à FATREC - FSIZ. Le catalogue racine commence à FATREC + FSIZ, et le premier secteur de données commence à DATREC. La zone où sont stockées les données de tous les fichiers du disque s'appelle le zone de blocs.

**ENTREES DANS LE CATALOGUE**

Chaque entrée dans le catalogue contient le nom de fichier, ses attributs, la date et l'heure de création du fichier, sa taille , et le numéro du premier bloc du fichier. L'entrée elle-même est une structure de 32 octets constituée ainsi :

Entrée du catalogue

| nom sur 8 caractères |
|---|
| extension sur 3 caractères |
| octet d'attributs |
| (10 octets inemployés) |
| heure de création (un mot) |
| date de création (un mot) |
| n° du prémier bloc (un mot) |
| taille du fichier (long-mot) |

Tous les mots et les longs-mots dans les entrées du catalogue sont au format inversé 8086.

Lors de l'effacement d'un fichier le premier octet de son nom est mis à 0xE5.

Un sous-répertoire est un fichier qui contient des entrées de catalogue. Les deux premières entrées dans un sous_répertoire sont toujours les répertoires spéciaux "." et "..".





**ENTREES DANS LES FATS**

La table d'allocation des fichiers sert à l'allocation des blocs pour un fichier et à leur lien entre eux. Les entrées dans une FAT peuvent se faire sur 12 ou 16 bits. Une entrée dans un catalogue contient le numéro du premier bloc alloué à ce fichier. Chaque entrée de 'FAT' associée à un bloc contient le numéro du bloc suivant ou un nombre qui indique une fin de fichier.

Entrées de FAT sur 12 bits

| valeur | interprétation |
|---|---|
| 0x000 | bloc libre |
| 0x001 | (impossible) |
| 0x002 - 0xfef | numéro du prochain bloc |
| 0xff0 - 0xff7 | secteur défectueux |
| 0xff8 - 0xfff | fin de fichier |

Entrées de FAT sur 16 bits

| valeur | interprétation |
|---|---|
| 0x0000 | bloc libre |
| 0x0001 | (impossible) |
| 0x0002 - 0x7fff | numéro du prochain bloc |
| 0x8000 - 0xffef | (impossible) |
| 0xfff0 - 0xfff7 | secteur défectueux |
| 0xfff8 - 0xffff | fin de fichier |

En cas de FAT sur 12 bits, pour obtenir le bloc suivant d'un fichier, BLS, à partir du bloc courant, BL :

1. Multiplier BL par 1.5
   BLS = BL + (BL / 2)
2. Prendre pour BLS le mot de 16 bits pointé par BLS.Le retourner pour l'obtenir au format 68000.
   Le mot peut ne pas être aligné sur une adresse paire.
3. Extraire les 12 bits significatifs:
   Si BL est impair , BLS = BLS >> 4.
4. Masquer les bits incorrects.
   BLS = BLS & 0x0FFF.
5. Interpréter le résultat:
   Si BLS est plus grand ou égal à 0x0FF8, alors BL était le dernier bloc du fichier. Si BLS est égal à zéro ou compris entre 0x0FF0 et 0x0FF7, il y a un problème système. Autrement BLS est le numéro du bloc suivant du fichier.

Dans une FAT sur 16 bits, pour obtenir le bloc suivant dans un fichier, BLS, à partir du bloc courant, BL :

1. Prendre pour BLS le mot de 16 bits pointé par BL. Retourner le mot dans le format 68000.
2. Si BLS est égal à 0xFFF8 ou plus grand, Bl était le dernier bloc du fichier; Si BLS est égal à zéro ou compris entre 0x8000 et 0xFFF7, il y a un problème système de fichier. Autrement, BLS est le numéro du bloc suivant dans le fichier.

Pour passer d'un numéro de bloc, BL, à un numéro de secteur logique, LSN :

1. Ajuster à cause des entrées réservées pour la FAT: LSN = BL -2.
2. Multiplier LSN par le nombre de secteurs par bloc (BLSIZ).
3. Ajouter à LSN le numéro de secteur logique du premier bloc (DATREC).





**LE MANUEL DE REFERENCE**

**DU**

**BIOS ATARI**

## INTRODUCTION

Pas de panique ! Voici la nouvelle version testée du 'Manuel de Référence du BIOS Atari', où est décrit le BIOS, et différents autres aspects des ordinateurs Atari de la série ST. L'introduction ne vous dit peut-être pas encore grand chose, mais elle vous avertit au moins de ne pas paniquer.

Le public visé par ce guide comprend :

Les auteurs d'applications (qui y trouveront la plupart des fonctions, et des suggestions inestimables).

Ceux qui désirent faire usage de quelques éléments de configuration typiques du ST (trafiquer la palette de couleurs, configurer le port RS232, etc..).

Les habituels curieux ( comprenant les drogués d'informations et de documentations, les chercheurs de la petite bête..).

Pour différentes raisons ce manuel devrait encore être considéré comme un document préliminaire. Beaucoup de choses restent non documentées, beaucoup de points du GEMDOS n'ont pas été approchés par nos amis de Digital Research, et il y aurait un bon paquet de possibilités que nous aurions aimé rajouter au software.

Périodiquement, à mesure que nos reporters découvriront de nouveaux moyens de se simplifier la vie, nous ferons une mise à jour du 'Manuel de référence' pour refléter ces soudains et violents changements de la réalité. Vous pouvez, à l'occasion appeler Atari pour voir si une mise à jour est parue, mais n'appelez pas trop souvent...

ATTENTION !

Une récompense de un centime symbolique est offerte au premier qui portera à notre connaissance une erreur de documentation, deux centimes au second, etc...

## 2. Les fonctions du BIOS GEMDOS

### Description et écarts par rapport aux spécifications du GEMDOS

Le BIOS du ST est appelable en mode utilisateur 68000.

Le BIOS est réentrant sur trois niveaux. Cela dit il peut se passer plus de trois appels récursifs du BIOS avant que le système ne plante. Aucun test de niveau n'est fait; le premier signe de débordement se produira lorsque le système commencera à avoir des comportements mystérieux, ou éventuellement plantera.

Une application NE DOIT PAS tenter d'accéder à un disque ou à l'imprimante (ceci comprend un appel de 'getbpb', ou une sortie redirectionnée sur un périphérique d'impression) lors de l'éxécution d'un module d'erreur critique, d'interruption timer ou de fin de processus.

REMARQUES

Le BIOS modifie les numéros de fonctions et les adresses de retour mis dans la pile par l'application. Le numéro de fonction sera ZERO au retour. (Pour les curieux, cette configuration permet de sauver plusieurs cycles machines à chaque appel du BIOS).

Module typique d'appel C du trap 13 :

```
_trap13:
     move.l  (sp)+,tr13ret      * empile l'adresse de retour
     trap    #13                * appel de la fonction BIOS
     move.l  tr13ret,-(sp)      * retour à la fonction
     rts                        * d'appel

bss
tr13ret:  ds.l 1                * stockage de l'adresse
                                * de retour
```





### 0x00 getmpb Informations sur la mémoire

```
void getmpb(tampon)
LONG tampon;
```

Lors de l'appel, 'tampon' pointe sur un tampon de 12 octets, représentant une structure MPB (Memory Parameter Block) à remplir. Au retour la structure MPB est pleine.

Les structures se définissent ainsi :

```
#define MPB    struct mpb
#define MD     struct md
#define PD     struct pd

MPB {
     MD   *mp_mfl;    /* liste des blocs de mémoire libre */
     MD   *mp_mal;    /* liste des blocs alloués */
     MD   *mp_rover;  /* idem mp_mfl */
     };

MD  {
     MD   *m_link;        /* MD suivant (ou NULL) */
     long m_start;        /* adresse du bloc */
     long m_length;       /* taille du bloc, en octets */
     PD   *m_own;         /* adresse de la page de base du */
    }                     /* programme propriétaire */
```

(voir 'Varibles Systèmes' pour plus d'information sur l'installation de la TPA de départ.)

### 0x01 bconstat Teste l'état d'un périphérique en entrée

```
WORD bconstat(periph)
WORD periph;
```

Retourne l'état d'un périphérique en entrée, D0.L sera 0L si aucun caractère n'est disponible, -1L si au moins un caractère est disponible.
Le code du périphérique peut être :

| | | |
|---|---|---|
| 0 | PRT: | Sortie parallèle (imprimante) |
| 1 | AUX: | Port RS232 (périphérique AUX:) |
| 2 | CON: | La console écran |
| 3 | Le port MIDI | |
| 4 | Le port clavier (envoi de commande au 6301) | |
| 5 | La sortie écran brute | |

Les opérations valides sur les périphériques sont :

| Opérations | 0 PRT | 1 AUX | 2 CON | 3 MIDI | 4 KBD | 5 écran |
|---|---|---|---|---|---|---|
| bconstat() | non | oui | oui | oui | non | non |
| bconin() | oui | oui | oui | oui | non | non |
| bconout() | oui | oui | oui | oui | oui | oui |
| bcostat() | oui | oui | oui | oui | oui | non |

L'entrée MIDI a un tampon de 80 caractères, géré sous interruption.

Le port clavier (périphérique 4) ne marche qu'en sortie et peut servir à reconfigurer le clavier intelligent.

La sortie écran brute (périphérique 5) imprime des caractères à l'écran sans les interpréter ( les codes de contrôle et 'escape' n'ont pas de signification particulière.)

## 0x02 bconin Attente d'entrée d'un caractère

```
WORD bconin(periph)
WORD periph;
```

'periph' est le numéro de périphérique décrit ci_dessus. La fonction ne revient que lorsqu'un caractère est disponible. Le caractère est renvoyé dans D0.L, avec le mot fort à zéro.

Pour la console (CON:, périphérique 2) la fonction retourne dans l'octet faible du mot fort le 'code clavier' compatible IBM-PC, et le code ASCII dans le l'octet faible du mot faible.

Si le bit 3 de la variable système 'conterm' est à 1, l'octet fort du mot fort contiendra la valeur de la variable système 'kbshift' pour cette touche. (La valeur par défaut du bit 3 de 'conterm' est OFF.)

## 0x03 bconout Envoi d'un caractère sur un périphérique

```
WORD bconout(periph,c)
WORD periph,c;
```

'periph' est le numéro de périphérique décrit dans la fonction 0x01. La fonction envoie le caractère sur le périphérique et ne revient que lorsque le caractère a été écrit.





## 0x04 rwabs Lecture/Écriture de secteurs sur disque

```
LONG rwabs(drap,buf,nbre,sect,disq)
WORD drap;
LONG buf;
WORD nbre,sect,disq;
```

Lecture ou écriture de secteurs logiques sur un disque. Vecteur logique $476.

'drap' peut être :

0 lecture
1 écriture
2 lecture, sans test de 'media change'
3 écriture, sans test de 'media change'

'buf' pointe sur un tampon de stockage. Les transferts à partir d'adresses impaires sont possibles, mais très lents.
'nbre' est le nombre de secteurs à transférer,
'sect' est le secteur de départ du transfert.
'disq' est le numéro du périphérique qui peut être sur le ST :

0 lecteur A:
1 lecteur B: (ou lecteur 'logique' A: sur un système à lecteur unique.)
2 et + disque dur, disque virtuel, réseau, etc..

Les modes 2 et 3 forcent une opération sur disque qui N'affectera PAS le drapeau de changement de disquette (media change). Ceci permet, par exemple, à la routine GEMDOS de formatage, de lire et d'écrire des secteurs logiques après avoir formaté un disque, et permet ensuite au BIOS de reconnaitre un changement de disquette sur une disquette qui vient juste d'être formatée.

NdT:
Lorsque 'buf' est 0L, rwabs a une fontion particulière. Elle permet de provoquer un 'media change' ou d'en retirer un. Dans ce cas là, 'flag' et 'sect' ne sont pas pris en compte, 'nbre' est à 0 si l'on veut qu'il n'y ait pas de 'media change', à 1 si l'on désire en provoquer un, 'disq' doit être inférieur à deux . 0L est retourné dans D0.L.

RETOUR

0L si l'opération a réussie
Un nombre négatif si une erreur s'est produite (c'est au BIOS de détecter un changement de disquette, et de renvoyer le code erreur correspondant.)

## 0x05 setexc  Fixe ou lit un vecteur

```
LONG setexc(vecnum,advec)
WORD vecnum;
LONG advec;
```

'vecnum' est le numéro du vecteur à lire ou à placer.
'advec' est l'adresse du vecteur. Si 'vecnum' est -1L, l'ancienne valeur du vecteur est lue, sans modification.

Les vecteurs 0x00 à 0xFF sont réservés au 68000.

Les vecteurs logiques 0x100 à 0x1FF sont réservés par le GEMDOS. Les vecteurs actuellement implantés sont :

```
0x100      Interruption timer système
0x101      Module d'erreur critique
0x102      Module de fin de programme
0x103 à 0x107  actuellement inemployés (réservés)
```

Les vecteurs 0x200 à 0xFFFF sont réservés pour l'usage OEM (pour le constructeur). Le BIOS du ST n'a rien prévu pour eux.

## 0x06 tickcal

```
LONG tickcal()
```

Fournit la base du temps du timer système en millisecondes (contenu de l'adresse $442). C'est une fonction idiote, puisque le nombre de millisecondes écoulées est placé dans la pile pendant un trap 'timer système'.

## 0x07 getbpb  Fournit l'adresse du bloc de paramètres d'un disque

```
BPB  *getbpb(disq)
WORD disq;
```

'disq' est le numéro du disque (0 = A:, 1 = B:, etc..).

RETOUR

Un pointeur sur le Bloc de Paramètres pour le disque spécifié, 0L si, pour quelque raison, le BPB ne peut être déterminé.

Pour le disque dur, le vecteur getbpb se trouve en l'adresse $472.





## 0x08 bcostat  Demande l'état d'un périphérique en sortie

```
LONG bcostat(periph)
WORD periph;
```

'periph' est un périphérique comme décrit dans la fonction 1.

N.D.T.

Contrairement à ce qui est spécifié, la fonction 'bcostat' inverse les numéros des périphériques 'MIDI' et 'KBD'. Lors de l'appel de cette fonction, 'periph' est 4 pour la sortie MIDI et 3 pour 'KBD'.

RETOUR

```
-1   Le périphérique est prêt à envoyer (pas d'attente du
     prochain appel de sortie du périphérique)
0    le périphérique n'est pas prêt
```

## 0x09 mediach  Teste le changement de disque

```
LONG mediach(lect)
WORD lect;
```

Teste si un changement de disquette a eu lieu sur un lecteur. 'lect' est le numéro du lecteur testé (0 = A:, 1 = B:)

RETOUR

```
0    Le disque n'a pas changé
1    Le disque peut avoir changé
2    Le disque a changé
```

Le GEMDOS renverra une valeur de '1' lors d'une opération de lecture. Si le BIOS a détecté un changement de disquette, il retournera une erreur 'media change'.

## 0x0A drvmap  Demande la carte des disques actifs

```
LONG drvmap()
```

Retourne un vecteur descriptif des disques actifs. Les bits 0 à 31 sont mis à '1' lorsque le disque correspondant est actif, à 0 sinon. (bit 0 à 1 si disque A actif, bit 1 à 1 si disque B actif, etc..).

Les drivers d'installation de disques doivent tenir à jour le long mot '_drvbits' (voir 'Variables Systèmes').

## 0x0B kbshift  Teste ou active les touches spéciales du clavier

```
LONG kbshift(mode)
WORD mode;
```

Si 'mode' est négatif, la fonction retourne l'état, compatible IBM, des touches spéciales du clavier, dans l'octet bas de D0. Si 'mode' n'est pas négatif, la fonction fixe le nouvel état des touches spéciales et retourne l'ancien état.

Assignation des bits:

```
bit 0     touche Shift droite
bit 1     touche Shift gauche
bit 2     touche Control
bit 3     touche Alternate
bit 4     touche Caps_lock
bit 5     bouton droit de la souris (CLR/HOME)
bit 6     bouton gauche de la souris (INSERT)
bit 7     (réservé, actuellement 0)
```





## 3. FONCTIONS XBIOS

Ces fonctions sont accessibles par l'intermédiaire du trap 14. Les conventions d'appel sont les mêmes que pour le trap 13. L'appel peut se faire en mode utilisateur. C'est au programme appelant de nettoyer la pile des paramètres du trap (comme pour tout appel standard C).

Un module typique d'appel C du trap pourrait être:

```
_trap14:
     move.l  (sp)+,tr14ret      * empile l'adresse de retour
     trap    #14                * appel de la fonction XBIOS
     move.l  tr14ret,-(sp)      * retour à la fonction
     rts                        * d'appel

bss
tr14ret:  ds.l 1                * stockage de  l'adresse
                                * de retour
```

et pourrait être utilisé comme suit :

```
/* manière stupide de mettre l'écran à une seule valeur */

  set_screen_to(val)
  WORD val;
       {
        extern long trap14();
        register WORD *scrbase;
        register int  i;

        scrbase = (WORD *)trap14(3);
        for (i= 0x4000;i ;--i)
            *scrbase = val;
       }

/* Xor une gamme de la palette de couleur avec une valeur */

  set_pal(deb,fin,val)
  WORD deb,fin,val;

  {
   while (deb < fin)
       {
        trap14(7,deb,trap14(7,deb,-1) ^ val);
        deb++;
       }
  }
```

## 0x00 initmous Initialisation du gestionnaire de souris

```
void initmous(type,param,vec)
WORD type;
LONG param,vec;
```

Initialise le gestionnaire de souris. 'type' peut être :

| type | action |
|---|---|
| 0 | désactive la souris |
| 1 | active la souris, en mode relatif |
| 2 | active la souris, en mode absolu |
| 4 | active la souris, en mode clavier |

'param' pointe sur un bloc de paramètres définit comme suit:

```
struct param {
        BYTE topmode;
        BYTE buttons;
        BYTE xparam;
        BYTE yparam;
        };
```

'topmode' est : 0 ordonnée 0 en bas
1 ordonnée 0 en haut

'buttons' est un paramètre pour la commande clavier "Place les boutons souris".

'xparam' et 'yparam' sont le seuil, l'échelle, et le facteur delta dépendant du mode souris initialisé.

Pour le mode de souris absolu, quelques autres paramètres suivent immédiatement ce bloc de paramères.

```
struct extra {
        WORD xmax;
        WORD ymax;
        WORD xinitial;
        WORD yinitial;
        };
```

'xmax' et 'ymax' spécifient les coordonnées maximales de la souris. 'xinitial' et 'yinitial' spécifient le coordonnées de départ de la souris.

'vec' pointe sur un module d'interruption souris.(voir la fonction numéro 34, 'kbdvbase' pour plus d' informations sur les modules d'interruption clavier, ainsi que la documentation sur le clavier intelligent).





## 0x01 ssbrk Réservation en haut de la mémoire

```
LONG ssbrk(nbre)
WORD nbre;
```

Réserve 'nbre' octets en haut de mémoire. Retourne un pointeur long sur la base du bloc alloué. Cette fonction doit être appelée avant l'installation du système d'exploitation.

La fonction 'ssbrk' en fait ne sert pas à grand chose, vu qu'elle ne marche pas lorsque le GEMDOS est chargé, à partir du moment où la TPA a déjà été installée.

## 0x02 physBase Fournit la base physique de la mémoire écran

```
LONG _physBase()
```

Demande l'adresse de la base de l'écran physique (au début de la prochaine interruption verticale de trame 'vblank').

## 0x03 logBase Fournit la base logique de la mémoire écran

```
LONG _logBase()
```

Demande l'adresse de la base de l'écran logique. C'est l'adresse dont se sert AES lors de traçage à l'écran.

## 0x04 getRez Demande la résolution courante de l'écran

```
WORD _getRez()
```

Retourne selon la résolution courante :

| | |
|---|---|
| 0 | basse résolution |
| 1 | moyenne résolution |
| 2 | haute résolution |

## 0x05 setScreen Fixe les données de base de l'écran

```
void _setScreen(logbasee,physbase,resol)
LONG logbase,physbase;
WORD resol;
```

Fixe l'adresse logique de l'écran, 'logbase', l'adresse physique, 'physbase', et la résolution de l'écran. Les paramètres négatifs sont ignorés, permettant de ne modifier qu'un seul paramètre.

L'adresse de base logique de l'écran change immédiatement, le registre matériel d'adresse physique de l'écran est également modifié instantanément, mais la nouvelle résolution n'est prise en compte qu'après le prochain balayage vertical.

Lors d'un changement de résolution, l'écran est nettoyé, le curseur est positionné à sa position de départ (home) et l'état du terminal de l'émulateur VT52 est réinitialisé.

## 0x06 setPallete Fixe les couleurs de la palette courante

```
void _setPallete(ptpal)
LONG ptpal;
```

Place les 16 mots pointés par 'ptpal' dans le registre matériel de la palette. 'ptpal' doit être pair. Les nouvelles données ne sont prises en compte que lors de la prochaine interruption verticale.

## 0x07 setColor Fixe une couleur de la palette courante

```
WORD _setColor(numcoul,coul)
WORD numcoul,coul;
```

Place la couleur 'coul' dans la couleur d'index 'numcoul' de la palette . Retourne l'ancienne couleur dans D0.W. Si 'coul' est négatif, la couleur n'est pas changée.





## 0x08 floprd Lecture de secteurs sur disquette

```
WORD _flopr(buf,bidon,lect,sect,piste,face,nbre)
LONG buf,bidon;
WORD lect,sect,piste,face,nbre;
```

Lit un ou plusieurs secteurs à partir d'une disquette.
'bidon' n'est pas employé;
'buf' doit pointer sur un tampon, débutant à une adresse paire, assez grand pour contenir le nombre de secteurs à lire;
'lect' est le numéro de disque (0 ou 1);
'sect' est le numéro de secteur où doit commencer la lecture (habituellement de 1 à 9);
'piste' est le numéro de piste où se positionner;
'face' est le numéro de la face concernée et
'nbre' est le nombre de secteurs à lire (qui doit être égal ou inférieur au nombre de secteurs par piste).

RETOUR

0 si l'opération a réussie
Un numéro d'erreur négatif si l'opération a échouée

## 0x09 flopwr Ecriture de secteurs sur une disquette

```
WORD _flopwr(buf,bidon,lect,sect,piste,face,nbre)
LONG buf,bidon;
WORD lect,sect,piste,face,nbre;
```

Ecriture d'un ou de plusieurs secteurs sur une disquette.
'buf' doit commencer à une adresse paire;
'bidon' ne sert à rien;
'lect' est le numéro de diquette(0 ou 1);
'sect' est le numéro du secteur où doit commencer l'écriture (habituellement de 1 à 9);
'piste' et 'face' sont réspectivement les numéros de piste et de face de diquette concernée; 'nbre' est le nombre de secteurs à écrire (égal ou inférieur au nombre de secteurs par piste).

L'écriture du secteur boot (secteur 1, piste 0, face 0) met la disquette dans l'état 'a peut être changé'. Cela se reflétera lors de prochain appel de rwabs( ) ou de mediach( ).

RETOUR

0 si l'opération a réussie
Un numéro d'erreur négatif en cas d'echec

## 0x0A flopmt Formatage d'une piste de disquette

```
WORD _flopmt(buf,bidon,lect,spp,piste,face,esp,magic,rempl)
LONG buf,bidon;
WORD lect,spp,piste,face,esp,rempl;
LONG magic;
```

'buf' doit pointer sur un tampon débutant à une adresse paire assez grand pour contenir la totalité d'une image de piste (8K pour une piste de neuf secteurs);
'bidon' est inutilisé;
'lect' est le numéro du lecteur (0 ou 1);
'spp' est le nombre de secteurs par piste à formater (habituellement 9);
'piste' est le numéro de piste à formater (habituellement de 0 à 79);
'face' est le numéro de la face à formater (0 ou 1);
'esp' est un facteur d'entrelacement de secteurs (habituellement 1);
'magic' est un long-mot dont la valeur doit être 0x87654321;
'remp' est un mot de remplissage pour les nouveaux secteurs.

RETOUR

0 si l'opération est un succès
Un numéro d'erreur négatif en cas d'échec

La fonction peut retourner une erreur 'soft' lorsqu'elle rencontre un secteur défectueux lors de la passe de vérification. L'appelant peut alors essayer de reformater la disquette, ou choisit d'enregistrer les secteurs défectueux qui ne seront pas compris dans le disque.

Une liste de secteurs défectueux, terminée par un mot à zéro est retournée dans le tampon. Ils ne sont pas nécessairement en ordre croissant. S'il n'y a pas de secteurs défectueux, le premier mot du tampon est à zéro.

Une bonne valeur pour 'rempl' est 0xE5E5. Le quartet haut de chaque octet ne doit jamais être égal à 0xF. Résistez à la tentation de formater un disque avec des secteurs à 0x0000.

Le formatage d'un disque fera entrer la disquette dans l'état 'a definitivement changé'. Cela sera reflété lors d'un appel de la fonction rwabs( ) ou mediach( )

## 0xB used by bios

```
void used_by_bios( )
```

Fonction inutilisée et inutilisable !!!





## 0xC midiws Envoi d'une chaine sur le port MIDI

```
void midiws(nbc,ptr)
WORD nbc;
LONG ptr;
```

'nbc' est le nombre de caractères moins un, à écrire sur le port MIDI.
'ptr' pointe sur la chaine de caractères à écrire.

## 0x0D mfpint Fixe un vecteur d'interruption 68901

```
void _mfpint(numvec,vecteur)
WORD numvec;
LONG vecteur;
```

Initialise le vecteur numéro 'numvec' ( de 0 à 15) avec l'adresse 'vecteur'. Le nouveau vecteur écrase l'ancien et de ce fait rend ce dernier est irrécupérable.

## 0x0E iorec Fournit l'adresse du descripteur du tampon d'entrée

```
LONG iorec(periph)
WORD periph;
```

Retourne un pointeur sur le descripteur de tampon d'entrée série d'un periphérique. 'periph' peut être :

| 'periph' | périphérique |
|---|---|
| 0 | RS232 |
| 1 | clavier |
| 2 | MIDI |

Le descripteur du tampon d'entrée a la structure suivante :

```
struct iorec
    {
     LONG  ibuf;        /* pointeur sur le tampon */
     WORD  ibuftail;    /* taille du tampon */
     WORD  ibuftete;    /* index de tête*/
     WORD  ibuffin;     /* index de fin*/
     WORD  ibufbas;     /* base du débordement */
     WORD  ibufhaut;    /* haut du débordement */
    };
```

Pour le port RS-232, une structure de tampon de sortie suit immédiatement la structure de tampon d'entrée. Le format de la structure du tampon de sortie et le même que celui de la structure du tampon d'entrée.

Si le flux de contrôle est valide et si le nombre de caractères dans le tampon atteint le haut du débordement, le ST demande, (selon le protocole de contrôle de flux), un arrêt de l'envoi de caractères. Lorsque le nombre de caractères dans le tampon tombe en dessous de la base de débordement, le ST demande la reprise de la transmission.

Le processus de gestion de flux de sortie RS232 est identique.

## 0x0F rsconf  Configuration du port série RS-232

```
void rsconf(vit,prot,gen,recep,emet,sync)
WORD vit,prot,gen,recep,emet,sync;
```

Configure le port RS-232. Si un paramètre est -1 (0xFFFF), le registre matériel correspondant n'est pas modifié. 'vit' définit la vitesse de transfert en bauds comme suit:

| 'vit' | vitesse (bps) | 'vit' | vitesse |
|---|---|---|---|
| 0 | 19200 | 8 | 600 |
| 1 | 9600 | 9 | 300 |
| 2 | 4800 | A | 200 |
| 3 | 3600 | B | 150 |
| 4 | 2400 | C | 134 |
| 5 | 2000 | D | 110 |
| 6 | 1800 | E | 75 |
| 7 | 1200 | F | 50 |

'prot' définit le protocole comme suit :

| 'prot' | protocole |
|---|---|
| 0 | pas de protocole (configuration par défaut) |
| 1 | XON/XOFF (^S,^Q) |
| 2 | RTS/CTS |
| 3 | XON/XOFF et RTS/CTS |

'gen', 'recep', 'emet' et 'sync' fixent les registres concernés du 68901.

N.D.T:

Sur plusieurs versions du TOS, les registres 68901 n'étaient pas correctement modifiés par cette fonction. Pour ce faire utilisez de préférence une routine de modification directe des registres du 68901 (voir documentation sur le circuit).





## 0x10 keytbl  Fixe les adresses des tables de décodage

```
LONG keytbl(unshift,shift,capslock)
LONG unshift,shift,capslock;
```

Fixe les pointeurs sur les tables de décodage de touches du clavier. Renvoie un pointeur sur le début d'une structure :

```
struct keytab {
    LONG unshift;    /* -> table touches normales */
    LONG shift;      /* -> table touches, shift enfoncé */
    LONG capslock;   /* -> table de décodage lorsque */
        };           /* capslock est enfoncé */
```

Chaque pointeur de la structure doit pointer sur une table de 128 octets, indiquant comment convertir le code clavier en code ASCII.

## 0x11 random  Fournit un nombre aléatoire

```
LONG _random()
```

Retourne un nombre pseudo aléatoire sur 24 bits dans D0.L. Les bits 24 à 31 sont à zéro. Le nombre devrait être différent à chaque initialisation du système.
L'algorithme découle du vol.2 de Knuth:

$$S = [S * C] + K$$

K = 1, C = 3141592621, S est la semence. La valeur de départ de S est prise dans le compteur '_frclock'. S >> 8 est retouné.

Le bon comportement de la fonction est assez surprenant, excepté pour le bit zéro qui est positionné exactement une fois sur deux. Par conséquent cela ne semble pas une bonne idée de tester les bits individuellement en attendant d'eux un bon comportement.

## 0x12 protobt Fabrication d'une image du secteur 'boot'

```
void _protobt(buf,numser,disqtyp,drapex)
LONG buf,numser;
WORD disqtyp,drapex;
```

Fabrique une image prototype du secteur 'boot'. Une fois que celle ci est faite, cette image pourra être écrite sur le secteur 'boot' du disque (secteur 1, piste 0).

'buf' pointe sur un tampon de 512 octets (qui peut déjà contenir une image de secteur boot ou n'importe quoi).

'numser' est le numéro de série à inscrire dans le secteur boot. Si 'numser' est (-1) le numéro de série est inchangé; s'il est plus grand ou égal à 0x01000000, un nombre aléatoire est généré et placé dans le secteur boot.

'disqtyp' peut être soit (-1) si l'on ne veut pas toucher au type de disque, soit :

```
0 :  40 pistes, simple face (180K)
1 :  40 pistes, double face (360K)
2 :  80 pistes, simple face (360K)
3 :  80 pistes, double face (720K)
```

Si 'drapex' est à 1, le secteur boot est exécutable.
Si 'drapex' est à 0, le secteur boot n'est pas exécutable.
Si 'drapex' est à -1,le secteur boot n'est pas modifié.

## 0x13 flopver Vérification des secteurs d'un disque

```
WORD _flopver(buf,bidon,numdisq,numsec,piste,face,nbre)
LONG buf,bidon;
WORD numdisq,numsec,piste,face,nbre;
```

Vérifie, par simple lecture, un ou plusieurs secteurs d'une disquette.
'buf' doit pointer sur un tampon de 1024 octets, débutant à une adresse paire.
'bidon' est inutilisé.
'numdis' est le numéro du disque (0 ou 1);
'numsec' est le numéro du secteur où commence la vérification (habituellement entre 1 et 9);
'piste' est le numéro de la piste où se positionner;
'face' est le numéro de la face concernée;
'nbre' le nombre de secteurs à vérifier (qui doit être inférieur ou égal au nombre de secteurs par piste).

Une liste des secteurs défectueux, terminée par un mot à zéro (0.W), est retournée dans 'buf'. Ils ne sont pas obligatoirement dans un ordre numérique. S'il n'y a aucun mauvais secteur, le premier mot du tampon sera zéro.





## 0x14 scrdmp Copie d'écran sur imprimante

```
void scrdmp()
```

Fait une copie d'écran sur imprimante. Actuellement seule la version monochrome est disponible.

## 0x15 cursconf Fixe ou lit les attributs du curseur

```
WORD cursconf(code,clign)
WORD code,clign;
```

Configure le curseur d'écran. Le 'code' de fonction est l'un des suivants :

```
0   désactive le curseur
1   active le curseur
2   rend le curseur clignotant
3   rend le curseur fixe
4   fixe la période de clignotement du curseur à 'clign'
5   lit la période de clignotement de curseur
```

La période de clignotement est basée sur le balayage vidéo (60Hz pour la couleur, 70Hz pour le noir et blanc, 50Hz pour le PAL). 'clign' est le nombre de demi-cycles horloge.

## 0x16 settime Fixe la date et l'heure

```
void settime(datheure)
LONG datheure;
```

Place la date et l'heure dans le clavier intelligent (6301). 'datheure' est un long_mot au format DOS d'heure et de date. L'heure se place dans le mot bas, la date dans le mot haut.

## 0x17 gettime Lit l'heure et la date

```
LONG gettime()
```

Lit l'heure et la date du clavier (6301) et les retourne dans un long mot de 32 bits au format DOS. L'heure se trouve dans le mot bas, la date dans le mot haut.

## 0x18 bioskeys

```
void bioskeys()
```

Réinitialise les tables de conversion des touches du clavier.

## 0x19 ikbdws  Envoi d'un commande au 6301

```
void ikbdws(nbre,ptcom)
WORD nbre;
LONG ptcom;
```

Envoi d'une chaîne de commandes au microprocesseur clavier (6301).
'nbre' est le nombre d'octets à envoyer moins un;
'ptcom' est l'adresse de la chaîne à envoyer.

## 0x20 jdisint

```
void jdisint(intnum)
WORD intnum;
```

Désactive l'interruption 'intnum' du 68901.

## 0x21 jenabint

```
void jenabint(intnum)
WORD intnum;
```

Active l'interruption 'intnum' du 68901.

## 0x1C giaccess

```
BYTE giaccess(data,regnum)
BYTE data;
WORD regnum;
```

Lit ou écrit un registre du synthétiseur de son. 'regnum' est le numéro du registre.Le bit 7 est à 0 lors d'une lecture et à 1 pour une écriture. 'data' est l'octet à écrire.

Les procédures qui lisent directement la valeur d'un registre, la modifient, et écrivent le résultat dans ce registre sont très critiques. En particulier, le BIOS, met à jour fréquemment, le registre du port A, et toute lecture/modification/écriture sur ce port peut être funeste. Il est très vivement conseillé de se servir des fonctions prévues à cet effet (cf. offgibit, ongibit).





## 0x1D offgibit

```
void offdigit(bitnum)
WORD bitnum;
```

Met à zéro un bit 'bitnum' du registre du port A.

## 0x1E ongibit

```
void ongibit(bitnum)
WORD bitnum;
```

Met à 1 un bit 'bitnum' du registre du port A.

## 0x1F xbtimer  Initialisation d'un timer du 68901

```
void xbtimer(timer,controle,data,advec)
WORD timer,controle,data;
LONG advec;
```

Le numéro du timer 'timer' (0,1,2,3) correspond aux timers du 68901 (A,B,C,D). 'controle' est le registre de contrôle du timer. 'data' est l'octet placé dans le registre de données.'advec' pointe sur une routine d'interruption.

Implantation des timers :

| Timer | emploi |
|---|---|
| A | réservé pour les applications et l'utilisateur final |
| B | Utilisé par le rafraîchissement d'écran (hblank sync, etc..) |
| C | Timer système (200Hz) |
| D | contrôle vitesse RS-232 (vecteur d'interruption disponible en cas de non utilisation de la sortie série). |

## 0x20 dosound  Envoi d'un commande de son

```
void dosound(ptcom)
LONG ptcom;
```

'ptcom' pointe sur un jeu de commandes, organisé en octets, à envoyer au synthétiseur de son.

Les numéros de commande, de 0x0 à 0xF, ont un octet d'argument. La commande 0x0 envoie l'octet sur le registre 0, la commande 0x1 sur le registre 1, etc...

La commande 0x80 place l'octet d'argument dans un registre temporaire.

La commande 0x81 demande 3 octets comme paramètres. Le premier paramètre est le numéro de registre à charger, se servant du registre temporaire. Le second est une valeur à ajouter au registre temporaire. Le troisième paramètre est la valeur de fin. L'instruction est exécutée, (une fois à chaque mise à jour), jusqu'a ce que le registre temporaire soit égal à la valeur de fin.

Les commandes 0x82 à 0xFF demandent un paramètre sur un octet. Si le paramètre est zéro, le son est fini. Autrement le paramètre représente le nombre d'interruptions 'timer système' avant la prochaine mise à jour.





## 0x21 setprt  Fixe ou lit la configuration imprimante

```
WORD setprt(config)
WORD config;
```

Si 'config' est -1 (0xFFFF), la fonction retourne la configuration courante de l'imprimante, sinon une nouvelle configuration est fixée et l'ancienne retournée.

Les bits de configuration sont établis ainsi :

| Bits | si bit à 0 | si bit à 1 |
|---|---|---|
| 0 | matricielle | marguerite |
| 1 | monochrome | couleur |
| 2 | imprimante Atari (1280 pts/ligne) | imprimante 'Epson' (960 pts/ligne) |
| 3 | qualité épreuve | qualité définitive |
| 4 | port parallèle | port série (RS232) |
| 5 | listing | feuille à feuille |
| 6 | (réservé) | |
| 7 | (réservé) | |
| 8 | (réservé) | |
| 9 | (réservé) | |
| 10 | (réservé) | |
| 11 | (réservé) | |
| 12 | (réservé) | |
| 13 | (réservé) | |
| 14 | (réservé) | |
| 15 | 0 | |

## 0x22 kbdvbase  Fournit la base du descripteur de vecteurs clavier

```
LONG kbdvbase()
```

Retourne un pointeur sur la base d'une structure :

```
struct kbdvecs {
        LONG midivec;     /* routine entrée MIDI */
        LONG vkbderr;     /* traitement erreurs clavier */
        LONG vmiderr;     /* traitement erreurs  MIDI   */
        LONG statvec;     /* gestion. état clavier intelligent*/
        LONG mousevec;    /* gestionnaire souris */
        LONG clockvec;    /* gestionnaire horloge */
        LONG joyvec;      /* gestionnaire joystick */
        LONG midisys;     /* routine système MIDI */
        LONG ikbdsys;     /* routine système 6301 */
              };
```

'midivec' pointe à l'initialisation sur une routine 'bufférisée' du BIOS. D0.B contiendra le caractère lu sur le port MIDI.
'vkbderr' et 'vmiderr' sont appelés à chaque fois qu'une erreur clavier ou MIDI est détectée sur son ACIA 6850 (routine guère utile).
'statvec', 'mousevec', 'clockvec',et 'joyvec' pointent respectivement sur des routines de traitement de l'état du clavier intelligent, de l'interruption souris, de l'interruption horloge, et de l'interruption manette de jeu.
Les gestionnaires reçoivent l'adresse d'un tampon, où ont été placés les codes envoyés, par l'intermédiaire d'un pointeur long se trouvant dans A0 et dans la pile. A1 pointe la fin de ce tampon.
GEM se sert du module d'interruption souris. Les routines doivent se terminer par un 'RTS' et ne doivent pas durer plus que 1 ms.

Les vecteurs 'midisys' et 'ikbdsys' sont appelés chaque fois qu'un caractère est disponible sur l'ACIA 6850 concerné. Au début ils pointent sur des routines par défaut (le module MIDI est détourné par 'midivec' et le module de clavier analyse les paquets claviers et appelle le vecteur de gestionnaire approprié.)

## 0x23 kbrate

```
WORD kbrate(initial,repet)
WORD initial,repet;
```

Lit ou fixe la fréquence d'autorépétition. 'initial' fixe le delai avant le début d'autorépétition, 'repet' fixe le délai entre chacune des répétitions. Si un paramètre est à -1 (0xFFFF), il n'est pas modifié. La base de temps est prise sur le 'timer système' (50Hz).

Au retour les anciennes valeurs se trouvent dans D0, 'initial' dans l'octet haut du mot bas et 'repet' dans l'octet bas du mot bas.

## 0x24 prtblk copie d'écran sur imprimante paramètrable

```
void _prtblk()
```

Primitive de Prtblk() (voir PTRBLK chapitre16)





## 0x25 vsync Attente de la prochaine interruption verticale

```
void vsync()
```

Attend la prochaine interruption verticale et revient. Utile pour la synchronisation graphique avec l'interruption verticale.

## 0x26 superex Exécution d'une routine en mode TRAP

```
void superex(adrout)
LONG adrout;
```

'adrout' doit pointer sur une routine, terminée par 'RTS' qui sera exécutée en mode superviseur. La routine ne peut faire d'appel au BIOS ou au GEMDOS. Cette fonction a été créée dans le but de permettre à un programmme de 'taper' dans les adresses systèmes protégées sans avoir à se soucier du passage en mode superviseur.

## 0x27 puntaes Suppression de l'AES

```
void puntaes()
```

Supprime l'AES du systéme, libérant ainsi la mémoire qu'il occupait. Si l'AES est résident, il sera mis de côté et le système 'rebootera'. Si l'AES n'est pas résident ( s'il a été préalablement retiré) la fonction reviendra.

Il n'existe aucun moyen de supprimer l'AES et de revenir sans 'rebooter' le système.

## 4. Les fonctions de déplacement du curseur

Voici les séquences 'escape' qui sont prises en compte par la fonction conout() du BIOS. La plupart d'entre elles émulent l'éditeur du terminal VT-52 (c'est la manière la plus facile de le faire).

**ESC A Curseur vers le haut**

Cette séquence remonte le curseur d'une ligne. Si le curseur est déjà sur la plus haute ligne de l'écran, cette séquence n'aura pas d'effet.

**ESC B Curseur vers le bas**

Déplace le curseur d'une ligne vers le bas. Si le curseur est déjà sur la dernière ligne de l'écran, cette séquence n'aura pas d'effet.

**ESC C Avance le curseur**

Le curseur est déplacé d'une position vers la droite. Cette séquence n'aura aucun effet si elle conduit le curseur à sortir de l'écran.

**ESC D Recule le curseur**

Le curseur est déplacé d'une position vers la gauche. C'est un déplacement non destructeur, le caractère se trouvant alors sous le curseur n'est pas remplacé par un blanc. Si le curseur est déjà en colonne 0, il ne se produira rien.

**ESC E CLEAR-HOME**

Le curseur est positionné en colonne 0, rangée 0 (coin haut à gauche de l'écran) et l'écran est nettoyé.

**ESC H HOME**

Le curseur est positionné en colonne 0, rangée 0. L'écran n'est pas nettoyé.





**ESC I Remonte le curseur**

Déplace le curseur sur la ligne précédente en gardant sa position horizontale. Si le curseur est sur la ligne du haut, un scrolling vers le bas est généré.

**ESC J Efface la fin d'une page**

Efface tous les caractères depuis la position courante du curseur, jusqu'à la fin de la page.

**ESC K Efface la fin d'une ligne**

Efface tous les caractères depuis la position courante du curseur, jusqu'à la fin de la ligne.

**ESC L Insere une ligne**

Insere une nouvelle ligne blanche en déplaçant la ligne sur laquelle se trouve le curseur, et toutes les lignes suivantes, d'une ligne vers le bas. Le curseur est alors positionné au début de la nouvelle ligne.

**ESC M Efface une ligne**

Efface le contenu de la ligne sur laquelle se trouve le curseur, place le curseur au début de la ligne, déplace toutes les lignes suivantes d'une position vers le haut, et rajoute une ligne blanche en bas.

**ESC Y Positionne le curseur**

Les deux caractères suivants le 'Y' spécifient le numéro de rangée et de colonne, moins $20, où positionner le curseur. Le premier caractère spécifie la rangée, le second la colonne.

**ESC b Fixe la couleur de caractère**

'ESC b' doit être suivi par un caractère de sélection de couleur. Seuls les quatres bits les moins significatifs sont

pris en compte.

Palette des bits de contrôle:

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| X | X | X | X | index de la couleur | | | |

('X' = ignoré)

**ESC c  Fixe la couleur de fond**

Fixe la couleur des cellules qui contient les caractères. 'ESC c' doit être suivi par un caractère de sélection de couleur. Seuls les quatres bits les moins significatifs sont pris en compte.(voir 'ESC b');

**ESC d  Efface le début de l'écran**

L'écran est effacé du coin haut gauche, à la position du curseur comprise.

**ESC e  Active le curseur**

Le curseur est rendu visible. Il peut être déplacé à l'écran grace aux séquences 'ESC' définies dans cette appendice.

**ESC f  désactive le curseur**

Le curseur est rendu invisible, cependant il peut encore être déplacé à l'écran grace aux séquences 'ESC'.

**ESC j  Sauve la position courante du curseur**

La position courante du curseur est sauvée. La dernière position du curseur sauvée peut être restituée grace à la fonction 'ESC k'.

**ESC k  Restitue la dernière position sauvée du curseur**

Le curseur est placé à la dernière position sauvée. Si cette fonction est appelée sans qu'une position ait été sauvée





préalablement, le curseur est positionné sur le point de départ de l'écran (coin en haut à gauche).

**ESC l  Efface une ligne entière**

La ligne sur laquelle se trouve le curseur est effacée et le curseur est placé au début de la ligne.

**ESC o  Efface le début d'une ligne**

Efface une ligne depuis son début jusqu'à la position courante du curseur incluse.

**ESC p  Passe en mode vidéo inversé**

Passe en 'vidéo inverse' de telle sorte que la couleur d'écriture de caractères devient la couleur de fond, et inversement.

**ESC q  Sort du mode vidéo inversé**

Repasse en mode vidéo normale.

**ESC v  Rend le retour de ligne automatique**

Le premier caractère envoyé après la dernière position affichable sur une ligne est automatiquement placé au début de la ligne suivante. Un 'scrolling' vertical a lieu si nécessaire.

**ESC w  Désactive le retour de ligne automatique**

Après l'appel de cette séquence, les caractères envoyés après la dernière position affichable sur une ligne sont imprimés en superposition. Seul le dernier caractère envoyé est donc visible à la dernière position de la ligne.

## 4. LES INTERRUPTIONS

Le ST se sert de quatres des seize vecteurs TRAP fournis par le 68000. Tous les autres traps sont accessibles aux applications.

| Trap | emploi |
|---|---|
| 0 | (sans) |
| 1 | interface GEMDOS |
| 2 | extensions DOS (VDI, AES) |
| 3 | (sans) |
| 4 | (sans) |
| 5 | (sans) |
| 6 | (sans) |
| 7 | (sans) |
| 8 | (sans) |
| 9 | (sans) |
| 10 | (sans) |
| 11 | (sans) |
| 12 | (sans) |
| 13 | BIOS |
| 14 | extensions BIOS Atari (XBIOS) |
| 15 | (sans) |

Les interruptions 68901 sont basées en 0x100. Les seize longs mots sont fixés par le matériel.

| vecteur | Fonction |
|---|---|
| 0x100 | (désactivé) interruption port parallèle |
| 0x104 | (désactivé) CD de l'entrée/sortie RS-232 |
| 0x108 | CTS de l'entrée/sortie RS-232 |
| 0x10C | (désactivé) |
| 0x110 | (désactivé) |
| 0x114 | horloge système 200Hz |
| 0x118 | ACIA 6850 MIDI et clavier |
| 0x11C | (désactivé) détection FDC/HDC |
| 0x120 | Hsync (inter.horizontales, désactivé au départ) |
| 0x124 | Erreur à l'émission RS-232 |
| 0x128 | Tampon d'émission RS-232 vide |
| 0x12C | Erreur de réception RS-232 |
| 0x130 | Tampon de réception RS-232 plein |
| 0x134 | (désactivé) |
| 0x138 | (désactivé) indicateur de sonnerie RS-232 |
| 0x13C | (désactivé) détection de moniteur monochrome |

Le vecteur de division par zéro pointe sur un 'RTE'.





Tous les autres Traps (Erreur Bus,erreur d'adresse,etc..) pointent sur un module qui stocke l'état du processeur et essaie de terminer le programme en cours. (voir Initialisation du Système).

Le vecteur 0x1010 (ligne 'A') sert de raccourci au VDI pour les primitives graphiques. C'est une interface puissante et très utile. Voir le document 'Ligne A' pour plus d'informations.

Le vecteur 0x1111 (ligne 'F') est actuellement employé par le système interne. Si vous trafiquez ce vecteur l'AES risque de planter.

L'interruption FDC/HDC peut être activée par le gestionnaire de disque dur. Le lecteur de disquettes s'assure que cette interrupton est désactivée. (Il boucle sur le bit d'état). C'est aux autres gestionnaires du système de s'assurer que l'interruption FDC/HDC est désactivée lors de la prise de contrôle d'un processus d'écriture/lecture/formatage de disquette.

Le niveau de priorité normal du processeur (IPL) est de 3, ceci afin de se prévenir de l'occurence de l'interruption horizontale de ligne 'HBLANK' (autovecteur de niveau 2) à chaque fin de ligne.(Cela consommerait à peu près 10% du temps du système en mode couleur et 22% en mode monochrome). Le module d'interruption HBLANK, par défaut, replace le niveau de priorité à 3 et fait un 'RTE' afin d'empécher les programmes d'employer le niveau de priorité 0. Pour employer HBLANK, servez vous d'un niveau de priorité de 1.

Pour ne pas avoir de problème lors du changement de palette de couleur en fin de ligne dans un programme se servant des vecteurs d'interruption HBLANK et HSYNC la séquence suivante permettra de garder le système intact et procurera un affichage correct:

1. Redirectionnez l'interruption MIDI/clavier vers une routine qui abaisse "l'IPL" à 5 et puis faites un saut au vecteur d'origine.

2. Pendant la partie 'critique' écran, redirectionnez le vecteur d'interruption horloge système 200Hz vers une routine qui incrémente un compteur et faites un 'RTE'. Le compteur gardera une trace du nombre d'interruptions système qui se produisent pendant la partie critique.

3. Après la partie critique, le bloc provoque une interruption (au niveau d'IPL 6) et appelle le module de gestion d'horloge système (JMP au vecteur d'interruption, avec un faux SR et l'adresse de retour dans la pile) avec le nombre d'interruptions du compteur.

## APPEL DU BIOS A PARTIR D'UN MODULE D'INTERRUPTION

Il est possible d'appeler le BIOS à partir d'un module d'interruption. Plus exactement, il est possible, pour un module à la fois, d'appeler le BIOS. Il n'est, par contre, pas possible de faire appel au GEMDOS, ni au VDI, ni à l'AES dans un module d'interruption.

La partie critique se situe dans le code du module du trap BIOS lorsque les registres sont sauvés et restitués dans la zone de sauvegarde des registres. La variable 'savptr' doit être maintenue correcte.

```
*
*    appel du BIOS à partir d'une interruption
*
     savptr    $4A2       * poiteur sur la zone BIOS de sauvegarde
                          * des registres
     sav_amt   $23*2      * nombre de mots BIOS sauvés dans la pile

     module d'interruption:
          .
          .
     * création de la zone de sauvegarde TRAP

          sub.l     #sav_amt,savptr
          .
          .
          . appel trap BIOS (#13 et # 14 seulement)
          .

     * restitution de l'ancien environnement trap

          add.l     #sav_amt,savptr
          .
          .
          rte                 * ou autre chose
```

ATTENTION

Seul UN module d'interruption peut faire ceci. Deux modules ne peuvent s'emboîter et faire un appel de cette manière.





## LES INTERRUPTIONS VERTICALES

Cette partie traite du module d'interruptions verticales du système (VBI), pointé par le vecteur VBI d'adresse $70.

Le module VBI incrémente le "compteur cadre", 'frclock' et teste ensuite 'vblsem' pour vérifier les différentes exclusions. Si 'vblsem' est inférieur ou égal à zéro, aucun autre code VBI n'est exécuté. Autrement, tous les registres sont sauvés dans la pile et le "compteur vblank", 'vbclock' est incrémenté.

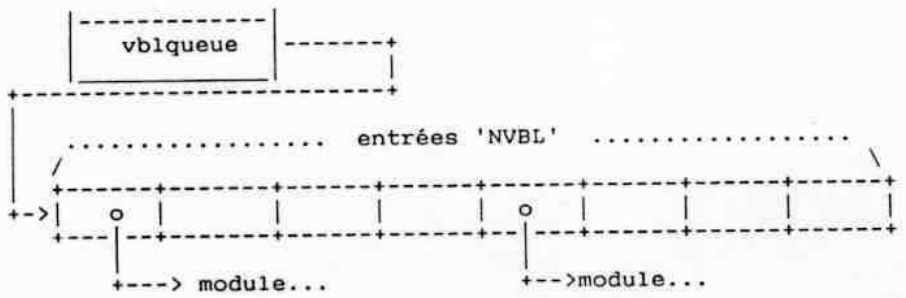
Si le système se trouve en mode haute résolution (shiftmd >= 2) et que l'on branche un moniteur basse résolution, la résolution est placée dans la variable 'defshiftmd'. Ce test est nécessaire pour empêcher qu'un moniteur couleur ne soit 'grillé' s'il était dirigé par le signal haute résolution du ST.

Le module appelle une routine d'effacement de curseur.

Si 'colorptr' est différent de zéro, la palette de couleurs est chargée avec les seize mots pointés par 'colorptr', puis 'colorptr' est remis à zéro.

Si 'screenpt' est différent de zéro, la base physique de l'écran est placée à l'adresse 'screenpt', puis 'screenpt' est remis à zéro.

Il n'y a pas de limitation du nombres de vecteur VBI rajoutés. Ils sont exécutés juste avant le retour du module d'interruption VBI. 'nvbls' contient le nombre d'emplacements de vecteurs NVBL et 'vlqueue' pointe sur un tableau contenant les adresses des modules ou NULL (en cas d'emplacement vide).




Le TOS alloue au départ 8 emplacements VBI. Le premier emplacement est réservé à la routine VBI, GEM. Pour ajouter un module 'différé', placez le pointeur de votre routine dans un emplacement libre (NULL). S'il n'y a plus de place pour ce vecteur, il vous faut allouer un tableau plus grand. Recopiez tous les vecteurs dans le nouveau tableau en y ajoutant le vôtre, nettoyez tous les emplacements inutilisés et mettez à jour 'vblqueue' et 'nvbls'.

Les modules rajoutés devront revenir avec un 'RTS' et non un 'rte' et pouront se servir de tous les registres,excepté le pointeur de pile.

Les applications sont chargées du nettoyage des vecteurs vbi qu'elles ont installés, avant qu'elles ne soient terminées.





## 6. VARIABLES SYSTEME

Voici la liste des variable du BIOS ST 'coulées dans le béton'. Leur emplacement et leur signification, lors de prochaines révisions du BIOS ST, sont garantis contre tout changement.

Toutes les autres variables en RAM et routines en ROM, ou vecteurs en dessous de $400 qui ne sont pas documentés sont à peu près certains de changer. Il est important de ne pas s'appuyer sur une variable ou sur une adresse en ROM non documentée.

N.D.T:

Toutes les précautions d'usage étant respectées, le programmeur expérimenté pourra retrouver les adresses système hors de cette zone à partir des variables mêmes de cette zone (ex: $4B2, $4B6, etc...) où à partir de fonctions trap (ex: Kbdbase, Iorec, etc..)

**etv_timer $400 (long)**

Pointeur sur la routine de traitement des interruptions Timer. (vecteur logique $100). Voir la documentation GEMDOS.

**etv_critic $404 (long)**

Pointeur sur la routine de traitement des erreurs fatales. (vecteur logique $101). Voir la documentation GEMDOS.

**etv_term $408 (long)**

Pointeur vers la routine de fin de programme. (vecteur logique $102). Voir la documentation GEMDOS.

**etv_xtra $40C (longs)**

Emplacement des 5 vecteurs logiques $103 à $107.

**memvalid $420 (long)**

Contient le nombre magique $752019F3, qui , avec 'memval2', valide 'memcntlr' et indique un Reset à froid bien effectué.

**memcntlr** **$424 (octet)**

Contient dans le quartet bas le contrôleur de configuration mémoire. Les valeurs peuvent être:

| Taille Mémoire | valeur |
|---|---|
| 128K | 0 |
| 512K | 4 |
| 256K (2 pages) | 1 |
| 1MB (2 pages) | 5 |

**resvalid** **$426 (long)**

Si 'resvalid' est le nombre magique $31415926, le système, au RESET, fera un saut à 'resvector'.

**resvector** **$42A (long)**

Pointeur vers la routine système de RESET à chaud, valide si 'resvalid' contient le nombre magique. Cette routine est appelée tout au début de l'initialisation du système (avant qu'aucun registre matériel, comprenant le registre de configuration du contrôleur de mémoire n'ait été touché). L'adresse de retour sera dans A6 et le contenu des deux pointeurs de pile sera sans intérêt.

**phystop** **$42E (long)**

Haut physique de la RAM. Contient un pointeur sur le premier octet non disponible. (ex. $80000 sur les ST 512K)

**_membot** **$432 (long)**

Bas de la mémoire vive utilisateur. La fonction BIOS 'getmpb' se sert de cette valeur comme le bas de la zone TPA GEMDOS.

**_memtop** **$436 (long)**

Haut de la mémoire vive utilisateur (non compris la mémoire écran. La fonction BIOS 'getmpb' se sert de cette valeur pour définir la fin de la zone TPA GEMDOS.

**memval2** **$43A (long)**

Contient le nombre magique $237698AA, qui avec 'memvalid' valide 'memcntlr' et indique un reset à froid réussi.





**flock** **$43E (mot)**

Sert à verrouiller l'emploi du circuit DMA. Doit être différent de zéro pour être sûr que le système ne touche pas les registres du circuit DMA pendant une interruption verticale. Doit être à 1 lors de l'emploi du bus DMA et remis impérativement à 0 après.

**seekrate** **$440 (word)**

Période de recherche sur disquette. Les bits zéro et un contiennent la période de recherche pour les deux lecteurs :

00 6 ms
01 12 ms
10 2 ms
11 3 ms (valeur par défaut)

**_timr_ms** **$442 (mot)**

Période du 'Timer' système. Doit être à $14 pour une fréquence d'interruption timer de 50Hz. Retourné par la fonction BIOS 'tickcal' et passé par l'intermédiaire de la pile au vecteur d'interruption timer.

**_fverify** **$444 (mot)**

Drapeau de vérification d'écriture. Lorsqu'il n'est pas à zéro, valeur par défaut, les écritures sur disquette sont vérifiées par une lecture. S'il est à zéro, il n'y a pas de vérification.

**_bootdev** **$446 (mot)**

Contient le numéro du disque avec lequel le système à été initialisé. Le BIOS construit une chaîne d'environnement à partir de cette variable avant d'installer le bureau.

**palmode** **$448 (mot)**

0 si le système est en NTSC (vidéo 60Hz)
sinon le système est en PAL (vidéo 50Hz)

N.D.T: Variable inutilisée dans le TOS français

**defshiftmd** **$44A (octet)**

Résolution vidéo par défaut. Si l'on essaie de passer de mode monochrome en mode couleur, 'defshiftmd' contiendra la résolution dans laquelle le système va passer.

**sshiftmd** **$44C (mot)**

Reflet du registre matériel 'shiftmd':

| | | |
|---|---|---|
| 0 | 320x200x4 | (basse résolution) |
| 1 | 640x200x2 | (moyenne résolution) |
| 2 | 640x400x1 | (haute résolution, monochrome) |

**v_bas_ad** **$44E (long)**

Pointeur sur la base de la mémoire écran qui est toujours un bloc mémoire de 32K et débute toujours à une adresse multiple de 512.

**vblsem** **$452 (mot)**

Drapeau pour forcer l'exclusion des interruptions verticales. Doit être à 1 si traitement autorisé.

**nvbls** **$454 (mot)**

Nombre de longs_mots sur lesquels pointe '_vblqueue'. (8 par défaut, au RESET).

**_vblqueue** **$456 (long)**

Pointeur sur une liste d'adresses de routines d'interruption verticale de trame (vblank).

**colorptr** **$45A (long)**

Pointeur sur une liste de 16 mots à charger dans les registres de la palette de couleurs lors au prochain vblank. Si NULL (0L), la palette n'est pas chargée. 'colorptr' est remis à zéro après chaque chargement de palette.

**screenpt** **$45E (long)**

Pointeur sur la base de la mémoire écran à installer lors de l'interruption fréquence de trame suivante. Si NULL, la base de l'écran n'est pas changée.





**_vbclock** **$462 (long)**

Compteur d'interruptions verticales de trame.

**_frclock** **$466 (long)**

Compteur d'interruptions verticale qui ont été prises en compte (non bloquées par 'vblsem').

**hdb_init** **$46A (long)**

Pointeur sur la routine d'initialisation des lecteurs de disquettes, NULL si pas employé.

**swv_vec** **$46E (long)**

Pointeur sur une routine exécutée lorsque le système détecte une transition sur l'entrée de détection de moniteur monochrome. (passage de haute en basse résolution ou vice_versa). 'swv_vec' pointant au départ sur une routine de reset, le système opérera un reset si l'utilisateur change de moniteur.

**hdv_bpb** **$472 (long)**

Pointeur sur une routine retournant un descripteur de disque dur (BPB). Mêmes conventions d'appel que pour la fonction BIOS 'getbpb'. NULL si inutilisé.

**hdv_rw** **$476 (long)**

Pointeur sur une routine de lecture/écriture de disque dur. Mêmes conventions d'appel que la fonction BIOS 'rwabs'. NULL si inutilisé.

**hdv_boot** **$47A (long)**

Pointeur sur une routine de secteur 'boot' à partir du disque dur et des disquettes. NULL si inutilisé.

**hdv_mediach** **$47E (long)**

Pointeur sur une routine renvoyant le code de changement de disque. Idem changement de disquette. NULL si inutilisé.

**_cmdload $482 (word)**

Non nul lorsque le 'boot' du disque a demandé de charger et d'exécuter le 'COMMAND.PRG'. Un programme 'shell' ou un programme est alors chargé à la place du bureau. Peut être positionné à une valeur par le secteur 'boot'.

**conterm $484 (octet)**

Attributs de la console :

| Bit | Fonction |
|---|---|
| 0 | à 1 : touches sonores |
| 1 | à 1 : autorépétition validée |
| 2 | à 1 : sonnerie valide à l'écriture de ^G sur CON: |
| 3 | à 1 : retourne la valeur des touches spéciales du clavier dans les bits 24 à 31 de D0.L lors de l'appel de la fonction BIOS conin(); |
| | à 0 : Ne place rien dans les bits 24 à 31 (cf. Bconout) |

**themd $48e (longs)**

Rempli lors de l'appel de la fonction BIOS 'getmpb'. Indique au GEMDOS les limites de la TPA. La structure est ainsi:

```
struct  MD
    {
     MD   *m_link;      /* -> MD suivante (NULL) */
     long m_start;      /* début de la TPA */
     long m_length;     /* nbre d'octets de la TPA */
     PD   *m_own;       /* propriétaire de MD (NULL)*/
    };
```

Cette structure NE peut PAS changer après l'initialisation du GEMDOS. De plus il ne peut y avoir qu'une telle structure (vous ne pouvez vous servir de 'm_link'). Un jour, dans un GEMDOS amélioré, ces limitations seront levées.

**savprt $4A2 (long)**

Pointeur sur la zone de sauvegarde des fonctions BIOS.

**_nflops $4A6 (mot)**

Nombre de lecteurs de disquettes attachés au système (0,1, ou 2).





**sav_context $4AE (long)**

Pointeur sur la zone de sauvegarde de contexte du processeur. (informations complémentaires plus tard).

N.D.T: Cette variable est actuellement inutilisée.

**_bufl $4B2 (long)**

Deux pointeurs de tampons descripteurs BCB (BUFFER CONTROL BLOCK). Le premier pointe sur tampon descripteur de secteurs données, le second sur un descripteur de secteurs FAT et catalogues.

```
struct BCB
     {
      BCB      *b_link;  /* pointe BCB suivant */
      int      b_bufdrv; /* numéro de disque ou -1 */
      int      b_buftyp; /* type de buffer */
      int      b_bufrec; /* numéro d'enregistrement */
      int      b_dirty;  /* drap. chmt contenu du tampon */
      DMD      *b_dm;    /* -> descripteur de disque */
      char     *b_bufr;  /* -> tampon de stockage */
     };
```

**_hz_200 $4BA (long)**

Compteur d'interruption horloge. Employé en division par quatre pour un timer système de 50Hz.

**the_env $4BE (octet[4]**

Chaine d'environnement par défaut. Quatre octets à $00.

**_drvbits $4C2 (long)**

Vecteur de 32 bits, retourné par la fonction BIOS $A 'drvmap', des disques actifs. Si les deux lecteurs de disquettes sont branchées, la valeur est 3.

**_dskbufp $4C6 (long)**

Pointe sur un tampon de disque de 1024 octets, quelque part dans la BSS (zone des données non initialisées) du système. Le tampon sert également pour quelques opérations graphiques VDI, et ne doit pas servir dans des routines d'interruption.

**_prt_cnt** **$4EE (mot)**

Drapeau de copie d'écran, initialisé à -1. La pression des touches ALT et HELP provoque son incrémentation. Lorsque sa valeur est $0000 une copie d'écran est activée, dès que la valeur diffère de $0000 l'impression est arrêtée.

**_sysbase** **$4F2 (long)**

Pointeur sur la base du TOS (en ROM ou en RAM).

**_shell_p** **$4F6 (long)**

Pointeur sur le bloc d'information du 'shell'.

**end_os** **$4fa (long)**

Pointe juste après le dernier octet du bas de la RAM employé par le TOS. Il sert comme départ de la zone TPA. (end_os est copié dans _membot).

**exec_os** **$4Fe (long)**

Pointe sur le 'shell' qui reçoit la main du BIOS après l'initialisation du système. Normallement pointe sur le premier octet de la zone de code AES.

N.D.T: Nouveaux vecteurs fournis avec les réserves d'usage

| | |
|---|---|
| $502 | vecteur routine de copie d'écran |
| $506 | vecteur routine de test de sortie parallèle |
| $50A | vecteur routine de sortie parallèle |
| $50E | vecteur routine de test de sortie série |
| $512 | vecteur routine de sortie série |

Ces vecteurs ne sont pas implantés sur les versions du TOS antérieures au 24/4/86.





## INFORMATIONS POST MORTEM

Si une cartouche de diagnostic n'est pas connectée, tous les vecteurs d'interruption non utilisés pointent sur une routine du BIOS qui sauve l'état du processeur dans le bas de la mémoire (voir plus bas) et affiche un certain nombre d'icônes (bombes) au centre de l'écran. Le module essaie de faire repartir le système après le crash. Ce n'est pas toujours (loin s'en faut) couronné de succès.

Le nombre d'icônes est fonction de l'exeption qui s'est produite (2 pour une erreur de bus, 3 pour une erreur d'adresse, etc.. voir le 'Processus d'Exception' dans manuel 68000 de Motorola).

L'état du processeur est sauvé dans une zone de mémoire qui n'est pas touchée par le reset du système. Il est ainsi possible d'examiner cette zone après le reset pour repartir.

```
*
*    Zone de sauvegarde de l'état
*    du processeur après une exception non récupérée.
*
     proc_lives     equ  $380      * $12345678 si valid
     proc_dregs     equ  $384      * sauvegarde de d0-d7
     proc_aregs     equ  $3a4      * a0-a6, a7 super (ssp)
     proc_enum      equ  $3c4      * numéro d'exception (un octet)
     proc_usp       equ  $3c8      * a7 utilisateur
     proc_stk       equ  $3cc      * 16 mots extraits de ssp
```

Si le long mot d'adresse $380 est $12345678, les informations suivantes sont valides (à moins qu'elles n'aient été recouvertes par un autre 'crash').

D0-D7, A0-A7, et l'adresse de la pile superviseur A7 sont recopiés de l'adresse $384 à $3c0. Le numéro d'exception (2 pour une erreur bus, etc..) est placé dans l'octet d'adresse $3c4. L'adresse de la pile utilisateur est placée en $3c8 et les seizes premiers mots de la pile superviseur sont recopiés dans le seizes mots commençants en $3cc.

## 7. SUPERVISEUR EN GEMDOS

DRI ne s'est pas encore occupé de documenter cette fonction alors...

Lorsque vous lirez ces lignes, gardez à l'esprit que l'idée de départ était de fournir une méthode utilisable en C. Il serait maladroit de s'en servir en assembleur.

Il existe un moyen de passer (ou de sortir) en mode superviseur grace au GEMDOS : C'est la fonction Trap 1, numéro 32 (0x20).

```
LONG _super(pile)
LONG pile;
```

Si 'pile' est -1 (0xFFFFFFFF), la fonction retourne dans D0.L soit 0, indiquant que le processeur est en mode utilisateur, soit 1, indiquant que le processeur est en mode superviseur.

Si 'pile' est différent de -1:

Si la fonction est appelée alors que le processeur est en mode utilisateur, le GEMDOS reviendra avec le processeur en mode superviseur. L'ancienne valeur de la pile utilisateur est retournée dans D0.L. Si 'pile' était NULL (0x00000000), la pile surperviseur sera la même que la pile superviseur avant l'appel, sinon la pile superviseur sera positionnée à 'pile'.
Si la fonction est appelée alors que le processeur est en mode superviseur, le GEMDOS reviendra en mode utilisateur. 'pile' devra être la valeur de la pile superviseur retournée lors du premier appel de la fonction.

L'ancienne valeur de la pile superviseur DOIT TOUJOURS être restituée avant la fin d'un programme, sinon le système plantera.





Exemple d'emploi en C:

```
supermachin()
     {
     long pile_ssp;
     long trap1();

     /* passage en mode superviseur : */

          pile_ssp = trap1(0x20,0L);

          ... partie à faire en mode superviseur...

     /*   retour en mode utilisateur et restitution de
         l'ancienne pile superviseur */

          trap1 (0x20,pile_ssp);
     }
```

Et en assembleur :

```
* supermachin  fait un aller_retour en mode superviseur

supermachin:
     .
     . codes en mode utilisateur
     .
     clr.l     -(sp)              * garde la pile utilisateur
     move.w    #$20,-(sp)         * passe en superviseur
     trap      #1                 *
     addq.l    #6,sp              * nettoie la pile
     move.l    d0,pile_ssp        * sauve l'ancienne pile ssp
     .
     .
     . opérations à faire en superviseur
     .
     .
     move.l    pile_ssp,-(sp)     * empile l'ancienne ssp
     move.w    #$20,-(sp)         * repasse en utilisateur
     trap      #1                 *
     addq.l    #6,sp              * nettoie la pile
     .
     .
     .

bss
     pile_ssp  ds.l 1
```

## 8 LES NUMEROS D'ERREUR

Certaines fonctions BIOS et la plupart des fonctions GEMDOS retournent un numéro d'erreur. Il est à noter que certaines fonctions GEMDOS ne retournent pas un long mot comme code d'erreur mais un MOT (les bits 16 à 31 de D0.L sont ignorés).

Tous les numéros d'erreur sont négatifs. Il existe deux plages d'erreurs : Les erreurs BIOS rangées de -1 à -31 et les erreurs GEMDOS rangées de -32 à -127.

0 (OK)
action couronnée de succès. (l'anti-erreur).

-1 (ERREUR)
Erreur générale.

-2 (LECTEUR PAS PRET)
le périphérique n'est pas prêt, ou pas branché, où est occupé .

-3 (COMMANDE INCONNUE)
le périphérique ne reconnait pas cette commande.

-4 (ERREUR CRC)
erreur soft durant la lecture d'un secteur.

-5 (MAUVAISE REQUETE)
le périphérique ne peut exécuter une commande par suite d'un mauvais contexte ou à cause de mauvais paramètres.

-6 (ERREUR POSITION)
impossible de se positionner sur un disque.

-7 (MEDIUM INCONNU)
essai de lire un médium inconnu. Signifie habituellement que le secteur boot est détérioré ou est inexistant.

-8 (SECTEUR INTROUVABLE)
le secteur ne peut être trouvé.





-9 (PAS DE PAPIER)
l'imprimante est à cours de papier.

-10 (ERREUR D'ECRITURE)
échec lors d'une opération d'écriture.

-11 (ERREUR DE LECTURE)
échec lors d'une opération de lecture.

-12 (ERREUR GENERALE)
réservé pour de futures catastrophes.

-13 (ECRITURE PROTEGEE)
essai d'écriture sur une disquette protégée ou sur un média ouvert en lecture seule.

-14 (CHANGEMENT DE MEDIUM)
le médium a changé depuis la dernière écriture; l'opération (lecture ou écriture) n'a pas eu lieu. C'est plus un message qu'une réelle erreur.

-15 (PERIPHERIQUE INCONNU)
le BIOS ne connait pas le périphérique sur lequel une opération a été demandée.

-16 (MAUVAIS SECTEUR)
l'opération de formatage a reussie mais des secteurs sont défecteux.

-17 (INSERER UNE DISQUETTE)
demande à l'utilisateur d'introduire une disquette.C'est, en fait, un message à l'adresse du shell (GEM ou COMMAND.PRG) pour débuter un dialogue avec l'utilisateur.

N.D.T: Voir le fichier inclus Tosdefs.h

## NUMEROS D'ERREUR

Codes Erreur GEMDOS
(les nombres entre parenthèses sont les numéros d'erreur équivalents MSDOS)

| nom | numéro | description |
|---|---|---|
| EINVFN | -32 (1) | Numéro de fonction invalide |
| EFILNF | -33 (2) | Fichier introuvable |
| EPTHNF | -34 (3) | Chemin introuvable |
| ENHNDL | -35 (4) | Plus d'identificateur disponible |
| EACCDN | -36 (5) | Accès interdit |
| EIHNDL | -37 (6) | Identificateur invalide |
| ENSMEM | -39 (8) | Mémoire insuffisante |
| EIMBA | -40 (9) | Adresse de bloc mémoire invalide |
| EDRIVE | -46 (15) | Spécification de disque invalide |
| ENMFIL | -49 (18) | Plus de fichier |
| ERANGE | -64 | Erreur de champ |
| EINTRN | -65 | Erreur interne GEMDOS |
| EPLFMT | -66 | Format de fichier exécutable invalide |
| EGSBF | -67 | Echec d'accroissement de bloc mémoire |





## 9. LE SUPPORT CARTOUCHE

Il existe deux types de cartouches: les 'cartouches d'Application' qui sont reconnues par le GEM et le bureau et les 'cartouches de diagnostic' qui sont exécutées presqu'immédiatement après le reset du système (avant que le 68000 ne touche la RAM), et peuvent prendre en main la totalité du système.

La zone matérielle réservée aux cartouches s'étale sur 128K et va de $FA0000 à $FBFFFF. Le long mot contenu à l'adresse $FA0000 a une signification particulière pour le TOS. Il doit être l'une des valeurs suivantes :

$FA52255F — indique qu'une cartouche de diagnostic à été introduite.

$ABCDEF42 — indique qu'une cartouche d'application à été introduite.

Toutes les autres valeurs sont ignorées.

Si une cartouche de diagnostic est branchée lors du RESET du système le TOS fera presqu'immédiatement un saut à l'adresse $FA0004. A6 contiendra l'adresse de retour (au cas où la cartouche voudrait continuer avec l'initialisation du système). Le pointeur de pile contient n'importe quoi. La plupart des registres materiels du ST n'ont pas encore été touchés. Le registre le plus important est le contrôleur de mémoire. La cartouche de développement a la charge de dimensionner la mémoire et d'initialiser son contrôleur.

Les cartouches d'application doivent fournir un 'en-tête d'application' à l'adresse $FA0004 (immédiatement après le long mot magique). L'en-tête d'application contient des renseignements sur le programme en ROM.

En-tête de cartouche d'application :

| | | |
|---|---|---|
| CA_SUIV | 0 | -> prochain en-tête |
| CA_INIT | 4 | -> codes d'initialisation |
| CA_RUN | 8 | -> codes d'exécution |
| CA_HEUR | $c | heure en format DOS |
| CA_DATE | $e | date en format DOS |
| CA_TAIL | $10 | taille du programme |
| CA_NOM | $14 | nom du programme (NNNNNNNN.EEE) |

CA_SUIV pointe sur l'en-tête suivant. Si CA_SUIV est $00000000, il n'y a pas d'autres en-tête.

CA_INIT pointe sur le code d'initialisation de l'application. Si CA_INIT est NULL, il n'y a pas de code d'application. Le vecteur d'initialisation est appelé au démarrage du système, selon les bits de contrôle contenus dans le mot haut de ce long_mot (voir plus bas).

CA_RUN pointe sur l'entrée du programme.

CA_HEURE ET CA_DATE sont l'heure et la date, en format DOS, de création. (Ils n'ont pas d'autres intérêt que de conserver une trace du numéro de version).

CA_TAIL est la taille du programme.

CA_NOM est une chaîne terminée par zéro contenant le nom du programme. Il doit être au format DOS des noms de fichiers, sans chemin. (jusqu'à huit caractères de tête, qui peuvent être suivis par un point et jusqu'à trois caractères d'extension, et terminé par un NUL (00) ).





Les huits bits les plus hauts (24 à 31) de CA_INIT ont une signification particulière:

| bit | signification |
|---|---|
| 0 | positionné pour exécuter un programme, (par l'intermédiaire de CA_INIT) avant que les vecteurs d'interruption, la mémoire écran, etc.., n'aient été initialisés. |
| 1 | positionné pour exécuter un programme juste avant l'initialisation du GEMDOS |
| 2 | (inutilisé) |
| 3 | positionné pour lancer une application, juste avant le 'boot' disque. (ACTUELLEMENT n'est utilisable que pour un 'boot' an ROM |
| 4 | (inutilisé) |
| 5 | positionné si l'application est un accessoire de bureau |
| 6 | positionné si le programme n'est pas une application GEM. Il tourne sous TOS et ne fait aucun appel à l'AES |
| 7 | positionné si le programme est une application TOS avec paramètres. |

## 10. L'INITIALISATION DU SYSTEME EN ROM

1. Le PC (compteur de programme) est placé en $FC0000, la pile SP en $FC0004.

   Exécute le RESET du système. Le niveau de priorité d'interruption (IPL) du processeur est monté à 7 et une intruction RESET est exécutée pour réinitialiser les registres matériels.

   Si une cartouche de diagnostic est connectée, une adresse de retour est placée dans A6 et le pointeur de programme saute aux intructions de la cartouche.

2. Si la mémoire a déjà été installée (si c'est un départ à chaud) le contrôleur de mémoire est initialisé.

3. Si le vecteur de reset à chaud est valide, une adresse de retour est placée dans A6 et le programme saute à la routine de reset.

4. Initialise le PSG (désélectionne les lecteurs de disquettes), fixe la fréquence de balayage (50 ou 60Hz), écrit les valeurs par défaut dans la palette de couleur, et positionne le pointeur d'écran à 0x10000.

Si la mémoire a déjà été dimensionnée dans un reset antérieur, passe à l'étape numéro 8.

5. Prend les dimensions des deux bancs de mémoire.

6. Exécute un test de mémoire.

7. Une fois la mémoire dimensionnée et mise à zéro, enregistre le fait en mettant deux mots 'magiques' dans le bas de la mémoire.





8. Nettoie les 64K du bas de la mémoire, de 'endosbss' à 0xffff. Initialise toutes sortes de variables du système. Installe les vecteurs d'interruption. Appelle le point d'entrée du programme d'initialisation de la série BIOS.

9. Exécute les programmes de type 'deux' de la cartouche d'application

10. Initialise la résolution de l'écran.

11. Exécute les programmes de type 'zér'o de la cartouche d'application.

12. Autorise les interruptions (toutes sauf HBLANK), en portant l'IPL à 3.

13. Exécute les programmes de type 'un' de la cartouche d'application.

14. Appelle la routine d'initialisation du GEMDOS.

15. Tente de lancer un 'boot' à partir du lecteur de disquette, si la variable 'bootdev' est plus petite que deux et si un lecteur de disquette est branché.

16. Exécute les programmes de type 'trois' de la cartouche d'application

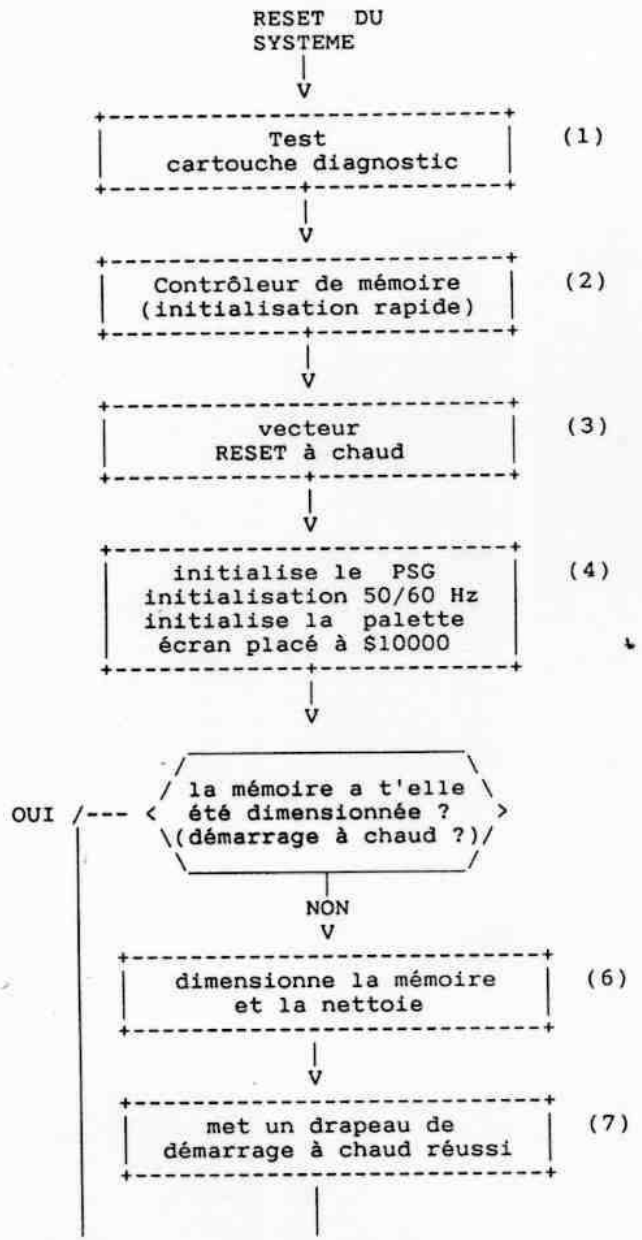
17. Essaie de charger un secteur boot à partir du bus DMA. Pour chacun des huit périphériques du bus DMA, une tentative de lecture est faite sur le secteur logique 0. Si la lecture est un succès et si le 'checksum' de secteur égale $1234, le secteur est exécuté. (voir la partie " Boot du bus DMA")

    TOUS les disques durs sont testés. Si le programme de boot revient, le BIOS essaiera de charger les secteurs boot des autres disques.

18. Met en marche le cucseur. Exécute \AUTO*.PRG. Essaie d'exécuter le COMMAND.PRG.

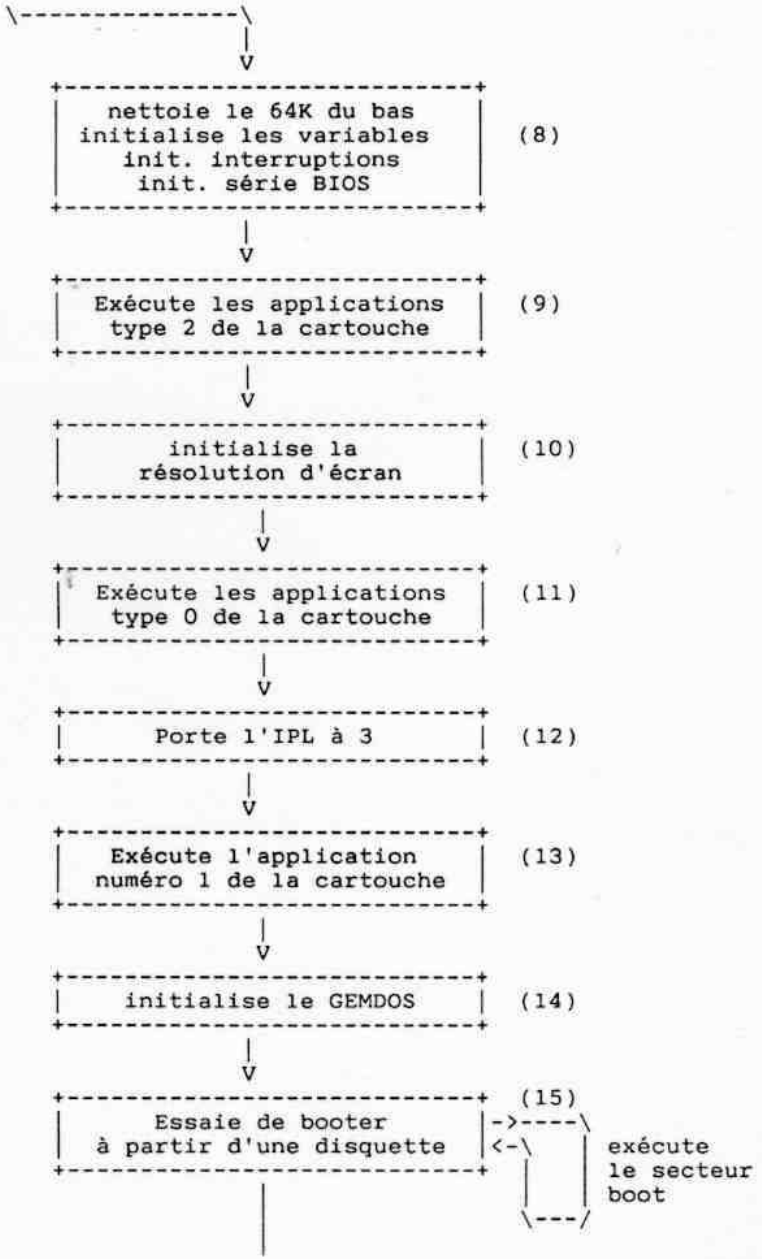
19. Exécute \AUTO*.PRG. Saisit la chaîne d'environnement. Exécute l'AES (en ROM).

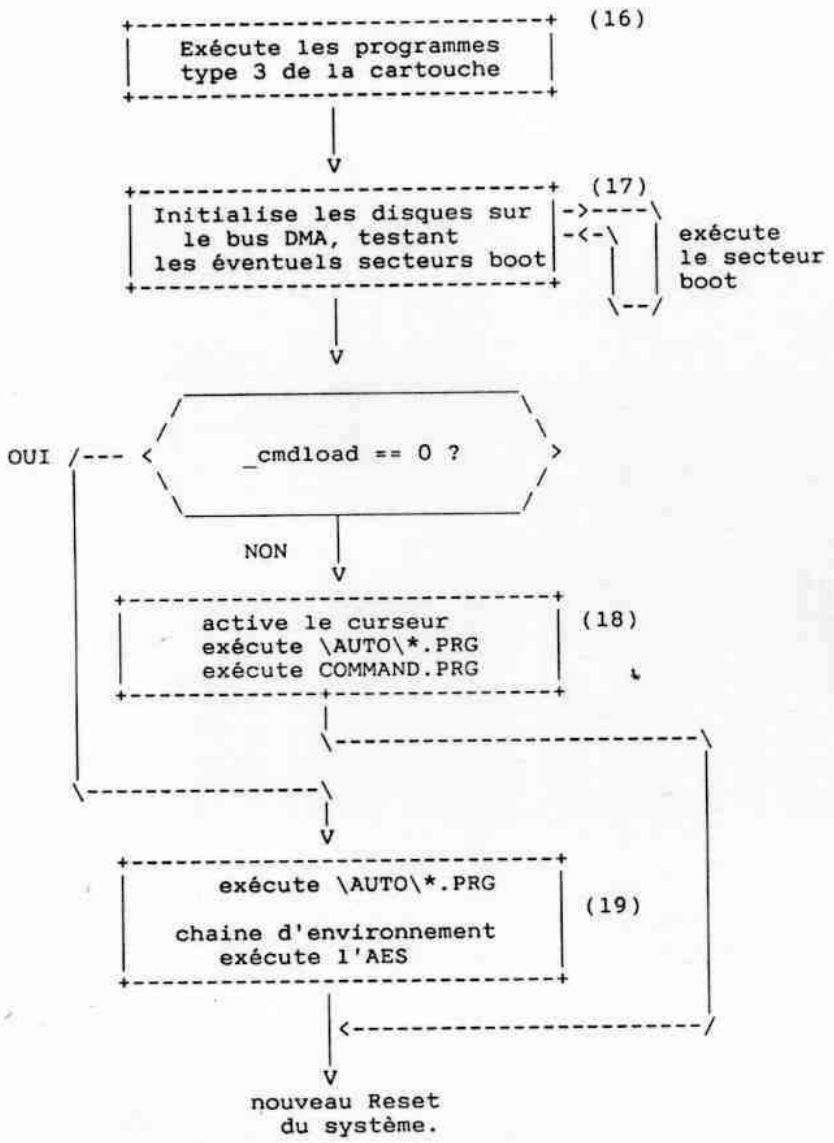
20. Si les opérations (18) ou (19) ont été menées à bien, le système est réinitialisé en retournant à la procédure numéro (1). (En principe le système 'reste dans l'AES')


RESET DU SYSTEME
Test cartouche diagnostic
(1)
Contrôleur de mémoire (initialisation rapide)
(2)
vecteur RESET à chaud
(3)
initialise le PSG
initialisation 50/60 Hz
initialise la palette
écran placé à $10000
(4)
la mémoire a t'elle été dimensionnée ? (démarrage à chaud ?)
OUI
NON
dimensionne la mémoire et la nettoie
(6)
met un drapeau de démarrage à chaud réussi
(7)






nettoie le 64K du bas
initialise les variables
init. interruptions
init. série BIOS
(8)
Exécute les applications type 2 de la cartouche
(9)
initialise la résolution d'écran
(10)
Exécute les applications type 0 de la cartouche
(11)
Porte l'IPL à 3
(12)
Exécute l'application numéro 1 de la cartouche
(13)
initialise le GEMDOS
(14)
Essaie de booter à partir d'une disquette
(15)
exécute le secteur boot



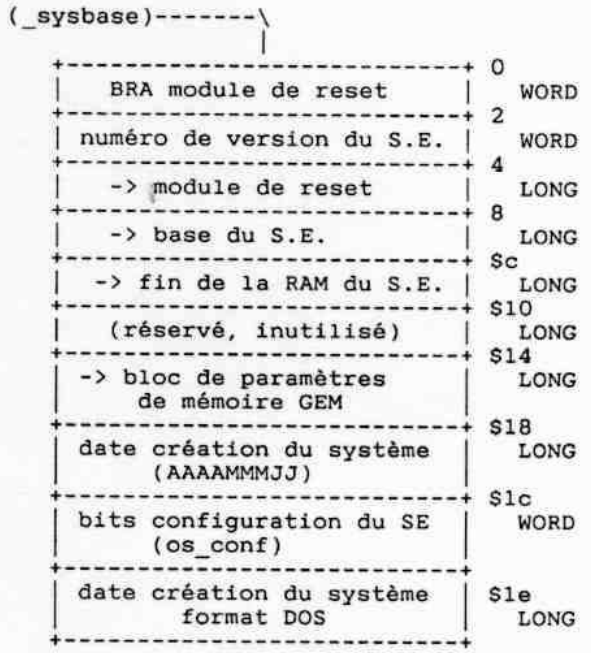





## 11. EN-TETE DE MEM (ROM)

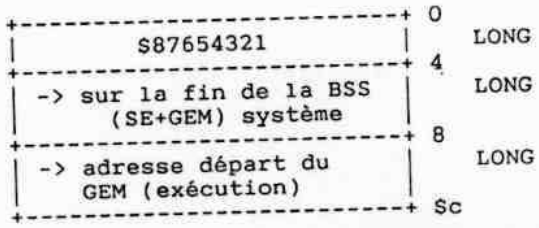
La variable système '_sysbase' ($4F2) pointe sur la base du système d'exploitation qui peut être en ROM (si '_sysbase' est plus grand que 'phystop') ou en RAM.

La base du système d'exploitation est structurée comme suit :




Le bloc de paramètres d'utilisation mémoire GEM, appelé ci_dessous 'le magic', informe le système d'exploitation des exigences en mémoire du GEM, et de l'adresse de départ du GEM.

Structure du 'magic':

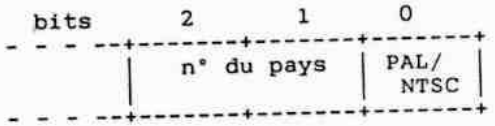



L'en-tête du SE (système d'exploitation) pointe sur le 'magic'. Le bloc de paramètres 'magic' n'est validé que si le premier long mot est $87654321. Le GEM n'est pris en compte que si le 'magic' est valide. De plus, sur un système en RAM, si le 'magic' n'est pas valide, la mémoire habituellement employée par le GEM est incluse dans la zone TPA de départ.

La fonction BIOS étendu $27, puntaes(), teste si le 'magic' est valide. Si le magic n'est pas valide, il revient immédiatement. Sinon il teste si le 'magic' est situé en ROM, et s'il l'est, puntaes() revient. Finalement puntaes() invalide le 'magic' (en mettant à zéro son premier long mot) et saute au module de reset système.

Soit puntaes() reviendra simplement, signifiant que la fonction a déjà été exécutée ou plus précisément que le 'magic' n'était pas valide, soit il détruira le 'magic' et relancera le système d'exploitation. Le GEMDOS n'autorisant pas que la zone TPA soit agrandie après que celui-ci ait été initialisé, le SE doit être relancé.

Le mot de configuration spécifique au pays ('os_conf') a cet aspect:




Les numéros de pays sont :

| | |
|---|---|
| 0 | USA |
| 1 | Allemagne |
| 2 | France |
| 3 | Angleterre |





Le bit zéro signifie NTSC lorsqu'il est à 0 et PAL lorsqu'il est à 1. Le registre matériel de synchronisation 'syncmode' est positionné en conséquence à l'initialisation du système. Les bits de pays peuvent être étendus par la suite.

Le numéro de version est $0000 pour le boot ROM, et différent de zéro pour le système entier en ROM. Le format de ce mot de version est $VVRR (VV = numéro de version, RR = numéro de release). Le premier système en ROM a le numéro de version $0100.

L'en-tête contient plusieurs dates, dans des formats différents. Le premier, plus facilement lisible humainement, est un long mot en hexadécimal décomposable ainsi :$AAAAMMJJ (AAAA pour l'année, MM pour le mois, JJ pour le jour). Le second est au format DOS.

DECHARGE

Atari ne promet pas que les numéros de version, dans les mises à jour futures du système, refléteront la réalité. Nous pouvons faire des versions corrigées sans pour cela changer le numéro de version, et, à contrario, nous pouvons changer le numéro de version sans modifier le système.
Autrement dit, n'écrivez pas de programmes dépendant du numéro de version du système.

## 12. LE SECTEUR DE DEPART (BOOT)

Le secteur de départ appelé secteur 'boot' contient:

- Un numéro de série
- Un bloc de paramètres BIOS
- D'éventuels code de 'boot' et paramètres de 'boot'

Un secteur 'boot' exécutable doit obligatoirement avoir un 'checksum' de mot égal à $1234. Lors de l'initialisation le secteur boot est chargé du disque dans un tampon. Si le 'checksum' est correct le système fait un saut avec retour (JSR) au premier octet du tampon. (L'emplacement du tampon n'étant pas connu, les codes contenus dans le secteur boot doivent être indépendants de leur adresse). Voir la partie sur l'initialisation du système pour plus de détails sur l'écriture d'applications pouvant être lancées au démarrage.

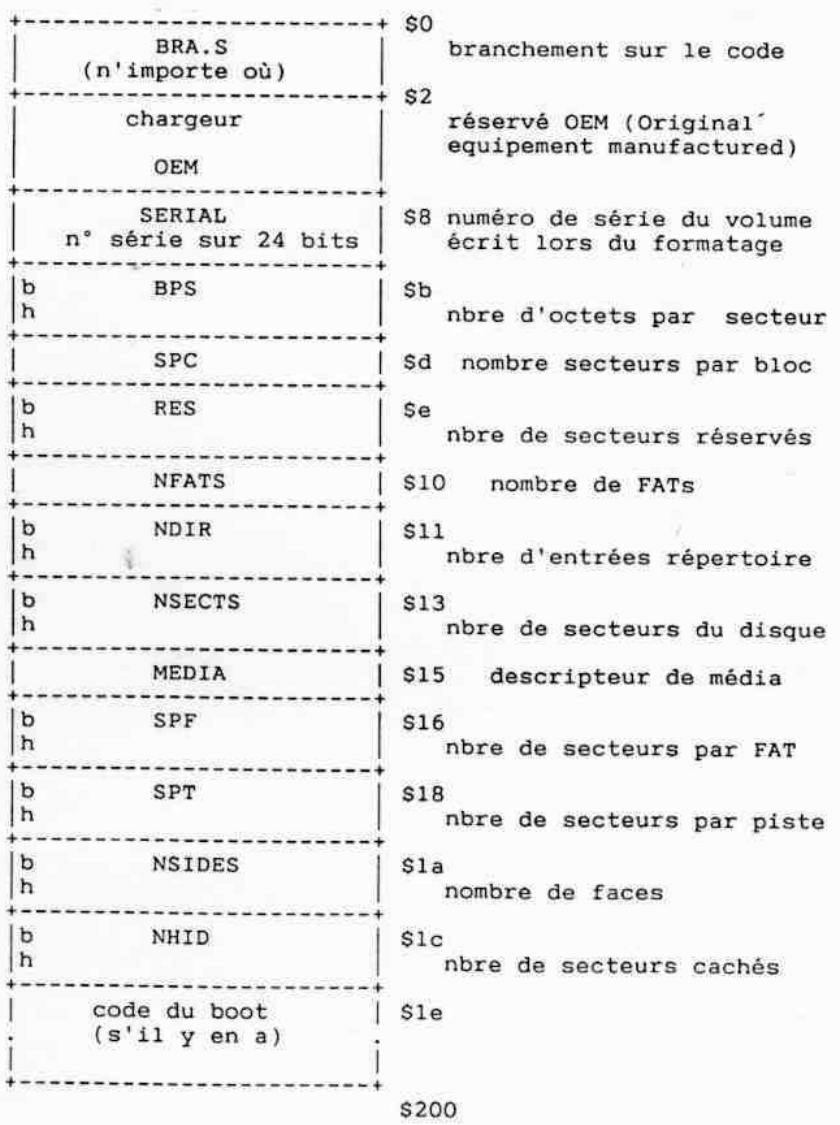
Lors d'un appel 'getbpb', le BIOS lit le secteur 'boot' et examine le prototype de bloc de paramètre BIOS (BPB). Le BPB est construit à partir de ce prototype. S'il semble incorrect, par exemple si un champ critique est à zéro, le BIOS renvoit NULL (0L) (comme indication d'erreur).

Le BPB est habituellement fabriqué et écrit lors du formatage du volume.

Le numéro de série de 24 bits sert à détecter le changement de disque. Le numéro de série est fabriqué et écrit lors de l'exécution de l'utilitaire de formatage. Il est unique.







Le prototype du BPB est compatible soft avec un BPB MS-DOS de la version 2.x. (Ce qui ne veut pas dire que le ST puisse lire des secteurs écrits, ou rendus conformes pour l'écriture, par un contrôleur de disque autre que le WDC 1770/1772.)

. Les octets faibles des champs de 16 bits (mots) dans le BPB (tel que 'BPS') occupent les parties basses des adresses (comme sur le 8086).

BPS représente le nombre d'octets par secteur. (512 pour les disquettes du ST).

SPC est le nombre de secteurs par bloc. (habituellement 2 sur une disquette, totalisant une taille de 1K par bloc).

RES est le nombre de secteurs réservés au début du disque, comprenant le secteur boot. RES est habituellement 1 sur les disquettes.

NFATS est le nombre de tables d'allocation de fichier sur le disque. (habituellement deux)

NDIRS est le nombre d'entrées dans le catalogue.

NSECTS est le nombre total de secteurs sur le disque, les secteurs réservés inclus.

MEDIA est l'octet de descripteur de media. Le BIOS ST ne se sert pas de cet octet, mais d'autres système le peuvent.

SPF est le nombre de secteurs dans chaque FAT.

SPT est le nombre de secteurs par piste.

NSIDES est le nombre de faces sur le disque. (Les disquettes simple face peuvent être lues avec un lecteur double face, mais le contraire n'est pas possible.)

NHID est le nombre de secteurs cachés. (Habituellement le ST ignore cette valeur pour les disquettes).

Le dernier mot du secteur boot (à l'offset $1FE) est réservé pour équilibrer le checksum. En particulier la fonction BIOS '_protobpb ()' modifie ce mot.





CODE DE 'BOOT' DU BUS DMA

Ce code, issu du BIOS ST, essaie de charger des secteurs 'boots' à partir de périphériques branchés sur le bus DMA. Le code peut servir comme:

- Un exemple d'emploi du bus DMA (utile pour l'écriture de secteurs 'boot' et de 'drivers' de périphériques).
- Un fournisseur d'informations sur le Timing et les caractéristiques de commande attendus par un périphérique qui peut être 'booté' sur le bus DMA.

```
gpip      equ  $fffffa01      * (B) registre d'entrée du 68901
diskctl   equ  $ffff8604      * (W) accès données contrôleur
                              * de disque
fifo      equ  $ffff8606      * (W) contrôle de mode DMA
dmahigh   equ  $ffff8609      * (B) haut de la base du DMA
dmamid    equ  $ffff860b      * (B) milieu de la base du DMA
dmalow    equ  $ffff860d      * (B) bas de la base DMA
flock     equ  $43e           * (W) variable verrouillage
                              * accès DMA
_dskbufp  equ  $4c6           * (L) -> tampon de disque de 1K
_hz_200   equ  $4ba           * (L) compteur 200hz
bootmagic equ  #$1234


*
*    bootdma     essai de 'booter' à partir d'un périphérique
*    branché sur le bus DMA.
*
*    Paramètres passés :  aucun
*
*    Retour : peut-être jamais...
*
*    Registres utilisés : tous
*
*    Commentaires :
*
*         Essaie de lire les secteurs 'boot' des huit
*         périphériques connectés au bus DMA. Si le secteur est
*         lu et que le checksum est égal à $1234, il est exécuté.
*         Ce code prendra 0.5 secondes d'exécution si rien n'est
*         connecté au bus DMA. Evidemment si quelque chose est
*         connecté, il devrait nous fournir un secteur boot.

bootdma:
```

```
     moveq     #0,d7              * départ avec périph 0
dmb_1:
     bsr       dmaread            * essai de lecture du boot secteur
     bne       dmb_2              * échec -> essai sur disque suivant
     move.l    _dskbufp,a0        * a0 -> tampon disque
     move.w    #$00ff,d1          * checksum $100 mots
     moveq     #0,d0              * checksum = 0
dmb_3:
     add.w     (a0)+,d0           * ajoute le mot (suivant)
     dbra      d1,dmb_3
     cmp.w     #bootmagic,d0      * est-ce un secteur exécutable ?
     bne       dmb_2              * ne fait rien
     move.l    _dskbufk,a0        * a0 -> tampon disque
     jsr       (a0)
dmb_2
     add.b     #$20,d7            * périphérique suivant
     bne       dmb_1              * traite les huit périphériques
     rts

     *
     * litdma  essaie de lire le secteur boot du disque sur le bus
     * DMA
     *
     * paramètre passé : d7.b = ddd00000   'ddd' est le code ASCII
     *                   du périphérique
     *
     * retour :    NE   lecture ratée
     *             EQ   lecture réussie,  données du secteur  dans
     *                  (* _dskbuff)[]
     *
     * registre préservé : d7.w
     *
     * registres utilisés : tous les autres
     *
litdma:
     lea       fifo,a6            * a6-> registre de contrôle du DMA
     lea       diskctl,a5         * a5 -> registre données DMA
     st        flock              * verrouille le DMA contre les
                                  * vblanks
     move.l    _dskbufp,-(sp)     * installe le pointeur DMA
     move.b    3(sp),dmalow
     move.b    2(sp),dmamid
     move.b    1(sp),dmahigh
     addq      #4,sp

     move.w    #$098,(a6)         *  inverse lect/écrit.,laisse  dans
     move.w    #$198,(a6)         * l'état lecture
     move.w    #$098,(a6)
     move.w    #1,(a5)            * registre de compteur de secteur
     move.w    #$088,(a6)         * sélection bus dma (pas SCR)
     move.b    #$d7,d0            * d0.L = n° disque + commande
     or.b      #$08,d0            *  d0.b = n° disque <<5 OR  bits  de
                                  * commande lecture
     swap      d0
```





```
     move.w    #$08a,d0           * d0.L =
     bsr       wcbyte             * xxxxxxxxddd01000xxxxxx010001010
     bne       dmr_q              * temps écoulé

     moveq     #3,d6              * compte = 4
     move.l    #$0000008a,d0      * d0.1 = commande générique ($0000)
dmr_lp
     bsr       wcbyte             * écriture des octets 2,3,4 et 5
     bne       dmr_q              * temps écoulé
     dbra      d6,dmr_lp          * boucle pour plus d'octets
     move.l    #$0000000a,(a5)    * écriture octet 6 (octet de fin)
     move.w    #400,d1            * temporisation de 2.0 sec
     bsr       wwait              * attende d'achèvement
     bne       dmr_q              * temps écoulé
     move.w    #$08a,(a6)         * sélection registre de status
     move.w    (a5),d0            * prend code retour du DMA
     beq       dmr_r              * (retour si OK)

*----reset du DMA, retour NE
dmr_q
     moveq     #-1,d0             * retourne -1 (erreur)
dmr_r
     move.w    #$080,(a6)         * nettoie broche DMA pour disquettes
     tst.b     d0                 * test pour avoir NE au retour
     sf        d0                 * dévérouille la broche DMA
     rts                          * retour

*
* wcbyte  - écrit un octet de commande ACSI, attend pour IRQ
*
* paramètres passés : d0.L  octet de commande et bits de contrôle
*                           16 à 23 = octet de commande
*                           0 à 7 = bits de contrôle de FIFO
*                           a5 -> $ff8604
* retour : NE en cas d'échec
*          EQ en cas de succès ACK
*
* registre utilisé : d1
*
wcbyte:
     move.l    d0,(a5)            * écrit WDC,WDL
     moveq     #10,d1             * temporise 1/20ème de sec.
wwait:
     add.l     _hz_200,d1         * moment où quitter
ww_1
     btst.b    #5,gpip            * commande enregistrée ?
     beq       ww_w               * oui, retour
     cmp.l     _hz_200,d1         * temps écoulé ?
     bne       ww_1               * pas encore, nouvel essai
     moveq     #-1,d1             * s'assure que NE (erreur temps)
ww_w:
     rts                          * retour
```

## 13. LE FORMATAGE D'UNE DISQUETTE

1. Servez vous de la fonction BIOS étendu $A,'flopfmt(), pour formater toutes les pistes d'une disquette. Si les pistes 0 ou 1 ont un secteur détérioré la disquette sera inutilisable.

   Le format standard du ST est ainsi:

   1 ou 2 faces,
   80 pistes,
   9 secteurs par pistes,
   pas d'entrelacement (secteurs séquentiels).

   Mettre à zéro les deux premières pistes (ce qui mettra à zéro les secteurs de FAT et de catalogue).

2. Appeler la fonction BIOS étendu $12 ,'protobt()' pour créer un secteur boot. Le paramètre 'disktype' devra être 2 ou 3 selon que la disquette 80 pistes a une ou deux faces. Le numéro de série 'serialno' sera un nombre aléatoire (<= $1000000).

   Le paramètre d'exécution 'execflag' devra être zéro à moins que le tampon de prototype ne contienne du code (par exemple un chargeur) qui devra être exécuté au 'boot' de la disquette.

3. Ecrire le secteur 'boot',(fabriqué dans le buffer par l'opération numéro (2) ), sur la piste 0, face 0, secteur 1 de la nouvelle disquette. NE PAS se servir de la fonction 'rwabs'; utiliser la fonction BIOS étendu $9 'flopwr'.

Il est tout à fait possible de créer des disques ayant un format spécial en faisant varier le nombre de secteurs par piste, en formatant quelques pistes en plus, ou en spécifiant des facteurs d'entrelacement étranges.





Les codes employés par l' "écriture de piste" du 1772 lors du formatage d'une piste sont:

| Nombre | octet | utilisation |
|---|---|---|
| 60 | $4e | début de piste |
| pour chaque secteur: | | |
| 12 | $00 | |
| 3 | $f5 | écrit $a1 |
| 1 | $fe | marque d'adresse ID |
| 1 | n° piste | 0 à $4f |
| 1 | n° face | 0 ou 1 |
| 1 | n° sect. | 1 à 9 |
| 1 | $02 | 512 octets par secteur |
| 1 | $f7 | 2 CRCs écrits |
| 22 | $4e | |
| 12 | $00 | |
| 3 | $f5 | écrit $a1 |
| 1 | $fb | marque d'adresse de données |
| 512 | xx | données vierges |
| 1 | $f7 | 2 CRCs écrits |
| 40 | $4e | |
| Fin de piste | | |
| 1401 | $4e | remplissage de fin de piste |

## 14. PARTITION DU DISQUE DUR

Le premier secteur, (secteur logique 0), d'un disque dur contient les informations de partition.

| | offset |
|---|---|
| hd_siz | $1c2 |
| p0_flg | $1c6 |
| p0_id | $1c7 |
| p0_st | $1ca |
| p0_siz | $1ce |
| p1_flg | $1d2 |
| p1_id | $1d3 |
| p1_st | $1d6 |
| p1_siz | $1da |
| p2_flg | $1de |
| p2_id | $1df |
| p2_st | $1e2 |
| p2_siz | $1e6 |
| p3_flg | $1ea |
| p3_id | $1eb |
| p3_st | $1ee |
| p3_siz | $1f2 |
| bsl_st | $1f6 |
| bsl_cnt | $1fa |
| (réservé) | $200 |

N.D.T:

Les octets $18c à $1c2 servent au programme de lancement du disque dur (AHDI.PRG). Il est déconseillé de les modifier, de même que les suivantes naturellement.

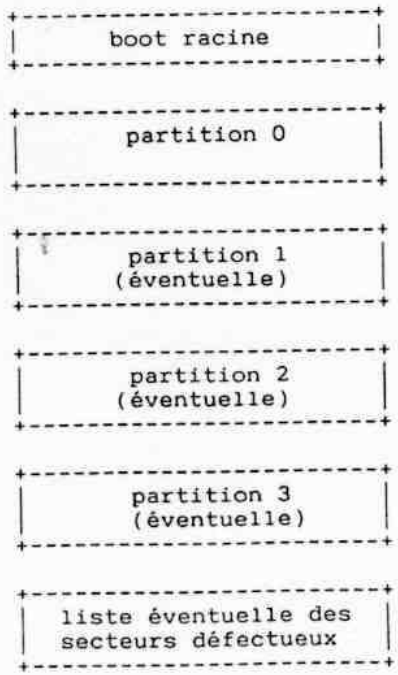




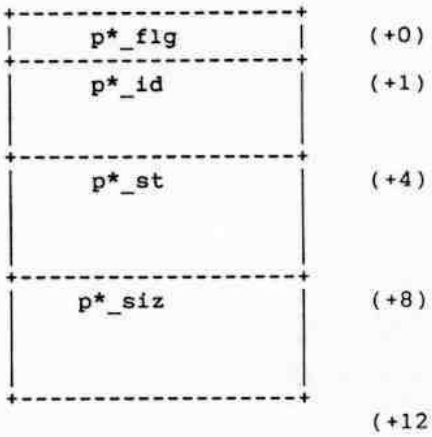
'hd_siz' est la taille totale du disque en secteurs logiques. Le numéro du premier secteur de la liste des secteurs défectueux se trouve dans 'bsl_st'. Habituellement la liste des secteurs défectueux se trouve à la fin du disque.

'bsl_cnt' est le nombre de secteurs défectueux. Chaque secteur défectueux est représenté par un long mot contenant le numéro du secteur. Le nombre total de secteurs défectueux est égal à 'bsl_siz'/ 4.

Un disque peut contenir jusqu'à quatre partitions. Le premier secteur d'une partition est le secteur 'boot', qui sur le ST contient un BPB.



Chaque partition est décrite par une structure de 12 octets:




'p*_flg' doit être différent de zéro pour indiquer que la partition existe. Le BIOS 'bootera' la première partition qui a le bit 7 de cet octet à 1.

'p*_id' est un champ de 3 octets qui identifie la partition. Pour les partitions GEMDOS, le champ contient les 3 caractères ascii "GEM".

'p*st' spécifie le numéro du secteur logique de début de partition.

'p*siz' spécifie la taille de la partition en secteurs logiques.





## 15. LE FORMAT DU SECTEUR DE DEMARRAGE AUTOMATIQUE

Le secteur de démarrage automatique, 'Loader' ou Chargeur, est un chargeur système type. Il se trouve sur le secteur boot, et est chargé en RAM et exécuté pendant l'initialisation du système. Le Chargeur a la possibilité de charger un fichier 'image' ou une suite de secteurs contigus à partir du disque.

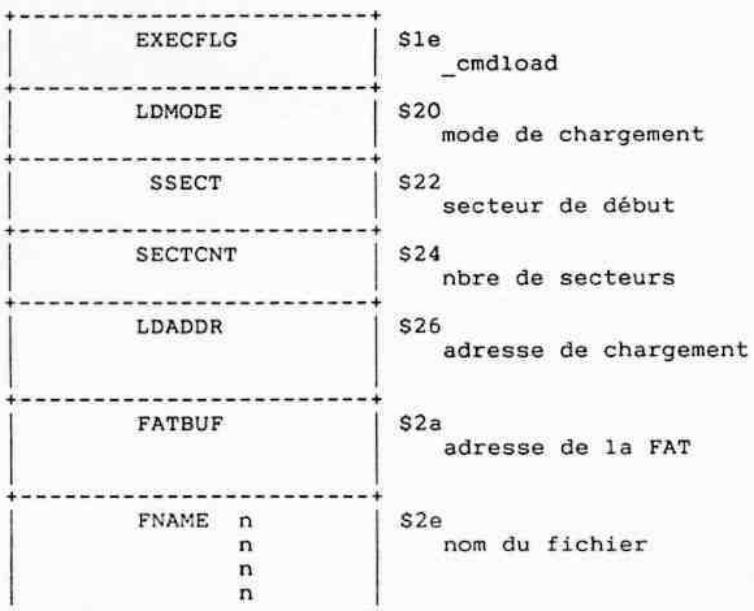
Les six octets réservés à partir du deuxième octet du secteur boot doivent être :

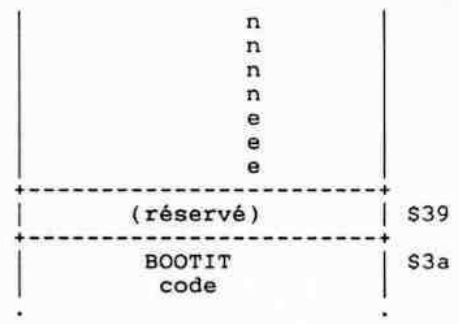
'Loader'

pour permettre à certains outils de manipuler les secteurs de démarrage.

Un fichier 'image' ne contient ni en-tête, ni informations de relocation. C'est une exacte image du programme à exécuter. Le Chargeur est capable de charger n'importe quel fichier d'un disque, quelle que soit sa position dans le répertoire, qu'il soit contigu ou non.

Les informations du Chargeur suivent immédiatement le BPB dans le secteur boot.






EXECFLG est un mot qui est copié dans '_cmdload'.

Si LDMODE est à zéro, un fichier est recherché et chargé. Si LDMODE est différent de zéro, 'SECTCNT' secteurs, débutant au secteur logique 'SSECT', sont chargés à partir du disque.

SSECT est le numéro de secteur logique où commencer le chargement (valide si LDMODE différent de zéro).

SECTCNT est le nombre de secteurs à charger (valide si LDMODE différent de zéro).

LDADDR est l'adresse de chargement du fichier (ou des secteurs).

FATBUF pointe sur l'emplacement de la FAT et des secteurs catalogues.

FNAME est le nom du fichier à charger (valide si LDMODE différent de zéro). Huit caractères de nom et trois d'extension.

(Voir aussi: documentation sur l'utilitaire BOOTGEM.)





## SEQUENCE DE DEMARRAGE

1. Le secteur de démarrage 'boot' est chargé. Le Chargeur prend le contrôle du système.

2. Le catalogue du disque 'booté' et le tampon de la deuxième FAT sont écrits en mémoire, à partir de '_membot'. Le Chargeur recherche un fichier (habituellement) appelé TOS.IMG. S'il n'est pas trouvé, il revient avec un code erreur dans D0.

3. TOS.IMG est écrit en mémoire, à partir de l'adresse $40000.

4. Le contrôle est passé au premier octet de TOS.IMG.

TOS.IMG se compose de trois parties:

1. Un programme de relogement (RELOCRL) qui déplace TOS.IMG à l'emplacement mémoire où il doit être exécuté. RELOCRL prend le contrôle du système, efface l'écran, exécute une copie rapide de bloc, et donne le contrôle au premier octet du système d'exploitation.

2. Une image du système d'exploitation (plus ou moins 90K)

3. Une image du bureau et le GEM (plus ou moins 110K)

L'initialisation du système se passe normalement (sauf pour le nettoyage de mémoire), à partir du moment où le TOS a pris la main.

**MEM DE BOOT**

La ROM de 'boot' du ST ne contient qu'une petite partie du BIOS. Les seules fonctions valides ont rapport à la lecture sur disquette.

L'initialisation du système suit la même procédure que pour un TOS normal. Cependant, les emplacements et les significations des variables systèmes peuvent avoir changés.

Le cours normal des événements est :

La ROM boot exécute le RESET et initialise le système. Il affiche un joli dessin en couleur

Un essai de 'boot' est fait à partir des deux lecteurs. 'bootdev' contiendra le numéro du disque en cas de chargement réussi du secteur boot. (Un jour il existera peut être une version de ROM de boot qui prendra en compte les disques durs).

Le secteur de démarrage est exécuté (voir aussi: 'Secteur de démarrage automatique').

Le numéro de version de la ROM de boot (le second mot de la ROM, en $fc0002) est $0000.

Les fonctions BIOS trap 13 sont:

| fonct. | noms(voir spécifications GEMDOS) |
|---|---|
| 0 | inutilisé |
| 1 | inutilisé |
| 2 | inutilisé |
| 3 | inutilisé |
| 4 | rwabs (en lecture seulement) |
| 5 | inutilisé |
| 6 | inutilisé |
| 7 | getpbp |

Les fonctions BIOS étendu trap 14 sont:

| fonct. | noms (voir fonctions BIOS étendu) |
|---|---|
| 0 | inutilisé |
| 1 | ssbrk |
| 2 | inutilisé |
| 3 | inutilisé |





| | |
|---|---|
| 4 | inutilisé |
| 5 | inutilisé |
| 6 | inutilisé |
| 7 | inutilisé |
| 8 | _floprd (lecture de secteurs) |

Le 'boot' utilise les mémoires allant de $10000 à $20000 pour le tampon d'écran. Eviter de charger quelque chose dans cet emplacement lors d'écriture de programmes qui peuvent être 'bootés' (à moins que vous ne preniez possession du système).

## 16. LE RESUME DES FONCTIONS DU GEMDOS

Les fonctions sont appelables par l'intermédiaire du trap #1. Le premier nombre est le numéro du trap (premier mot dans la pile à l'appel du trap). Le nom de la fonction (comme nommé dans OSBIND.H) vient après, suivi des noms de paramètres. Le nombre d'octets de la pile à nettoyer après l'appel du trap (pour les appels en assembleur) se trouvent entre crochets. Les déclarations d'arguments (s'il y en a), se trouvent sur la ligne suivante. Une courte description de la fonction suit.

En théorie, les appels GEMDOS reviennent avec D0.L comme paramètre de retour. En fait il y a des exceptions. Il est préférable d'examiner D0.W pour tester d'eventuelles erreurs au retour de la fonction. De plus, GEMDOS peut retourner quelquefois des numéros d'erreur BIOS (compris entre -1 et -31).

**$00 Pterm0() [2]**

Fini un programme avec 0 comme code de retour.

**$01 Cconin() [2]**

Lit un caractère sur stdin.

**$02 Cconout(chr) [4]**
```
char chr;
```

Ecrit un caractère sur stdout.

**$03 Cauxin() [2]**

Lit un caractère sur AUX:

**$04 Cauxout(chr) [4]**
```
char chr;
```

Ecrit un caractère sur AUX:





**$05 Cprnout(chr) [4]**
```
char chr;
```

Ecrit un caractère sur PRN:

**$06 Crawio(mot) [4]**
```
WORD mot;
```

si (mot == 0x00ff) lit un caractère sur stdin.
si (mot != 0x00ff) l'écrit sur stdout.

**$07 Crawcin() [2]**

Lit un caractère brut sur stdin (sans écho)

**$08 Cnecin() [2]**

Lit un caractère sur stdin, sans écho. Les caractères de contrôle (^S, ^Q, ^C) sont interprétes et ont un effet.

**$09 Cconws(chaine) [6]**
```
char *chaine;
```

Ecrit une chaine finie par zéro sur stdout.

**$0a Cconrs(tamp) [6]**
```
char *tamp;
```

Lit une chaine éditée sur stdin. En entrée tamp[0] contient la taille de la partie réservée aux données de tamp[]. En sortie, tamp[1] contient le nombre de caractères lus dans tamp[]. La partie 'donnée' débute à tamp[2].

**$0b Cconis() [2]**

Retourne -1 (différent de zéro) si un caractère est disponible sur stdin, 0 sinon.

**$0e Dsetdrv(drv) [4]**
```
WORD drv;
```

Sélectionne le disque courant (0 = A; 1 = B; etc). Retourne la carte des disques actifs (bit 0 = A, bit 1 = B, etc...).

**$10 Cconos() [2]**

Retourne -1 (différent de zéro) si l'écran est prêt à recevoir un caractère, 0 s'il est indisponible.

**$11 Cprnos() [2]**

Retourne -1 (différent de zéro) si PRN: est prêt à recevoir un caractère, 0 s'il n'est pas prêt.

**$12 Cauxis() [2]**

Retourne -1 (différent de zéro) si un caractère est disponible sur AUX:, 0 sinon.

**$13 Cauxos() [2]**

Retourne -1 (différent de zéro) si AUX: est prêt à recevoir un caractère, 0 s'il n'est pas prêt.

**$19 Dgetdrv() [2]**

Retourne le numéro du disque par défaut (0=A, etc.)

**$1a Fsetdta(ptr) [6]**
LONG ptr;

Fixe l'adresse de transfert de disque (utilisée par Fsfirts()).

$20 Super(pile) [6]
LONG pile;

Change le processeur de mode. Si 'pile' est -1L, retourne 0 ou 1, selon que le processeur est en mode utilisateur ou superviseur. Si le processeur est en mode utilisateur, passe en mode superviseur et se sert de 'pile' comme pile superviseur ( ou de la valeur de USP si 'pile' est NULL). Si le processeur est en mode superviseur, passe en mode utilisateur, se sert de 'pile' comme pile superviseur et retourne l'ancienne pile superviseur.





**$2a Tgetdate() [2]**

Retourne la date :

| | | |
|---|---|---|
| bits | 0 à 4 | jour 1 à 3 |
| bits | 5 à 8 | mois 1 à 12 |
| bits | 9 à 15 | année 0 à 119 (0 = 1980) |

**$2b Tsetdate(date) [4]**
WORD date;

Fixe la date dans le format décrit ci_dessus.

**$2c Tgettime() [2]**

Retourne l'heure :

| | | |
|---|---|---|
| bits | 0 à 4 | seconde 0 à 58 (par blocs de 2 sec.) |
| bits | 5 à 10 | minute 0 à 59 |
| bits | 11 à 15 | heure 0 à 23 |

**$2d Tsettime(heure) [4]**
WORD heure;

Fixe l'heure dans le format décrit ci_dessus.

**$2f Fgetdta() [2]**

Demande l'adresse du DTA.

**$30 Sversion() [2]**

Demande le numéro de version.

**$31 Ptermres(nbrest,ret) [8]**
LONG nbrest;
WORD ret;

Termine un programme qui reste résident. 'nbrest' est le nombre d'octets à garder dans le descripteur de programme, 'ret' est le code de retour.

**$36 Dfree(tamp,drv)** [8]
```
LONG tamp;
WORD drv;
```

Retourne des indications sur le disque. 'drv' : numéro du disque (0= par défaut, 1=A, 2=B, etc..), 'tamp' pointe sur une structure où seront placés :

```
LONG b_free    nombre de blocs libres
LONG b_total   nombre total de blocs
LONG b_sectail nombre d'octets par secteur
LONG b_bltail  nombre de secteurs par bloc
```

**$39 Dcreate(chem)** [6]
```
char *chem;
```

Crée un catalogue.

**$3a Ddelete(chem)** [6]
```
char *chem;
```

Efface un catalogue.

**$3b Dsetpath(chem)** [6]
```
char *chem;
```

Fixe le répertoire courant.

**$3c Fcreate(nom,attrib)** [8]
```
char *nom;
WORD attrib;
```

Crée un fichier avec le chemin courant. Retourne un identificateur ou un numéro d'erreur (négatif). Les bits d'attributs sont :

```
$01  Lecture unique
$02  Fichier caché au répertoire
$04  Fichier système, caché au répertoire
$08  numéro de volume (les 11 premiers octets)
```

**$3d Fopen(nom,mode)** [6]
```
char *nom;
WORD mode;
```

Ouvre un fichier 'nom' en 'mode' lecture (0), écriture (1) ou lecture/écriture (2). Retourne un identificateur ou un code d'erreur négatif.





**$3e Fclose(id)** [4]
```
WORD id;
```

ferme un fichier.

**$3f Fread(idfich,nbre,tampon)** [12]
```
WORD idfich;
LONG nbre;
char *tampon;
```

Lit 'nbre' octets d'un fichier. Retourne le nombre d'octets lus ou un numéro d'erreur négatif.

**$40 Fwrite(idfich,nbre,tampon)** [12]
```
WORD idfich;
LONG nbre;
char *tampon;
```

Ecrit 'nbre' octets dans un fichier. Retourne le nombre d'octets écrits ou un code d'erreur négatif.

**$41 Fdelete(nom)** [6]
```
char *nom;
```

Efface un fichier.

**$42 Fseek(offset,idfich,mode)** [10]
```
LONG offset;
WORD idfich;
WORD mode;
```

Positionne un pointeur dans un fichier (idfich). 'offset' est le nombre d'octets à partir duquel se positionner. 'mode' peut être :

```
0    à partir du début
1    à partir de la position courante
2    à partir de la fin
```

**$43 Fattrib(chem,mode,attrib) [10]**
```
char *chem;
WORD mode;
WORD attrib;
```

Demande (si 'mode' 0) ou fixe (si 'mode' 1). Les bits d'attributs sont:

| | |
|---|---|
| $01 | lecture seule |
| $02 | fichier caché |
| $04 | fichier système (caché) |
| $08 | nom de volume |
| $10 | sous répertoire |
| $20 | écrit et refermé |

**$45 Fdup(stdid) [4]**
```
WORD stdid;
```

Retourne un identificateur non standard qui pointe le même fichier.

**$46 Fforce(stdid,nonstdid) [6]**
```
WORD stdid;
WORD nonstdid;
```

Force un identificateur standard à pointer le même fichier ou périphérique qu'un identificateur non standard.

**$47 Dgetpath(tamp,lect) [8]**
```
char *tamp;
WORD lect;
```

Retourne le répertoire courant pour le lecteur 'lect' (0=défaut, 1=A, etc..) dans le tampon 'tamp'. 'tamp' doit être d'au moins 64 octets.

**$48 Malloc(compte) [6]**
```
LONG compte;
```

'compte' contient le nombre d'octets à allouer, (ou -1L qui retourne la taille maximum disponible). Retourne un pointeur sur le bloc de mémoire alloué (adresse paire), ou zéro en cas d'échec.

**$49 Mfree(adr) [6]**
```
char *adr;
```

Libère un bloc de mémoire alloué. 0 si libération réussie, différent de zéro en cas d'échec.





**$4a Mshrink(zero,mem,taille) [12]**
```
WORD zero;
LONG mem;
LONG taille;
```

'zero' est un mot à 0. 'mem' l'adresse du bloc mémoire, 'taille' est le nombre d'octets mémoire à conserver. Retourne un nombre différent de zéro en cas d'échec.

**$4b Pexec(mode,nom,comm,envir) [16]**
```
WORD mode;
char *nom;
char *comm;
char *envir;
```

'mode' peut être :

| | |
|---|---|
| 0 | chargement et exécution |
| 3 | chargement seul |
| 4 | exécution seule |
| 5 | création de page de base |

'comm' est une chaîne de commande, recopiée dans la page de base. 'envir' est une chaîne d'environnement. Si 'envir' est NULL, le programme hérite de la chaîne d'environnement de son parent.

Pour le 'mode' 0, le code de retour est le code de retour de l'enfant, ou un nombre négatif système. Si la création de page de base échoue, un nombre d'erreur négatif est retourné.

**$4c Pterm(code) [4]**
```
WORD code;
```

Termine le programme courant. Retourne 'code' à son parent.

**$4e Fsfirst(rech,attr) [8]**

```
char *rech;
WORD attr;
```

'attr' est un jeu d'attributs de recherche (voir fonction $43 pour plus de détails). 'rech' peut contenir de caractères "spéciaux" ('?','*') dans le nom du fichier, mais pas dans le chemin. Retourne 0 si un fichier est trouvé, EFILNF s'il n'est pas trouvé. Place une structure dans le DTA:

octets :

| | |
|---|---|
| 0 à 20 | sans intérêt |
| 21 | attributs du fichier |
| 22 - 23 | heure de création du fichier |
| 24 - 25 | date de création du fichier |
| 26 - 29 | taille du fichier |
| 30 - 43 | nom et extension du ficher |

**$4f Fnext() [2]**

Suite de Fsfirst().

**$56 Frename(zero,anc,nouv) [12]**

```
WORD zero;
char *anc;
char *nouv;
```

Change le nom du fichier d' 'anc' en 'nouv'. 'zero' est réservé et doit être 0.

**$57 Fdatime(idfich,tamp,drap)**

```
WORD idfich;
char *tamp;
WORD drap;
```

'tamp' pointe sur un tampon contenant au retour la date et l'heure du fichier. 'idfich' est l'identificateur du fichier. Si 'drap' est 0, la date et l'heure sont demandées. Si 'drap' est 1, fixe l'heure et la date.





## 17. PRTBLK

NOM

prtblk Imprime un bloc de Bitmap ou de texte sur une imprimante.

SYNOPSIS

```
#include <prtblk.h>

int prtcnt;

int prtblk(args)
PRTARG    *args;
```

DESCRIPTION

PRTBLK est une primitive liée qux périphériques qui imprime un bloc de mémoire sur une imprimante, matricielle monochrome/couleur ou à marguerite monochrome, de la série Atari ST. Prtblk peut être utilisé par n'importe quel programme GEM via l'appel de une fonction trap 14 XBIOS La fonction accepte l'impression de Bitmap (alignement de bits) et de données texte (caractères). La source Bitmap et la destination peuvent être dans différentes résolutions, compatibles entre l'écran hôte du ST et le périphérique d'impression destination. La fonction 'prtblk' est par essence une primitive et de ce fait ne s'occupe pas des transformations d'impression suivantes (Les transformations de Bitmap doivent être faites avant l'appel de 'prtblk') :

- fonctions du niveau GEM VDI pour imprimantes.
- utilitaires de niveau système d'impression d'écran.
- mise à l'échelle horizontale et verticale.
- compensation arbitraire du rapport d'aspect.
- emplacement de la page de données.

Les commandes d'imprimante pour placer la marge de gauche et le retour de ligne peuvent être employées pour positionner la page d'un bloc de données Bitmap. Les paramètres sont passés à 'prtblk' par l'intermédiaire d'un pointeur sur la structure suivante:

```
typedef struct                  /* PRTARG prtblk arguments */
     {
      char     *blkptr;          /* pointeur du bloc */
      unsigned short offset;     /* décalage de bit */
      unsigned short large;      /* largeur */
      unsigned short hautt;      /* hauteur */
      unsigned short gauche;     /* coord.x départ gauche*/
      unsigned short droit;      /* coord. x fin droite */
      unsigned short resecr;     /* résolution source */
      unsigned short resdest;    /* résolution destination*/
      unsigned short *coulpal;   /* -> palette de couleurs*/
      unsigned short type;       /* type d'imprimante */
      unsigned short port;       /* port imprimante */
      char     *masque;          /* ->masques de demi-ton */
     } PRTARG;
```

Tous les paramètres sont garantis contre toute modification. Les paramètres non valides, les valeurs hors normes, et les erreurs de temporisation sont gérés par 'prtblk'. En cas d'erreur la fonction coupe court et revient avec une erreur PBERR (-1), sinon zéro est retourné.

Il est à remarquer que la variable système d'impression 'prtcnt' ($4EE) doit être mise à 1 avant l'appel de 'prtblk'. En fin de routine, en mode Bitmap, 'prtblk' place 'prtcnt' à -1, remet l'espacement de ligne, et replace la couleur à noir (si applicable).

'blkptr' est un pointeur d'octets qui pointe soit sur le premier octet d'un Bitmap, soit sur le premier octet d'un bloc de texte à imprimer. Si le paramètre de hauteur 'haut' est zéro, les données sont interprétées comme des caractères de texte, sinon les données sont interprétées comme un Bitmap de la forme mémoire écran vidéo Atari ST (le bloc d'origine doit être dans le plan 0). Les Bitmaps sont imprimés sur une trame en mode vertical 8 pixels, l'espacement de ligne vertical étant placé au début de chaque passe d'impression. Les chaînes de caractères sont imprimées brutes sur une longueur égale à la valeur du paramètre 'large'.

'offset' est le bit de déplacement de gauche à droite dans le premier octet pointé par 'blkptr' (0 == bit le plus significatif, 7 == bit le moins significatif). Le paramètre de décalage ne s'applique que sur les origines de Bitmaps et est ignoré si le bloc de données est une chaîne de texte.

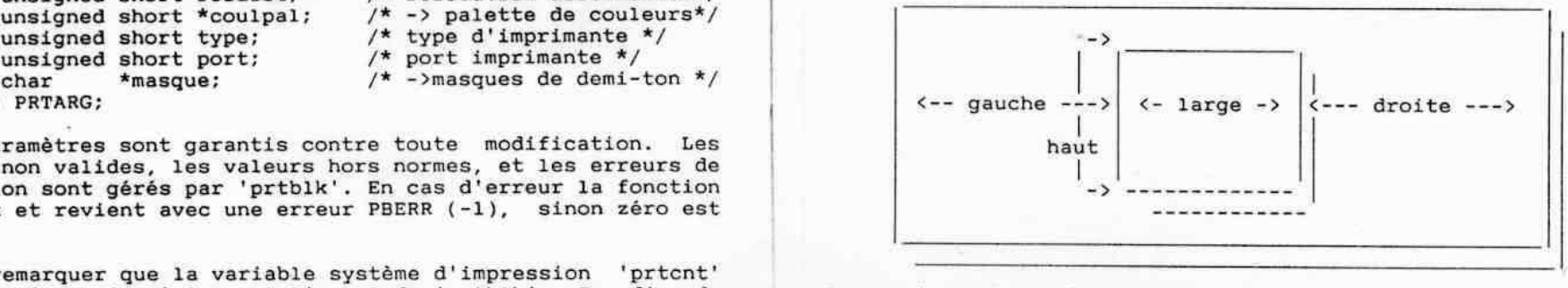
'large' et 'haut' déterminent la taille du bloc de données en bits ou en octets. 'large' peut être interprété soit comme la largeur d'un Bitmap, en pixels, soit comme la longueur d'une chaîne de texte en caractères. La largeur d'un Bitmap est coupée selon les dimensions de résolution d'écran appropriées afin de satisfaire une impression 'WYSIWYG'. 'haut' contient soit la hauteur du Bitmap en trame, soit zéro. Dans ce cas 'prtblk' interprétera le bloc de données comme étant une chaîne de texte.





'droite' et 'gauche' spécifient respectivement l'offset du pixel d'en-tête gauche et du pixel de fin de droite. Ils sont utilisés pour déterminer la ligne de trame suivante du Bitmap. Ils ne sont pas employés en cas de bloc de chaînes de caractères.

Le diagramme suivant dépeint un bloc de données Bitmap de la forme mémoire écran vidéo de l'Atari ST et récapitule les relations entre 'large', 'haut', 'gauche' et 'droite'




'resecr' est la résolution de l'écran source, le ST. Les différents modes de résolution valides supportés par 'prtblk' sont:

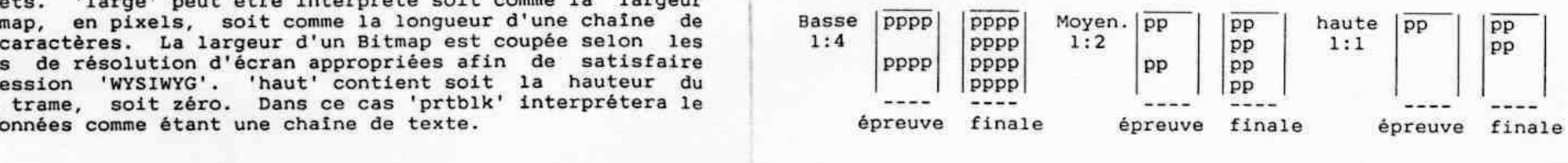
```
0    Basse  320 x 200,  4 plans, palette 16 couleurs
1    Moyenne 640 x 200, 2 plans, palette 4 couleurs
2    Haute  640 x 400,  1 plan, monochrome
```

'resecr' est ignoré si le bloc de données est une chaîne de texte.

'resdest' est la résolution de la destination qui spécifie la qualité et la densité d'impression. Deux résolutions d'impression sont acceptées par 'prtblk':

```
0    épreuve (basse densité)
1    finale (haute densité)
```

'resdest' est ignoré si le bloc de données est une chaîne de texte. Les résolutions de la source et de la destination servent à déterminer la technique de reproduction que 'prtblk' utilisera. Dans le schéma suivant chaque pixel source est tracé sur un espace représenté par la lettre minuscule 'p' (l'espacement vertical est de 1/144 de pouce):



La méthode de reproduction de pixel affichée si-dessus définit le rapport d'aspect et permet de rendre les demi-teintes qui en découlent, en monochrome et en couleur. La reproduction de Bitmap ne peut être désactivée.

'coulpal' est un pointeur de mots qui pointe sur le premier des 16 mots de la palette de couleur de l'Atari ST. La palette de couleur peut être physique ou logique. Chaque entrée dans la palette a 3 bits de niveau d'intensité, rouge, vert et bleu alignés, sur la base des quartets faibles (-----rrr-ggg-bbb). En mode source haute résolution le bit le moins significatif de la première entrée de la palette sert à déterminer la vidéo normale ( noir = 0, blanc = 1),ou la vidéo inverse (noir = 1, blanc = 0). Si le bloc de données est une chaîne de texte, 'coulpal' n'est pas utilisé.

'type' désigne les caractèristiques de traitement de Bitmap de l'imprimante. Actuellement quatre types d'imprimante sont traités par 'prtblk':

0 imprimante matricielle monochrome (mode Bitmap 1/160 de pouce)
1 imprimante matricielle couleur (mode Bitmap 1/160 de pouce)
2 imprimante à marguerite (pas de mode Bitmap)
3 imprimante matricielle monochrome (mode Bitmap 1/120 de pouce)

'type' n'est pas pris en compte si le bloc de données est une chaine texte. Le code de type d'imprimante sélectionne la séquence 'esc' conforme au mode de Bitmap et spécifie l'emploi du traitement demi-teintes monochrome ou couleur. L'Atari STC504 peut travailler en mode couleur et en mode monochrome et la couleur n'est pas placée en mode source haute résolution. Voici la liste des séquences 'esc' utilisées par 'prtblk':

| | |
|---|---|
| "ESC Y" n1 n2 données | mode Bitpap 1/160 de pouce |
| "ESC L" n1 n2 données | mode Bitmap 1/120 de pouce |
| "ESC X" 6 | met la couleur jaune |
| "ESC X" 5 | met la couleur magenta |
| "ESC X" 3 | met la couleur cyan |
| "ESC 3" 1 | espacement de ligne de pixel de 1/144 de pouce |
| "ESC 1" | espacement de ligne de trame de 7/72 de pouce |
| "ESC 2" | replace l'espacement de ligne à 1/6 de pouce |
| "ESC X" 0 | repasse de la couleur au noir |





'port', utilisé en données Bitmap et en texte, est le port de sortie sur lequel l'imprimante est branchée, paralléle ou série, dont 'prtblk' doit se servir :

0 port parallèle d'imprimante
1 port série modem

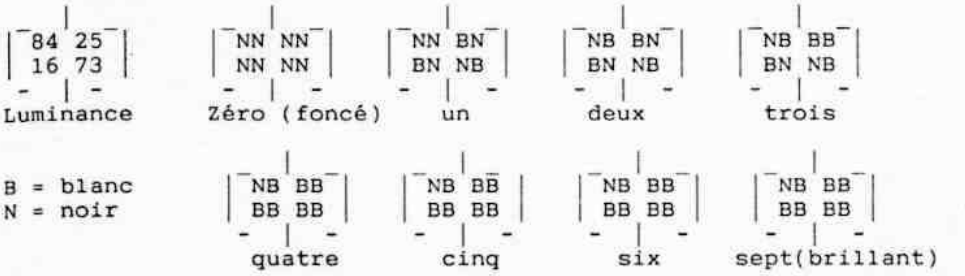
'masque' est un pointeur d'octets qui pointe sur le premier d'un groupe de neuf doubles octets contenant les masques de demi_ton 4 par 2, alignés sur les quartets faibles (matrice de demi-ton Foley et Van Dam). Si la valeur des masques est zéro, 'prtblk' utilisera son propre jeu de masques de demi-ton par défaut:

```
char dmasque[] =        /* masque par défaut demi_ton */
     {
      0x0f, 0x0f, 0x0d, 0x06, 0x09, 0x06,
      0x08, 0x06, 0x08, 0x02, 0x08, 0x00,
      0x08, 0x00, 0x08, 0x00, 0x00, 0x00
     };
```

Lors de l'emploi des masques de demi-ton, les espaces de fin de ligne sont coupés afin de réduire les mouvements de la tête d'impression. En mode source haute résolution chaque pixel est tracé directement, sans faire appel à la routine de demi-tons. 'masque' n'est pas utilisé avec un bloc de données texte.

Une technique de demi-tons monochromes est employée en basse et en moyenne résolution afin de faciliter le rendu sur une imprimante monochrome. Les entrées dans la palette de couleurs sont obtenues par l'intermédiaire de 'coulpal' et sont transformées individuellement en niveaux de demi_tons grâce à la formule de totalisation NTSC RGB; c'est le composant Y du modèle de couleur YIQ, pour la luminance. Un résumé du processus de conversion de la palette de couleurs en demi-tons monochromes suit:

* test si la couleur est blanc pur; un cas spécial.
* luminance = (30% rouge) + (59% vert) + (11% bleu).
* utilise la valeur de luminance pour indexer les demi_tons



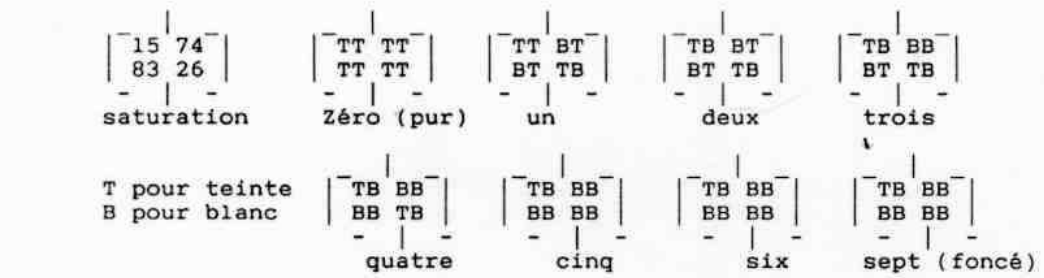

Le demi-ton résultant est masqué durant le processus 'prtblk' de reproduction.

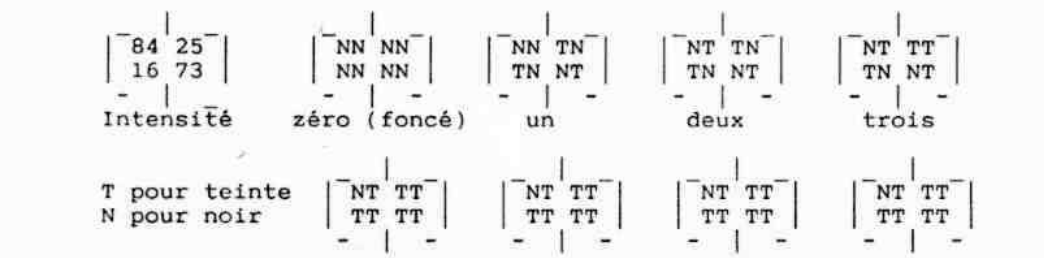
Le processus de demi_tons de nuance, de saturation et d'intensité est exécuté dans le but de rendre sur une imprimante huit couleurs une approche de 512 couleurs. Chaque entrée dans la palette de couleur est obtenue grâce au pointeur 'coulpal' et est convertie dans une couleur demi-ton par défaut, par le processus suivant:

* teste si la couleur est blanc pur; cas spécial.
* intensité = maximum (rouge, vert, bleu) + 1.
* saturation = minimum (rouge , vert, bleu).
* nuance = (rouge, vert, bleu) - (saturation + 1).

| | | |
|---|---|---|
| 0 | noir | (...) |
| 1 | bleu | (..B) |
| 2 | vert | (.V.) |
| 3 | cyan | (.VB) |
| 4 | rouge | (R..) |
| 5 | magenta | (R.B) |
| 6 | jaune | (RV.) |
| 7 | blanc | (RVB) cas spécial |

* index se servant des valeurs de saturation et d'intensité:



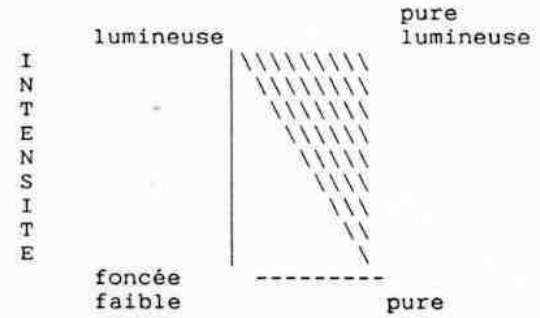




Les demi_tons résultants sont masqués durant le processus 'prtblk' de reproduction.





L'intensité a une priorité de facto sur la saturation à partir du moment où le demi-ton d'intensité est masqué après le demi-ton de saturation. La valeur d'intensité sera toujours égale ou plus grande que la valeur de saturation comme dépeint par la zone hachurée de la représentation en deux dimensions de l'intensité et de la saturation.




VOIR AUSSI:

Le guide du BIOS.
Le guide du programmeur GEM(GSX) volume 1 : VDI

RETOUR

PBERR (0xffff) s'il y a une erreur
0 si OK

ERREURS

* Pas besoin de dire que compacter une gamme de 512 couleurs en 9 niveaux de luminance ou 72 combinaisons de teintes, de saturation et d'intensité est une chose regréttable. Les deux algorithmes de demi-tons monochrome et couleurs pourraient être améliorés.

* Il est impossible de désactiver le de reproduction de pixels.

* Seuls la résolution de destination épreuve et l'affichage de pixel réduit sont valides en mode Bitmap 1/120 de pouce.

* 'prtblk' est dépendant du périphérique. Seuls les formats Atari ST et Epson MX des séquences de codes 'esc' sont acceptés. A partir du moment où un passage à la ligne suivante est généré à la fin de chaque trame, seules les lignes de huit pouces de larges sont acceptées.

IMPRIMANTES

Imprimante Atari SMM804 matricielle à impacte monochrome
Imprimante Atari STC504 matricielle thermique couleur
Imprimante Atari SDM124 à marguerite monochrome
Imprimantes compatibles Epson MX maticielles à impact monochrome
Imprimantes compatibles IBM 5152 matricielles à impact monochrome.





# L A  D O C U M E N T A T I O N  D E  L A  L I G N E  " A "

Afin de fournir un accès simple et rapide en assembleur aux routines graphiques, les ingénieurs d'ATARI ont émulé la ligne "A" du microprocesseur 68000 de façon à offrir une interface vers plusieurs routines utiles. L'interface LIGNE "A" est plus rapide que l'accès habituel par le GEM VDI et offre quelques possibilités supplémentaires. Ainsi, la LIGNE "A" réclame moins de code que des appels équivalents au VDI. Naturellement la LIGNE "A" ne saurait se substituer totalement au VDI, mais si une application ne fait appel qu'à un nombre restreint de fonctions graphiques (et si l'on recherche des performances optimales), alors la LIGNE "A" est suffisante (et parfaite).

L'interface LIGNE "A" est destinée au bidouilleur dans l'âme et on ne saurait se plaindre du manque de facilité de son utilisation. L'interface peut paraître anormalement inconsistante, mais la consistance n'était pas le but recherché; il s'agissait simplement de fournir un moyen d'accéder rapidement aux primitives graphiques. De fait, ces routines constituent le noyau du VDI.

L'interface LIGNE "A" est constituée de 16 codes opératoires. Les appels à la LIGNE "A" s'effectuent sur deux octets (un mot d'instruction 68000), les 4 bits forts de ce mot étant égaux à 1010 (soit 'A' en hexadécimal, d'où le nom LIGNE "A") et les 12 bits faibles constituant le code opératoire proprement dit. Voici un descriptif de ces 16 codes opératoires :

0 = Initialisation
1 = Placer un pixel à une couleur donnée
2 = Demander la couleur d'un pixel
3 = Tracé de ligne quelconque
4 = Tracé de ligne horizontale
5 = Tracé de rectangle rempli
6 = Remplissage d'une ligne d'un polygone
7 = Transfert de bloc de bits
8 = Transfert de matrice de caractère
9 = Visualisation de la souris
10 = Non-visualisation de la souris
11 = Transformation de la forme de la souris
12 = Effacement de lutin ('spirite')
13 = Affichage de lutin ('spirite')
14 = Copie de zone (FDB)
15 = Remplissage de zone

Les routines de la LIGNE "A" offrent quelques options non disponibles par des appels VDI. Les blocs de bits peuvent être traitées comme des matrices demi-tons à la source et les transferts de matrices de caractères peuvent s'effectuer selon les 16 modes logiques, pas seulement dans l'un des 4 modes d'écriture VDI. En dehors de ces ajouts, la LIGNE "A" autorise le programmeur aventureux à expérimenter certains effets spéciaux. Le transfert de blocs est particulièrement généreux sur ce plan.





## 1. (0) Initialisation

```
...         ...
dc.w        $A000              Initialise la LIGNE "A"
...         ...
```

Valeurs d'appel : aucune.

Valeurs retournées :
- d0 = pointeur vers l'adresse de base des variables d'interfaçage à la LIGNE "A".
- a0 = pointeur vers l'adresse de base des variables d'interfaçage à la LIGNE "A".
- a1 = pointeur vers un tableau de pointeurs vers les 3 en-têtes de polices de caractère système.
- a2 = pointeur vers un tableau de pointeurs vers les 16 routines de la LIGNE "A".

Remarque : La valeur retournée dans le registre a0 constitue la pierre angulaire de la LIGNE "A". Les appels à tous les autres codes opératoires de la LIGNE "A" nécessitent le positionnement de variables relatives à ce pointeur, autrement dit les variables de l'interface LIGNE "A" sont contenues dans une structure pointée par a0. Les décalages de ces variables dans la structure sont fournis en fin de chapitre.

Bugue : Dans la toute première version du TOS (en RAM, et non en ROM), a2 n'était pas retourné comme décrit ci-dessus. Car il était préservé durant l'appel à la LIGNE "A". Voir le programme exemple 2 à la fin de ce chapitre quant à la technique à utiliser pour que a2 pointe la bonne valeur.

## 2. (1) Placer un pixel à une couleur donnée

```
...         ...
dc.w        $A001          Colorie un pixel en x et y.
...         ...
```

Valeurs d'appel :
INTIN[0] = valeur du pixel (couleur).
PTSIN[0] = coordonnée horizontale.
PTSIN[1] = coordonnée verticale.

Valeurs retournées :
d0 = couleur du pixel.

Remarque: Se reporter au manuel GEM VDI pour un exposé sur les zones CONTRL, INTIN, PTSIN, INTOUT et PTSOUT.

### 3. (2) Demander la couleur d'un pixel

```
...         ...
dc.w        $A002           Demande la couleur d'un pixel en x,y
...         ...
```

Valeurs d'appel :
PTSIN[0] = coordonnée horizontale.
PTSIN[1] = coordonnée verticale.

Valeur retournée :
d0 = couleur du pixel.

### 4. (3) Tracé de ligne quelconque

```
...         ...
dc.w        $A003           Trace une ligne de (x1,y1) à (x2,y2)
...         ...
```

Valeurs d'appel :

| | |
|---|---|
| X1 | = coordonnée horizontale x1. |
| Y1 | = coordonnée verticale y1. |
| X2 | = coordonnée horizontale x2. |
| Y2 | = coordonnée verticale y2. |
| COLBIT0 | = valeur de bit pour le plan 0. |
| COLBIT1 | = valeur de bit pour le plan 1. |
| COLBIT2 | = valeur de bit pour le plan 2. |
| COLBIT3 | = valeur de bit pour le plan 3. |
| LNMASK | = masque du style de ligne. |
| WMODE | = mode d'écriture. |
| LSTLIN | = doit toujours être positionné à -1 si l'on est en mode XOR, sinon est ignoré. |

Valeur retournée :
LNMASK subit une rotation afin d'être aligné sur le point le plus à droite de la ligne.

Spécificités :
1) Si la ligne est horizontale, LNMASK est une matrice alignée sur un mot, pas un style de ligne. Cela étant, un bit autre que le bit 15 de LNMASK peut être utilisé au point le plus à gauche.

2) Comme la remarque précédente le laisse deviner, la ligne est toujours tracée de la droite vers la gauche, pas obligatoirement de (x1,y1) vers (x2,y2). De sorte que LNMASK est toujours appliqué de gauche à droite.

Note : Du fait de ces spécificités, une application ne peut préjuger de l'état de LNMASK entre des appels aux primitives de tracé de ligne. Si cela a une importance, l'application doit calculer et initialiser LNMASK avant chaque tracé de ligne.





LNMASK est appliqué à la ligne tracée par un algorithme fondé sur la direction de la pente maximale. Si la pente verticale est supérieure à la pente horizontale, alors LNMASK est appliqué dans la direction verticale.

Ces spécificités et ces remarques sur le tracé de ligne s'appliquent également au GEM VDI.

### 5. (4) Tracé de ligne horizontale

```
...         ...
dc.w        $A004           Trace une ligne de (x1,y1) à (x2,y1)
...         ...
```

Valeurs d'appel :

| | |
|---|---|
| X1 | = coordonnée horizontale x1. |
| Y1 | = coordonnée verticale y1. |
| X2 | = coordonnée horizontale x2. |
| COLBIT0 | = valeur de bit pour le plan 0. |
| COLBIT1 | = valeur de bit pour le plan 1. |
| COLBIT2 | = valeur de bit pour le plan 2. |
| COLBIT3 | = valeur de bit pour le plan 3. |
| WMODE | = mode d'écriture. |
| PATPTR | = pointeur vers la matrice de remplissage. |
| PATMSK | = index de la matrice de remplissage. |
| MFILL | = drapeau de mode de remplissage (0=mono-plan). |

Valeur retournée : aucune.

### 6. (5) Tracé de rectangle rempli.

```
...         ...
dc.w        $A005           Trace un rectangle rempli défini
...         ...             par les sommets (x1,y1) et (x2,y2).
```

Valeurs d'appel :

| | |
|---|---|
| X1 | = coordonnée horizontale x1. |
| Y1 | = coordonnée verticale y1. |
| X2 | = coordonnée horizontale x2. |
| Y2 | = coordonnée verticale y2. |
| COLBIT0 | = valeur de bit pour le plan 0. |
| COLBIT1 | = valeur de bit pour le plan 1. |
| COLBIT2 | = valeur de bit pour le plan 2. |
| COLBIT3 | = valeur de bit pour le plan 3. |
| WMODE | = mode d'écriture. |
| PATPTR | = pointeur vers la matrice de remplissage. |
| PATMSK | = index de la matrice de remplissage. |
| MFILL | = drapeau de mode de remplissage (0=mono-plan). |
| CLIP | = drapeau de restriction d'affichage. |
| XMINCL | = x minimum du rectangle restreint. |
| YMINCL | = y minimum du rectangle restreint. |
| XMAXCL | = x maximum du rectangle restreint. |
| YMAXCL | = y maximum du rectangle restreint. |

Valeur retournée : aucune.

### 7. (6) Remplissage d'une ligne d'un polygone

```
...         ...
dc.w        $A006           Trace une ligne horizontale d'un
...         ...             polygone.
```

Valeurs d'appel :

```
PTSIN[]  = zone des vecteurs définissant le polygone
           ((x1,y1),(x2,y2),...,(xn,yn),(x1,y1))
CONTRL[1] = n = nombre de vecteurs.
Y1       = coordonnée verticale de la ligne à tracer.
COLBIT0  = valeur de bit pour le plan 0.
COLBIT1  = valeur de bit pour le plan 1.
COLBIT2  = valeur de bit pour le plan 2.
COLBIT3  = valeur de bit pour le plan 3.
WMODE    = mode d'écriture.
PATPTR   = pointeur vers la matrice de remplissage.
PATMSK   = index de la matrice de remplissage.
MFILL    = drapeau de mode de remplissage (0=mono-plan).
CLIP     = drapeau de restriction d'affichage.
XMINCL   = x minimum du rectangle restreint.
YMINCL   = y minimum du rectangle restreint.
XMAXCL   = x maximum du rectangle restreint.
YMAXCL   = y maximum du rectangle restreint.
```

Valeurs retournées :
X1 et X2 sont modifiés.

Note :
Le premier point du polygone doit figurer en fin de liste des points, autrement dit le polygone doit être fermé.

### 8. (7) Transfert de bloc de bits

```
...         ...
dc.w        $A007           Effectue un transfert de bloc de
...         ...             bits.
```

Valeur d'appel :
a6 = pointeur vers une structure de paramètres d'appel.

Valeur retournée : aucune.

STRUCTURE DES PARAMETRES D'APPEL (relative à a6) :

```
B_WD        equ  +00        largeur du bloc en pixels.
B_HT        equ  +02        hauteur du bloc en pixels.
PLANE_CT    equ  +04        nombre de plans de la destination.
FG_COL      equ  +06        couleur de premier plan de destination.
```





```
BG_COL      equ  +08        couleur d'arrière-plan de destination.
OP_TAB      equ  +10        mode de transfert (voir ci-dessous).
S_XMIN      equ  +14        minimum horizontal de la source.
S_YMIN      equ  +16        minimum vertical de la source.
S_FORM      equ  +18        adresse de base du bloc source.
S_NXWD      equ  +22        décalage vers le mot suivant dans la
                            ligne (source).
S_NXLN      equ  +24        décalage vers la ligne suivante dans le
                            plan (source).
S_NXPL      equ  +26        décalage vers le plan suivant à partir du
                            début de la ligne (source).
D_XMIN      equ  +28        minimum horizontal de la destination.
D_YMIN      equ  +30        minimum vertical de la destination.
D_FORM      equ  +32        adresse de base du bloc destination.
D_NXWD      equ  +36        décalage vers le mot suivant dans la
                            ligne (destination).
D_NXLN      equ  +38        décalage vers la ligne suivante dans le
                            plan (destination).
D_NXPL      equ  +40        décalage vers le plan suivant à partir du
                            début de la ligne (destination).
P_ADDR      equ  +42        adresse du tampon de matrice destination
P_NXLN      equ  +46        décalage vers la ligne suivante dans la
                            matrice de remplissage.
P_NXPL      equ  +48        décalage vers le plan suivant dans la
                            matrice de remplissage.
P_MASK      equ  +50        index du masque de matrice de remplissage

P_BLOCK_LEN  equ +76        le bloc doit avoir 76 octets de long.
```

*** REMARQUES ***

Les contenus de PLANE_CT, FG_COL, BG_COL, P_ADDR peuvent avoir été modifiés après appel de la fonction 'A007'.

Le paramètre OP_TAB est constitué de 4 octets successifs :

| | | |
|---|---|---|
| +00 | octet | opération logique effectuée lorsque les bits de couleur de premier plan et d'arrière-plan sont tous les deux à 0. |
| +01 | octet | opération logique effectuée lorsque le bit de couleur de premier plan est à 0 et le bit de couleur d'arrière-plan est à 1. |
| +02 | octet | opération logique effectuée lorsque le bit de couleur de premier plan est à 1 et le bit de couleur d'arrière-plan est à 0. |
| +03 | octet | opération logique effectuée lorsque les bits de couleur de premier plan et d'arrière-plan sont tous deux à 1. |

#### 8.1. Préface

Avant de vous torturer les méninges à comprendre tous les tenants et aboutissants du transfert de bloc de bits, nous vous

conseillons de revoir le chapitre du GEM VDI traitant de ce sujet. L'auteur part du principe que votre connaissance en la matière est déjà importante et que les fondements du transfert de bloc de bits sont connus. Si l'auteur se trompe, vous l'en voyez désolé (et pour votre part, vous êtes sur le point de vous perdre dans un océan de souffrance, oh oui!).

## 8.2. Bloc de paramètres

La fonction de transfert de bloc de bits est accessible par le biais d'un bloc de paramètres de 76 octets. Le registre a6 pointe le début de ce bloc lors de l'appel à cette instruction de la LIGNE "A". Seuls les 52 premiers octets du bloc ont besoin d'être initialisés à l'appel. L'espace restant est utilisé par la routine de transfert elle-même. Notez que, dans la suite des explications, les paramètres de ce bloc seront désignés par leurs noms symboliques correspondant à leur position dans le bloc.

## 8.3. Formes mémoire

Les formes mémoire ressemblent à des pièces rapportées (une pièce rapportée est un emplacement pour programmeur attardé mental). Trève de plaisanterie, si vous pensez que les formes mémoire sont des pièces rapportées, retournez lire le manuel GEM VDI! Si vous connaissez un peu les formes mémoire, vous savez qu'elles sont presque complètement mais pas totalement distinctes d'un fourre-tout. Les formes de mémoire sont de deux types, source et destination.

Chaque type est défini par les mêmes quatre paramètres : adresse du bloc de la forme, largeur du bloc, décalage jusqu'au mot contigu suivant, et décalage jusqu'au plan suivant.

S_FORM et D_FORM pointent les premiers mots de la forme mémoire source et de la forme mémoire destination, respectivement. Les adresses doivent être paires, faute de quoi on s'expose à une riposte cinglante du système (une erreur d'adresse par exemple).

S_NXWD et D_NXWD correspondent aux décalages à ajouter pour pointer le mot suivant dans un plan de la forme mémoire. Ainsi, ce décalage est de 2 en haute résolution (un plan), de 4 en moyenne résolution (deux plans) et de 8 en basse résolution (4 plans). Ce décalage est exprimé en OCTETS.

S_NXLN et D_NXLN correspondent aux largeurs en OCTETS des blocs source et destination. Ces largeurs doivent nécessairement être paires car, comme vous ne l'ignorez pas, elles représenteront le décalage permettant de passer d'une ligne horizontale de la forme à celle qui suit. Or le début d'une forme devant être aligné sur une adresse paire, la largeur doit correspondre à un nombre pair d'octets. (remarque: la largeur d'écran en haute résolution correspond à 80 octets tandis qu'elle est de 160 octets en basse ou moyenne résolution).





S_NXPL et D_NXPL sont les décalages entre le début d'un plan et le début du plan suivant. Le ST ayant une gestion écran entrelacée, cette valeur est toujours égale à deux (2). D'innombrables possibilités sont ouvertes par l'implémentation de plans contigus dans lesquels les paramètres NXPL désigneraient le nombre d'octets de chaque plan. Ainsi il est possible de transférer un bloc de bits d'une structure contiguë vers la structure entrelacée du ST et réciproquement.

Les blocs de bits des formes mémoire actuelles sont définis dans la structure par leur sommet en haut à gauche, leur largeur et leur hauteur: (S_XMIN, S_YMIN, B_WD et B_HT). L'emplacement correspondant dans la forme de destination est défini par un point d'ancrage (D_XMIN, D_YMIN). La superposition partielle ou totale des deux formes ne pose aucun problème. On notera que l'affichage dans un rectangle restreint ('clipping') est pris en compte et qu'il n'y a aucun test pour déterminer si les blocs de bits sont situés hors ou dans le périmètre de ce rectangle d'affichage restreint. Enfin, le nombre de plans à transférer (nombre d'itérations de l'algorithme de transfert) est défini par le paramètre PLANE_CT.

## 8.4. Opérations logiques

OP_TAB constitue une table de quatre codes opératoires. Chacun des octets constituant cette table correspond à un des seize codes d'opérations logiques pouvant être pratiquées entre la source et la destination. Pour chaque plan, l'opération logique à effectuer est fixée par la valeur des bits de premier et d'arrière plan. Pour un plan donné "n", le bit "n" de la couleur de premier plan est le bit fort de la valeur d'index de deux bits et le bit "n" de la couleur d'arrière-plan est le bit faible de la valeur d'index de deux bits.

Pour ceux qui aiment les répétitions, voici la table :

| Premier plan (n) | Arrière-plan (n) | Entrée OP_TAB |
|---|---|---|
| 0 | 0 | première entrée |
| 0 | 1 | deuxième entrée |
| 1 | 0 | troisième entrée |
| 1 | 1 | quatrième entrée |

## 8.5. Matrices

Les matrices ont une largeur d'un mot et constituent des images alignées sur un mot qui sont appliquées par ET logique à la source avant d'effectuer la combinaison logique de la source et de la destination.

Les matrices sont formatées dans une grille imaginaire ancrée dans le coin haut gauche (0,0) de la forme mémoire de destination. Les matrices ont une largeur de 16 bits et se répètent donc tous les 16 bits.

Les matrices ont une hauteur en pixels égale à une puissance de 2 et sont répétées verticalement selon cette fréquence.

La source est décalée pour s'aligner sur la destination avant de combiner la source à la matrice. De sorte que la relation entre la source et la matrice est fonction de la position x,y du rectangle de destination.

P_ADDR pointe le premier mot de la matrice. Si ce pointeur est nul, aucune matrice n'est combinée à la source.

P_NXLN est le décalage (en OCTETS) entre deux mots consécutifs de la matrice. Pour des raisons obscures, ce nombre doit être une puissance entière de 2 (par exemple 2,4 ou 8).

P_NXPL est le décalage (en OCTETS) entre le début d'un plan et le début du plan suivant. Dans le cas d'une matrice mono-plan utilisée dans un environnement multi-plan, cette valeur doit être nulle de façon que cette matrice soit répétée pour tous les plans.

P_MASK définit, avec P_NXLN, la longueur de la matrice. Cette longueur (en MOTS) doit être une puissance entière de 2.

Si P_NXLN = 2 ^ n alors P_MASK = (longueur en mots - 1) << n

## 8.6. TRUCS ET ASTUCES

Q. Je désire transférer un bloc d'une source mono-plan vers une destination multi-plan.

R. Voilà qui n'est pas vraiment une question. D'autre part, j'ai du mal à réfléchir avec de l'eau dans les oreilles. Hé, c'est mon chat que vous mettez dans le four. Est-ce que vous croyez qu'on peut jouer comme ça avec un traitement de texte. Arrêtez de vouloir appuyer la touche Delete. Non vraiment, je suis sérieux. Voulez-vous vraiment qu'on s'en arrête là ?

Q. C'est parti et vous n'êtes pas loin de remporter le bonus.

R. ok, ok... je vais parler.

S_NXPL = 0 => chaque plan de la source est transféré vers tous les plans de la destination.

Q. bon, je savais ça mais de quels opérateurs logiques dois-je me servir ?

R. pour placer des 1 comme couleur de premier plan et des 0 comme couleur d'arrière-plan, placer les octets d'OP_TAB ainsi:





| décalage | code logique | |
|---|---|---|
| +00 | 00 | tous à 0 |
| +01 | 04 | D' <= [NOT S] AND D |
| +02 | 07 | D' <= S OU D |
| +03 | 15 | tous à 1 |

charge la couleur de premier plan dans FG_COL et la couleur d'arrière-plan dans BG_COL.

Q. Vous ne pourriez pas m'en dire plus ?

A. Pour placer des 1 comme couleur de premier plan et rendre transparents les 0, fixer OP_TAB selon les valeurs:

| décalage | code logique | |
|---|---|---|
| +00 | 04 | D' <= [NOT S] ET D |
| +01 | 04 | D' <= [NOT S] ET D |
| +02 | 07 | D' <= S OU D |
| +03 | 07 | D' <= S OU D |

Chargez la couleur de premier plan dans FG_COL.
La couleur placée dans BG_COL n'a aucune importance.

N'oubliez pas de placer S_NXPL à 0.

Trêve de banalités, laissez-moi terminer mon travail. Voici quelques-unes des recettes les plus succulentes de Tante Marie :

1. Transfert d'une matrice sans Source vers une Destination.

Pour cela, vous aurez besoin d'un mot de 1. Nommez-le "un:" puis faites pointer S_FORM vers "un:". Placez S_NXLN, S_NXPL, S_NXWD, S_XMIN et S_YMIN à 0. Initialisez la matrice avec les valeurs que vous voulez et avant de comprendre ce qu'il se passe, vous aurez un rectangle rempli avec une fabuleuse matrice.

2. Recette simple pour faire d'un lutin un être animé.

Il vous faudra d'abord cuire un masque mono-plan. Chaque fois qu'il y a un 1 dans le masque, l'arrière-plan sera recouvert et là où se trouvera un 0, l'arrière-plan sera conservé tel quel.

Placer OP_TAB avec :

| décalage | code logique | |
|---|---|---|
| +00 | 04 | D' <= [NOT S] ET D |
| +01 | 04 | D' <= [NOT S] ET D |
| +02 | 07 | D' <= S OU D |
| +03 | 07 | D' <= S OU D |

Initialiser FG_COL avec la couleur de premier plan.
Peu importe ce que vous mettrez dans BG_COL.

Ensuite, prenez la forme mono-plan (ou multi-plans) et faites un OU logique entre cette forme et la zone que vous voulez maquiller avec ce masque.

Dégustez cela en famille.

## 9. (8) Transfert de matrice de caractère

```
...         ...
dc.w        $A008       Effectue un transfert de matrice d'un
...         ...         caractère.
```

Valeurs d'appel :

| | |
|---|---|
| WMODE | = mode d'écriture (0 à 3 => modes VDI, 4 à 19 => modes transfert). |
| TEXTFG | = couleur d'écriture du texte. |
| TEXTBG | = couleur de fond du texte (pour modes 4 à 19). |
| FBASE | = pointeur vers les données de la police. |
| FWIDTH | = taille de la police de caractère. |
| SOURCEX | = coord. horizontale du caractère dans police. |
| SOURCEY | = coord. verticale du caractère dans police. |
| DESTX | = coordonnée horizontale du caractère à l'écran |
| DESTY | = coordonnée verticale du caractère à l'écran. |
| DELX | = taille du caractère. |
| DELY | = hauteur du caractère. |
| STYLE | = vecteur des effets spéciaux d'écriture. |
| LITEMASK | = masque pour les caractères grisés. |
| SKEWMASK | = masque pour les caractères italiques. |
| WEIGHT | = taille d'épaississement de caractère gras. |
| ROFF | = décalage au-dessus de la ligne de base du caractère pour le tracé d'italique. |
| LOFF | = décalage en-dessous de la ligne de base du caractère pour le tracé d'italique. |
| SCALE | = drapeau de mise à l'échelle (0=pas d'échelle) |
| XDDA | = accumulateur pour la pente horizontale. |
| DDAINC | = proportion de réduction ou augmentation d'échelle. |
| SCALDIR | = drapeau de direction d'échelle (0 vers bas). |
| CHUP | = vecteur de rotation de caractère. |
| MONO | = drapeau de police mono-espacée. |
| SCRTCHP | = pointeur vers le tampon de travail pour les effets spéciaux d'écriture. |
| SCRPT2 | = décalage du tampon de travail échelle dans le tampon décrit ci-dessus. |

Valeur retournée : aucune.

## 10. (9) Visualisation de la souris

```
...         ...
dc.w        $A009           Montre le pointeur de souris.
...         ...
```





Valeurs d'appel :
INTIN[0] force la visualisation si nul.

Valeur retournée : aucune.

## 11. (10) Non-visualisation de la souris

```
...         ...
dc.w        $A00A           Cache le pointeur de souris.
...         ...
```

Aucun paramètre en entrée ou sortie.

## 12. (11) Transformation de la forme de la souris

```
...         ...
dc.w        $A00B           Transforme la forme de la souris.
...         ...
```

Valeurs d'appel :

| | |
|---|---|
| INTIN[0] | = coordonnée x du sommet haut gauche de forme. |
| INTIN[1] | = coordonnée y du sommet haut gauche de forme. |
| INTIN[2] | = 1. |
| INTIN[3] | = couleur du masque (1 par défaut). |
| INTIN[4] | = couleur des données (1 par défaut). |
| INTIN[5] à INTIN[20] | = masque de la forme (16 mots). |
| INTIN[21] à INTIN[36] | = données de la forme (16 mots). |

Valeur retournée : aucune.

## 13. (12) Effacement de lutin

```
...         ...
dc.w        $A00C           efface un lutin déjà affiché.
...         ...
```

Valeur d'appel :
a2 = pointeur vers le bloc de sauvegarde du lutin.

Note : Le bloc de sauvegarde du lutin sert à sauvegarder le contenu de l'écran recouvert par le lutin. Sa taille est de 10 octets + 64 octets par plan, soit (10 + VPLANES *64) octets.

Valeur retournée :
a6 modifié.

Cette primitive replace à l'écran l'ancien contenu de la zone écran placé sous le lutin lors de l'affichage précédent de ce dernier (voir ci-dessous la primitive $A00D).

## 14. (13) Affichage de lutin

```
...         ...
dc.w        $A00D           Affichage de spirite.
...         ...
```

Valeurs d'appel :
- d0 = x du point significatif.
- d1 = y du point significatif.
- a0 = pointeur vers le bloc de définition du lutin.
- a2 = pointeur vers le bloc de sauvegarde du lutin.

Le point significatif d'un lutin correspond au point de la matrice servant de base de coordonnées lors du déplacement du lutin, en général c'est le coin en haut à gauche.

STRUCTURE DU BLOC DE DEFINITION D'UN LUTIN

```
ds.w    1    décalage horizontal entre le sommet en haut à
             gauche du lutin et le point significatif.
ds.w    1    décalage vertical entre le sommet en haut à
             gauche du lutin et le point significatif.
ds.w    1    drapeau de format (1 => Format VDI,-1 => XOR).
```

Format VDI

| bit écriture | bit fond | action |
|---|---|---|
| 0 | 0 | transparent à l'écran |
| 0 | 1 | couleur de fond placée |
| 1 | 0 | couleur caract. placée |
| 1 | 1 | couleur caract. placée |

Format XOR

| bit écriture | bit fond | action |
|---|---|---|
| 0 | 0 | transparent à l'écran |
| 0 | 1 | couleur de fond placée |
| 1 | 0 | couleur caract. placée |
| 1 | 1 | couleur caract. placée |

```
ds.w    1    couleur de fond (index de table de couleurs).
ds.w    1    couleur de caractère (index table de couleurs)
ds.w   32    image entrelacée premier plan/arrière-plan.
             (mot 0 = arrière-plan ligne 0.
              mot 1 = premier plan ligne 0.
              mot 2 = arrière-plan ligne 1.
              mot 3 = premier plan ligne 1.
              etc.)
```

Valeurs retournées :
modifie a6. (Programmeurs en "C", prenez-garde!)

Bugues : Cette fonction n'est pas utilisable dans la 1ère version du TOS. Voir programme 2 en fin de chapitre.





## 15. (14) Copie de zone (FDB)

```
...         ...
dc.w        $A00E       Copie une zone d'une source vers une
...         ...         destination.
```

Valeurs d'appel :
- CONTRL[7-8] = adresse de la forme source FDB.
- CONTRL[9-10] = adresse de la forme destination FDB.
- INTIN[0] = mode d'écriture.
- INTIN[1] = couleur affectée aux bits à 1.
- INTIN[2] = couleur affectée aux bits à 0.
- PTSIN[0] = coordonnée horizontale du sommet de la source
- PTSIN[1] = coordonnée verticale du sommet de la source.
- PTSIN[2] = coordonnée hor. du sommet opposé de la source
- PTSIN[3] = coordonnée ver. du sommet opposé de la source
- PTSIN[4] = coordonnée horizontale sommet de destination
- PTSIN[5] = coordonnée verticale sommet de destination.
- PTSIN[6] = coordonnée hor. sommet opposé de destination.
- PTSIN[7] = coordonnée ver. sommet opposé de destination.
- typcop = 0 si mode vro_cpyfm, -1 si mode vrt_cpyfm.

Valeur retournée : aucune.

Note : voir la discussion sur la primitive $A007 plus haut.

## 16. (15) Remplissage de zone ligne à ligne

```
...         ...
dc.w        $A00F           Remplit une zone définie par
...         ...             un contour.
```

Valeurs d'appel :
- INTIN[0] = couleur de contour.
- PTSIN[0] = coordonnée horizontale du point de départ.
- PTSIN[1] = coordonnée verticale du point de départ.
- SEEDABORT= adresse de routine d'arrêt de remplissage.

Valeur retournée : aucune.

N.D.T.: Le traducteur s'est permis de documenter cette fonction de la ligne "A" bien que la documentation originale d'Atari Corp. n'en fasse mention qu'incidemment. Il doit donc être tenu comme seul responsable des informations fournies.

Cette primitive correspond à la fonction vdi v_contourfill. Avec une couleur de contour négative, l'algorithme de remplissage sélectionne une couleur de contour différente de celle du pixel de départ. SEEDABORT doit pointer vers une routine qui sera exécutée après chaque remplissage de ligne horizontale. Si cette routine renvoie 0, le remplissage se poursuit; sinon, il est stoppé.

## 17. Utilisation de la LIGNE "A"

Les paramètres d'appel des routines de la LIGNE "A" sont placées dans une structure pointée par la valeur retournée par a0 suite à un appel de la primitive d'initialisation ($A000). Cette initialisation n'a besoin d'être faite qu'une fois, les valeurs retournées pouvant être sauvées et resservir à tout moment.

L'interface LIGNE "A" peut être utilisée parallèlement au VDI et à l'AES, mais on ne saurait préjuger de l'état des variables après l'appel d'une routine VDI ou AES (NDT: Certaines fonctions GEM partagent en effet avec la ligne "A" les mêmes variables). En conséquence, si une application veut mêler des appels au VDI et à l'AES avec des appels à la ligne "A", il faut préserver toutes les variables communes entre chaque appel.

Le programme appellant doit tenir compte du fait que les registres d0 à d2 et a0 à a2 sont modifiés au retour d'une primitive de la ligne "A". Les autres registres sont inchangés, excepté a6 modifié en retour de "$A00C" et "$A00D".

STRUCTURE DES PARAMETRES D'APPEL DE LA LIGNE "A" :

| déc. | nom | type | description |
|---|---|---|---|
| 0 | VPLANES | mot | nombre de plans video. |
| 2 | VWRAP | mot | nombre d'octets par ligne video |
| | note: | | Ces variables peuvent être modifiées si l'on désire créer des effets spéciaux. Ainsi, en doublant VWRAP on obtiendra un affichage une ligne sur deux. Bien sûr de telles modifications sur ces variables sont à éviter lorsqu'on recherche un fonctionnement normal de la LIGNE "A" (ou du VDI). |
| 4 | CONTRL | long | pointeur vers la zone CONTRL (cf. VDI). |
| 8 | INTIN | long | pointeur vers la zone INTIN (cf. VDI). |
| 12 | PTSIN | long | pointeur vers la zone PTSIN (cf. VDI). |
| 16 | INTOUT | long | pointeur vers la zone INTOUT (cf. VDI). |
| 20 | PTSOUT | long | pointeur vers la zone PTSOUT (cf. VDI). |
| 24 | COLBIT0 | mot | couleur courante pour le plan 0 (un bit). |
| 26 | COLBIT1 | mot | couleur courante pour le plan 1 (un bit). |
| 28 | COLBIT2 | mot | couleur courante pour le plan 2 (un bit). |
| 30 | COLBIT3 | mot | couleur courante pour le plan 3 (un bit). |
| | note: | | La couleur courante (couleur d'écriture) est égale à: (1*COLBIT0 + 2*COLBIT1 + 4*COLBIT2 + 8*COLBIT3). |
| 32 | LSTLIN | mot | placez-le à -1 et n'y pensez plus. |
| 34 | LNMASK | mot | équivalent au style de ligne VDI. |
| 36 | WMODE | mot | équivalent au mode d'écriture VDI (0=mode REPLACE, 1=mode TRANSPARENT, 2=mode XOR, 3=mode TRANSPARENT INVERSE). (voir le manuel VDI pour les modes d'écriture). |





| | | | |
|---|---|---|---|
| 38 | X1 | mot | coordonnée horizontale x1. |
| 40 | Y1 | mot | coordonnée verticale y1. |
| 42 | X2 | mot | coordonnée horizontale x2. |
| 44 | Y2 | mot | coordonnée verticale y2. |
| 46 | PATPTR | long | pointeur vers la matrice de remplissage. |
| 50 | PATMSK | mot | masque de matrice de remplissage. |
| 52 | MFILL | mot | drapeau de remplissage multi-plan. (0 => matrice courante mono-plan, 1 => matrice courante multi-plan). |
| 54 | CLIP | mot | drapeau de restriction d'affichage (0 = pas de restriction). |
| 56 | XMINCL | mot | valeur hor. minimale rectangle restreint. |
| 58 | YMINCL | mot | valeur ver. minimale rectangle restreint. |
| 60 | XMAXCL | mot | valeur hor. maximale rectangle restreint. |
| 62 | YMAXCL | mot | valeur ver. maximale rectangle restreint. |
| 64 | DDA | mot | accumulateur pour la pente horizontale transfert de matrice de caractère (doit être initialisé à $8000 avant appel). |
| 66 | DDAINC | mot | fraction à ajouter ou retrancher pour mettre à l'échelle un caractère. |
| | note: | | Si augmentation de taille, placer DDAINC à: 256*(Taille désirée-Taille actuelle)/Taille actuelle Si réduction de taille, placer DDAINC à: 256*Taille désirée/Taille actuelle |
| 68 | SCALDIR | mot | drapeau de direction d'échelle (0 =>bas). |
| 70 | MONO | mot | 0 => la police courante n'est pas mono espacée OU on peut augmenter la largeur d'un caractère s'il est gras. 1 => la police courante est mono-espacée ET l'on ne doit pas augmenter la largeur d'un caractère gras. |
| 72 | SOURCEX | mot | coordonnée horizontale dans la police. |
| 74 | SOURCEY | mot | coordonnée verticale dans la police. |

note: SOURCEX peut être calculé d'après les informations placées dans l'en-tête de police. Exemple :

```
temp = valeur du caractère;
temp -= fnt_ptr->first_ade;(valeur 1er caractère)
SOURCEX = fnt_ptr->off_table(temp);
```

| | | | |
|---|---|---|---|
| 76 | DESTX | mot | coordonnée x du caractère à l'écran. |
| 78 | DESTY | mot | coordonnée y du caractère à l'écran. |
| 80 | DELX | mot | largeur de caractère. |
| 82 | DELY | mot | hauteur de caractère. |

note: DELX et DELY peuvent être calculés à partir de l'en-tête de police. Par exemple

```
temp = valeur du caractère;
temp -= fnt_ptr->first_ade; (valeur 1er caractère)
SOURCEX = fnt_ptr->off_table(temp);
DELX = fnt_ptr->off_table(temp-1) - SOURCEX;
DELY = fnt_ptr->form_height;
```

```
84   FBASE     long      pointeur vers base des données de police.
88   FWIDTH    mot       taille de la police de caractères.

        note: FBASE  et FWIDTH peuvent être calculés à partir de
              l'en-tête de police. Par exemple:
               FBASE  = fnt_ptr->dat_table;
               FWIDTH = fnt_ptr->form_width;

90   STYLE     mot       vecteur des drapeaux d'effets spéciaux.
                         bit 0 : drapeau de gras.
                         bit 1 : drapeau de grisé.
                         bit 2 : drapeau d'italique.
                         bit 3 : drapeau de soulignement.(ignoré)
                         bit 4 : drapeau de contour ('outlined').

        note: Placer  les bits à 1 pour obtenir l'effet  désiré.
              Le soulignement n'est pas pris en compte.

92   LITEMASK  mot       masque à utiliser pour le texte grisé.
94   SKEWMASK  mot       masque à utiliser pour le texte italique.
96   WEIGHT    mot       taille de graissage de texte.
98   ROFF      mot       décalage sur la ligne de base si italique
100  LOFF      mot       décalage sous ligne de base pour italique

        note: Les 5 variables ci-dessus peuvent être calculées à
              partir de l'en-tête de police:
              LITEMASK = fnt_ptr->lighten;
              SKEWMASK = fnt_ptr->skew;
              WEIGHT = fnt_ptr->thicken;
              si italique : { ROFF = fnt_ptr->right_offset;
                              LOFF = fnt_ptr->left_offset; }
              sinon { ROFF = 0; LOFF = 0; }

102  SCALE     mot       drapeau d'échelle. (0 => pas d'échelle.)
104  CHUP      mot       vecteur de rotation de caractère.
                         0 => direction normale horizontale
                         900 => rotation de 90° sens d'horloge
                         1800 => rotation de 180° sens d'horloge
                         2700 => rotation de 270° sens d'horloge
106  TEXTFG    mot       couleur de premier plan du texte
108  SCRTCHP   long      pointeur vers le début du tampon de tra-
                         vail pour les effets spéciaux texte.
112  SCRPT2    mot       décalage du tampon d'échelle par rapport
                         au début du tampon ci-dessus.

        note: Ces pointeurs de tampons d'effets spéciaux doivent
              être initialisés avant l'appel de $A007.

114  TEXTBG    mot       couleur d'arrière-plan du texte.
116  COPYTRAN  mot       drapeau de type de copie de zone.
                         (0 => type opaque
                            source n-plans -> destination n-plans
                            modes d'écriture transfert de bloc
                          <>0 => type transparent
```





```
                            source monoplan -> destination n-plans
                            modes d'écriture VDI.
118  SEEDABORT long      pointeur vers la routine qui est appelée
                         après chaque ligne de remplissage ($A00F)

        note: Ce pointeur n'existe pas dans la première version
              du TOS en RAM. Voir l'exemple 2 pour la méthode à
              utiliser pour identifier la version 1 du TOS.
```

## 18. Exemples de programmes

```
*
* Initialisation des décalages de la structure
* des paramètres d'appel.
*
VPLANES     equ     0
VWRAP       equ     2
CONTRL      equ     4
INTIN       equ     8
PTSIN       equ     12
INTOUT      equ     16
PTSOUT      equ     20
COLBIT0     equ     24
COLBIT1     equ     26
COLBIT2     equ     28
COLBIT3     equ     30
LSTLIN      equ     32
LNMASK      equ     34
WMODE       equ     36
X1          equ     38
Y1          equ     40
X2          equ     42
Y2          equ     44
PATPTR      equ     46
PATMSK      equ     50
MFILL       equ     52
CLIP        equ     54
XMINCL      equ     56
YMINCL      equ     58
XMAXCL      equ     60
YMAXCL      equ     62
XDDA        equ     64
DDAINC      equ     66
SCALDIR     equ     68
MONO        equ     70
SRCX        equ     72
SRCY        equ     74
DSTX        equ     76
DSTY        equ     78
DELX        equ     80
DELY        equ     82
FBASE       equ     84
```

```
FWIDTH          equ             88
STYLE           equ             90
LITEMSK         equ             92
SKEWMSK         equ             94
WEIGHT          equ             96
ROFF            equ             98
LOFF            equ             100
SCALE           equ             102
CHUP            equ             104
TEXTFG          equ             106
SCRTCHP         equ             108
SCRT2           equ             112
TEXTBG          equ             114
COPYTRAN        equ             116
SEEDABORT       equ             118
*
*
* Codes des fonctions de la LIGNE "A"
*
INIT            equ             $A000
PUTPIX          equ             INIT+1
GETPIX          equ             INIT+2
ABLINE          equ             INIT+3
HABLINE         equ             INIT+4
RECTFILL        equ             INIT+5
POLYFILL        equ             INIT+6
BITBLT          equ             INIT+7
TEXTBLT         equ             INIT+8
SHOWCUR         equ             INIT+9
HIDECUR         equ             INIT+10
CHGCUR          equ             INIT+11
DRSPIRITE       equ             INIT+12
UNSPIRITE       equ             INIT+13
COPYRSTR        equ             INIT+14
SEEDFILL        equ             INIT+15
```

PROGRAMME EXEMPLE 1

```
        .text
start:
        dc.w        INIT                * initialisation LIGNE "A"
        move.w      #-1,LSTLIN(a0)      * une fois pour toutes
        move.w      #$5555,LNMASK(a0)   * ligne pointillée
        move.w      #0,WDMODE(a0)       * mode REPLACE
        move.w      #1,COLBIT0(a0)      * plan 0
        move.w      #1,COLBIT1(a0)      * plan 1
        move.w      #1,COLBIT2(a0)      * plan 2
        move.w      #0,COLBIT3(a0)      * plan 3 (couleur = 7)
        move.w      #0,X1(a0)           * abcisse de départ
        move.w      #0,Y1(a0)           * ordonnée de départ
        move.w      #99,X2(a0)          * abcisse d'arrivée
        move.w      #99,Y2(a0)          * ordonnée d'arrivée
        dc.w        ABLINE              * tracé de ligne
```





```
        .
        .
        .
        move.w      #0,-(sp)            * Pterm0
        trap        #1                  * Fin du programme
.end
```

PROGRAMME EXEMPLE 2

```
        .text
*
*
*
start:
        clr.l       -(sp)
        move.w      #$20,-(sp)          * Passage en mode Superviseur
        trap        #1                  * pour appeler les routines de
        addq.l      #6,sp               * la ligne "A" par jsr
        move.l      d0,stksave          * préserve ancien pointeur pile
*
* Recherche quelle version du gestionnaire de la ligne "A"
* est implantée dans le système
*
        move.l      #0,a2               * valeur correcte pour le test
        dc.w        INIT                * initialisation de la LIGNE "A"
        move.l      a2,d2               * ancienne version ?
        bne         a2ok                * non, a2 pointe la zone des
*                                         pointeurs de routines LIGNE "A"
        lea         -4*15(a1),a2        * oui, a2 est intact, donc on
*                                        utilise a1 plus un déplacement.
*
* a2 pointe maintenant la zone des adresses de routines LIGNE "A"
*
a2ok:
        move.w      #1,old_linea        * drapeau routine appelable
        move.l      4*$D(a2),drawaddr   * adresse routine tracé lutin
*
* Initialisation complète
*
        moveq       #0,d0               * initialise x
        moveq       #0,d1               * initialise y
        lea         spirite,a0          * pointe le bloc du lutin
        lea         save,a2             * pointe le bloc de sauvegarde
loop:
        movem.w     d0-d1,-(sp)         * sauve x et y
        movem.l     a0/a2,-(sp)         * sauve les pointeurs
        move.l      a6,-(sp)            * sauve le registre modifié
        tst.w       old_linea           * vieux ou nouveau gestionnaire?
        beq         new                 * si nouveau, branchement
        move.l      drawaddr,a3         * adresse routine tracé de lutin
        jsr         (a3)                * tracé du lutin
        bra         merge
*
```

```
new:
     dc.w      DRSPIRITE          * trace par fonction normale
*
merge:
     move.l    (sp)+,a6           * récupère contenu de a6
     movem.l   (sp)+,a0/a2        * restitue les registres
*
     move.w    #2000,d2           * longueur temporisation
wait:
     dbra      d2,wait            * petite attente
*
     movem.l   a0/a2,-(sp)        * sauve les pointeurs
     move.l    a6,-(sp)           * sauve le registre a6
     dc.w      UNSPIRITE          * effacement de spirite
     move.l    (sp)+,a6           * restitue a6 à sa valeur
     movem.l   (sp)+,a0/a2        * restitue les pointeurs
     movem.w   (sp)+,d0-d1        * restaure x et y
     addq.w    #1,d0              * incrémente l'abcisse
     cmpi.w    #640,d0            * est-on en fin d'écran ?
     blt       loop               * non, recommence le tracé
*
     move.l    stksave,-(sp)      * restitue le ptr de pile
     move.w    #$20,-(a7)         * Retour en mode utilisateur
     trap      #1
     addq.w    #6,sp
*
     move.w    #0,-(sp)           * Pterm0
     trap      #1                 * Fin du programme
*
*
.data
*
*
spirite:
     dc.w      0,0                * décalages x,y point signific.
     dc.w      1,0,1              * format, couleur masque,données
     dc.w      $FFFF              * ligne 0 d'arrière-plan
     dc.w      $07F0              * ligne 0 de premier plan
     dc.w      $FFFF              * ligne 1 d'arrière-plan
     dc.w      $0FF8              * ligne 1 de premier plan
     dc.w      $FFFF              * ligne 2 d'arrière-plan
     dc.w      $1FEC              * ligne 2 de premier plan
     dc.w      $FFFF              * ligne 3 d'arrière-plan
     dc.w      $1804              * ligne 3 de premier plan
     dc.w      $FFFF              * ligne 4 d'arrière-plan
     dc.w      $1804              * ligne 4 de premier plan
     dc.w      $FFFF              * ligne 5 d'arrière-plan
     dc.w      $1004              * ligne 5 de premier plan
     dc.w      $FFFF              * ligne 6 d'arrière-plan
     dc.w      $1E3C              * ligne 6 de premier plan
     dc.w      $FFFF              * ligne 7 d'arrière-plan
     dc.w      $1754              * ligne 7 de premier plan
     dc.w      $FFFF              * ligne 8 d'arrière-plan
     dc.w      $1104              * ligne 8 de premier plan
```





```
     dc.w      $FFFF              * ligne 9 d'arrière-plan
     dc.w      $0B28              * ligne 9 de premier plan
     dc.w      $FFFF              * ligne 10 d'arrière-plan
     dc.w      $0DD8              * ligne 10 de premier plan
     dc.w      $FFFF              * ligne 11 d'arrière-plan
     dc.w      $0628              * ligne 11 de premier plan
     dc.w      $FFFF              * ligne 12 d'arrière-plan
     dc.w      $07D0              * ligne 12 de premier plan
     dc.w      $FFFF              * ligne 13 d'arrière-plan
     dc.w      $2E10              * ligne 13 de premier plan
     dc.w      $FFFF              * ligne 14 d'arrière-plan
     dc.w      $39E0              * ligne 14 de premier plan
     dc.w      $FFFF              * ligne 15 d'arrière-plan
     dc.w      $3800              * ligne 15 de premier plan
*
*
.bss
*
*
stksave:     ds.l    1
save:        ds.b    10+64
old_linea:   ds.w    1
drawaddr:    ds.l    1
*
.end
```

# CLAVIER INTELLIGENT

**PROTOCOLE DU CLAVIER INTELLIGENT**
**(Mise à jour : 26/02/85)**

## 1. Introduction

Le clavier intelligent (IKBD) d'Atari Corp. est un contrôleur clavier d'usage commun suffisamment flexible pour être utilisé avec un grand nombre de produits sans modifications. Le clavier, avec son contrôleur, fournit des points de connexion convenables pour une souris et des manettes de jeu. Le processeur clavier comporte également une horloge avec gestion de la date d'une précision d'une seconde.

Le clavier intelligent a été conçu comme suffisamment général pour pouvoir servir dans un grand nombre de nouveaux produits informatiques. Des ajustements quant au nombre de touches du clavier, la résolution de la souris, etc. peuvent être effectués.

Le clavier intelligent communique avec le processeur maître par une liaison série bi-directionnelle à haute vitesse. Il peut fonctionner dans plusieurs modes afin de rendre possibles diverses utilisations du clavier, des manettes de jeu ou de la souris. Un usage restreint du contrôleur est également autorisé pour des applications dans lesquelles un mode de communication unidirectionnel est utile, ceci par une gestion soigneuse des modes par défaut.

## 2. Documents de référence

Atari Corp. RBP/GHU/SD Keyboard Schematic (14/09/84)
Atari Corp. RBP/GHU/SD Keyboard Layout (non paginé, non daté)

## 3. Clavier

Le clavier retourne toujours les codes d'appui et de relâchement de touche, chaque fois qu'une touche est pressée ou relâchée. Les codes d'appui de touche débutent à 1 et sont définis dans l'appendice A en fin de chapitre. A titre d'exemple, la position de la touche ISO dans la table des codes de touches pressées existera même si aucune touche ne correspond à cette position sur un clavier particulier. Le code de relâchement de touche peut être obtenu en effectuant un OU logique entre 0x80 et le code d'appui de la touche (bit 7 positionné à 1).

Les codes spécifiques 0xF6 à 0xFF sont réservés pour les fonctions suivantes :





| | |
|---|---|
| 0xF6 | compte-rendu d'état |
| 0xF7 | position absolue de la souris |
| 0xF8-0xFB | position relative de la souris<br>(bit 0 à 1 si bouton droit appuyé<br>bit 1 à 1 si bouton gauche appuyé) |
| 0xFC | heure et date |
| 0xFD | descripteur d'état manettes de jeu |
| 0xFE | événement manette de jeu 0 |
| 0xFF | événement manette de jeu 1 |

Les deux touches 'Shift' retournent des codes d'appui différents. Il en va de même pour les touches 'Enter' et 'Return'.

## 4. Souris

Le port souris doit être capable de contrôler une souris avec une résolution d'environ 200 comptes (déplacements ou cliquages) par pouce (NDT: soit 80 positions par centimètres environ). La souris doit être scrutée suffisament souvent pour autoriser le suivi de déplacements à une vitesse allant jusqu'à 10 pouces par seconde (NDT: soit environ 25 cm/s).

Le clavier intelligent peut rendre compte des actions sur la souris selon trois modes distincts. Il peut rendre compte de la position relative (à la précédente), de la position absolue de la souris dans un système de coordonnées convenu, ou convertir les déplacements de la souris en codes clavier curseur équivalents.

Les boutons de la souris peuvent être traités comme partie intégrante de la souris ou comme des touches de clavier supplémentaires.

### 4.1. Positionnement relatif

En mode positionnement relatif, le clavier retournera un descripteur de position de la souris lorsqu'un événement souris se produira. Constitue un événement souris tout appui ou tout relâchement de bouton, ou tout déplacement dépassant dans l'un des axes de coordonnées un seuil donné. En l'absence de seuil, tout bit de résolution provoque un envoi vers le processeur maître.

On notera que le clavier intelligent peut retourner une position relative de la souris avec un incrément horizontal ou vertical significativement plus élevé que le seuil. Cela arrivera lorsqu'aucun compte-rendu de positionnement relatif de la souris n'aura été renvoyé : a) parce que le clavier a été inhibé (les événements seront stockés jusqu'à ce que la communication avec le clavier reprenne) b) parce qu'un événement est en train d'être transmis.

Le descripteur de positionnement relatif de la souris est un bloc constitué de trois octets (peu importe le mode clavier):

```
%1111 10xx          drapeau de positionnement relatif
        ||
        | ---------- état du bouton droit
        ---------- état du bouton gauche
deltaX              décalage horizontal signé
deltaY              décalage vertical signé
```

On notera que la valeur des bits d'état des boutons doit être valide même si le MODE D'ACTIVATION SOURIS a été positionné pour que les boutons soient considérés comme partie intégrante du clavier.

Si le décalage cumulé des déplacements horizontaux ou verticaux de la souris déborde le domaine +127 à -128, le compte-rendu de positionnement relatif est effectué sur plusieurs blocs descriptifs successifs.

On notera que le signe du décalage vertical deltaY est fonction de l'origine verticale choisie.

### 4.2. Positionnement absolu

Le clavier intelligent a la possibilité de rendre compte de la position de la souris en coordonnées absolues. Des commandes existent pour réinitialiser la position de la souris, définir les échelles horizontale et verticale, et demander la position courante de la souris.

### 4.3. Mode déplacement curseur

Le clavier intelligent peut convertir les actions sur la souris en équivalences clavier. Le nombre d'appuis de touche correspondant à une action peut être spécifié différemment pour chaque axe. Le clavier intelligent fournit des informations sur les actions souris selon la plus haute résolution valable et génère simplement une paire d'événements appui de touche pour chaque multiple du facteur d'échelle.

Ce mode de fonctionnement provoque l'envoi du code de relâchement de touche immédiatement après le code d'appui de touche. Les appuis de bouton de la souris provoquent également l'envoi de codes d'appui et de relâchement de touche correspondant à ceux qui leur sont normalement assignés dans la table des codes clavier (soit 0x74 pour le bouton droit et 0x75 pour le gauche).

## 5. Manettes de jeu

### 5.1. Compte-rendu d'événement manette de jeu

Dans ce mode, le clavier intelligent génère un bloc descripteur chaque fois que la position de la manette de jeu a changée (c'est-à-dire pour chaque ouverture ou fermeture d'un interrupteur ou contact de la manette de jeu).





Le bloc descripteur d'un événement manette de jeu est constitué de deux octets:

```
%1111 111x          désigne un événement manette de jeu
          |
           --------- manette de jeu 0 ou 1
%x000 yyyy
  |    ||||
  |     ------------ position de la manette
   ----------------- état du bouton de feu
```

### 5.2. Interrogation de manette de jeu

L'état courant des ports manette de jeu peut être interrogé à n'importe quel moment dans ce mode en envoyant une commande d'"interrogation manette de jeu" vers le circuit clavier.

La réponse du clavier intelligent à cette interrogation prend la forme d'un bloc de trois octets ayant le sens suivant :

```
0xFD                en-tête de réponse manette de jeu
%x000 yyyy          manette de jeu 0
%x000 yyyy          manette de jeu 1
```

où x correspond à l'état du bouton de feu et yyyy à la position de la manette.

### 5.3. Surveillance manette de jeu

Ce mode permet de consacrer presque toute l'activité du circuit clavier au compte-rendu des états successifs des manettes de jeu selon une cadence définie par l'utilisateur. Le clavier reste dans ce mode jusqu'à une commande de reset ou une commande le placant dans un autre mode. Dans ce mode, la commande PAUSE ne stoppe pas seulement les communications entre le clavier et le processeur mais également la scrutation des manettes de jeu (les informations ne sont pas empilées).

### 5.4. Surveillance Bouton de feu

Ce mode permet de dédier le circuit clavier au seul contrôle du bouton de feu de la manette de jeu. Dans ce mode, l'état du bouton de feu est testé à la fréquence maximale autorisée par la vitesse de transmission série. Les données sont compactées sur huit bits pour envoi au processeur maître. Le circuit clavier reste dans ce mode jusqu'à une commande de reset ou une commande de changement de mode. Dans ce mode la commande PAUSE ne stoppe pas seulement l'envoi des informations du clavier vers le processeur mais arrête aussi temporairement la scrutation du bouton de mise à feu (les informations ne sont pas empilées).

### 5.5. Mode Clavier des manettes de jeu

Le circuit clavier peut être commandé pour que les actions

sur les manettes de jeu soient converties en équivalents clavier (touches curseur).

Les événements manette de jeu produisent un code d'appui de touche immédiatement suivi d'un code de relâchement, de la même façon que pour les événements souris. Les boutons de feu des manettes de jeu ont également leur équivalents clavier, codes juste supérieurs à ceux de la matrice clavier (soit 0x74 pour la manette 0 et 0x75 pour la manette 1).

## 6. Horloge calendrier

Le clavier intelligent contrôle également une horloge calendrier pour le système. Des commandes sont disponibles pour remettre à jour et interroger cette horloge calendrier. La mise à jour est garantie pour une précision de une seconde.

## 7. Demandes d'état

L'état courant du clavier, les modes actifs et les paramètres peuvent être obtenus par des commandes de demande d'état correspondant à des commandes du clavier intelligent.

## 8. Mise sous tension

Le contrôleur clavier effectue un auto-test simple à la mise sous tension afin de détecter des erreurs majeures (somme de contrôle des mémoires mortes et test des mémoires vives) et divers problèmes comme des touches coincées. Toute touche appuyée lors de la mise sous tension est considérée comme coincée et son code de relâchement est renvoyé (ce qui, en l'absence de code d'appui le précédant, est considéré comme une erreur clavier). Si l'auto-test du contrôleur s'est déroulé sans erreur, le code 0xF0 est renvoyé. (La première version du clavier intelligent était la version 0xF0, les suivantes auront les numéros 0xF1, etc.)

Les options par défaut du clavier sont les suivantes :
1) Compte-rendu du positionnement de la souris en mode relatif avec un incrément d'une unité pour chaque axe.
2) Origine verticale Y=0 correspondant au haut de l'écran.
3) Mode compte-rendu d'événement pour les manettes de jeu.
4) Deux boutons pris en compte pour la souris.

Après toute commande manette de jeu, le circuit clavier considère que les manettes de jeu sont connectées aux ports 0 et 1 des manettes de jeu. Toute commande souris (sauf la DESACTIVATION DE SOURIS) entraîne la scrutation du port 0 comme étant le port relié à la souris et les deux boutons dépendant de celle-ci. Si une commande de désactivation de la souris est reçue alors que le port 0 est logiquement lié à une souris, le bouton est alors assigné à la manette 1.





## 9. Jeu des commandes clavier

Cette partie comprend la liste des commandes qui peuvent être envoyées au clavier intelligent. Les codes de commande (comme 0x00) qui ne sont pas développés ici ne provoquent aucune opération (ils peuvent être assimilés à des NOP).

### 9.1. RESET

0x80
0x01

REMARQUE: La commande RESET est la seule commande clavier qui comprenne deux octets en dehors des paramètres. Tout octet suivant 0x80, autre que 0x01, sera ignoré (et provoquera la non prise en compte de la commande 0x80).

Un reset du circuit clavier peut également être provoqué en envoyant un break durant au moins 200ms vers le circuit clavier.

L'envoi d'une commande reset amène la remise des paramètres clavier à leurs valeurs par défaut (voir 8. Mise sous tension). Il n'affecte pas l'horloge calendrier.

Une commande de RESET provoque l'auto-test du circuit clavier. Si ce test est positif, le clavier envoie le code 0xF0 dans un délai de 300ms après la réception de la commande RESET (ou à la fin du break s'il s'agit d'un reset matériel). Le clavier teste ensuite la matrice clavier pour déceler d'éventuelles l'envoi du code break (relâchement) correspondant à cette touche (un code de relâchement survenant sans avoir été précédé par un code d'appui de touche doit être considéré comme une erreur au niveau de la matrice clavier).

Notes du traducteur : De par son architecture matérielle, l'unité centrale du ST est reliée au clavier intelligent via le processeur multifonctions MC68901 et un ACIA MC6850. C'est ce dernier circuit qui assure les communications série entre le MC68000 et le 6801 responsable du clavier. Il en découle que toute commande vers le circuit clavier suppose que l'ACIA est actif et correctement programmé.

A titre d'information, voici l'ensemble des commandes de reset de circuit ACIA et clavier envoyées lors du boot du système par le système d'exploitation :

```
        move.b    #$3,$FFFC00        * reset ACIA MC6850
        move.b    #$96,$FFFC00       * division horloge/16, 8 bits
                                     * sans parité, 1 bit stop,etc.
        move.l    #_commande,-(sp)   * ptr commande reset et modes
        move.w    #3,-(a7)           * 4 octets à envoyer
        bsr       _Ikbdws            * fonction $19 du trap 14
_commande  dc.b     $80,$01,$12,$1A  * reset, désactive souris et
                                     * manettes de jeu
```

## 9.2. FIXE LE MODE DE PRISE EN COMPTE DES BOUTONS DE SOURIS

```
0x07
%0000 0mss          mode de prise en compte
        |||         (m est supposé à 1 en mode code CLAVIER
        |||          de la souris)
        ----------- 0wx    l'appui ou le relâchement de
                    ||     bouton provoque un compte-rendu
                    ||     de position souris (wx n'est
                    ||     significatif que si la souris
                    ||     est en positionnement absolu)
                    | --- 1 -> la pression du bouton pro-
                    |           voque un compte-rendu de
                    |           la position absolue
                     ---- 1 -> le relâchement du bouton
                                provoque un compte-rendu de
                                la position absolue
                100 : les boutons de la souris sont
                      considérés comme des touches.
```

Cette commande définit la façon d'agir des boutons de la souris, plus précisément la façon dont ils seront pris en compte. Le mode de traitement par défaut des boutons de la souris est %00000000, c'est-à-dire que les boutons sont considérés comme partie intégrante de la souris.

## 9.3. FIXE LE MODE DE POSITIONNEMENT RELATIF DE LA SOURIS

0x08

Place la souris en mode relatif (mode PAR DEFAUT), c'est à dire que chaque déplacement de la souris provoque l'envoi d'un descripteur comprenant les décalages horizontaux et verticaux relatifs à la précédente position. Les blocs descripteurs de positionnement sont générés de façon asynchrone par le clavier intelligent lorsqu'un déplacement selon l'un des axes dépasse le seuil minimal fixé (voir 9.6. SEUIL DECALAGE SOURIS). Selon le mode fixé par la commande précédente, des descripteurs de position peuvent également être envoyés lorsqu'un des boutons de la souris est pressé ou relâché. Dans les autres cas, les boutons de la souris sont considérés comme des touches clavier.





## 9.4. FIXE LE MODE DE POSITIONNEMENT ABSOLU DE LA SOURIS

0x09
Xsup  position horizontale maximale (en unités de
Xinf  déplacement) codée sur deux octets
Ysup  position verticale maximale (en unités de
Yinf  déplacement) codée sur deux octets

Les déplacements de souris seront décrits en positionnement absolu. Les coordonnées horizontale et verticale sont conservées au reset.

Dans ce mode, les valeurs des coordonnées conservées ne doivent pas sortir de l'intervalle 0 - grand nombre positif défini par la commande. Un déplacement jusqu'à une coordonnée négative sera ignoré. La commande fixe la valeur positive maximale qui peut être atteinte. Toute valeur supérieure est ignorée.

## 9.5. FIXE LA SOURIS EN MODE CLAVIER

0x0A
Xdelta  distance en unités horizontales provoquant
        l'envoi du code (LEFT) ou du code (RIGHT)
Ydelta  distance en unités verticales provoquant
        l'envoi du code (UP) ou du code (DOWN)

Place la souris en mode clavier à la place du mode position relative ou position absolue. Après chaque déplacement de souris excédant le seuil horizontal ou vertical fixé par cette commande, le circuit clavier envoie l'équivalent d'un appui de touche curseur du clavier, puis le code de relâchement de touche. A noter que cette commande n'est pas affectée par l'origine des déplacements souris.

## 9.6. FIXE LE SEUIL DE DEPLACEMENT SOURIS

0x0B
Xseuil  seuil horizontal en unités de déplacement
Yseuil  seuil vertical en unités de déplacement

Cette commande fixe le seuil à partir duquel un déplacement de la souris sur l'axe horizontal ou vertical provoquera l'envoi par le circuit clavier d'un descripteur de position souris. On notera qu'elle n'affecte pas la résolution des données renvoyées au processeur. Cette commande est UNIQUEMENT valide en mode de positionnement relatif de la souris. Les seuils horizontaux et verticaux sont fixés à 1 par défaut lors du RESET (ou à la mise sous tension).

NDT : Une unité de déplacement de la souris correspond environ à 0,2 mm, cela se traduit au bureau GEM par un déplacement d'un pixel.

## 9.7. FIXE L'ECHELLE DE LA SOURIS

```
0x0C
Xéchelle        nombre d'unités de déplacement par X interne
Yéchelle        nombre d'unités de déplacement par Y interne
```

Cette commande définit le facteur réducteur à appliquer aux déplacements horizontaux et verticaux. Elle est valide UNIQUEMENT en mode de positionnement absolu de la souris. Dans ce mode, un nombre déterminé d'unités de déplacement horizontal ou vertical doit survenir pour qu'un décalage interne de une unité soit enregistrée par le contrôleur clavier.

Il est important de relever que cette commande ne fonctionne qu'en mode de positionnement absolu même si l'on a commandé au clavier de rendre compte des appuis ou des relâchements de boutons de la souris (voir commande 0x07, #9.2.).

## 9.8. DEMANDE DE LA POSITION ABSOLUE DE LA SOURIS

```
0x0D

Cette commande renvoie :

0xF7            en-tête de descripteur de position absolue
0000 xxxx
     ||||
     |||`----- bouton droit pressé depuis dernière demande
     ||`------ bouton droit relâché depuis dernière demande
     |`------- bouton gauche pressé depuis dernière demande
     `-------- bouton gauche relâché depuis dernière demande
Xsup
Xinf            coordonnée horizontale sur deux octets
Ysup
Yinf            coordonnée verticale sur deux octets
```

La commande de demande de la position absolue de la souris est UNIQUEMENT valide en mode de positionnement absolu de la souris, peu importe le mode de traitement des boutons fixé par une commande 0x07 (voir #9.2.).





## 9.9. DEFINIT LA POSITION DE LA SOURIS

```
0X0E
0X00        octet de remplissage
Xsup
Xinf        position horizontale absolue de la souris
Ysup
Yinf        position verticale absolue de la souris
```

Cette commande permet au programmeur de fixer la position absolue de la souris, horizontale et verticale.

## 9.10. FIXE L'ORIGINE VERTICALE EN BAS

```
0x0F
```

Cette commande fixe l'origine de l'axe des ordonnées en bas du système de coordonnées logiques utilisé par le circuit clavier, tant en mode de positionnement absolu que relatif. Dans ce mode, le déplacement de la souris vers l'utilisateur se traduit par un décalage négatif, et un déplacement dans le sens opposé par un décalage positif.

## 9.11. FIXE L'ORIGINE VERTICALE EN HAUT

```
0x10
```

Cette commande fixe l'origine de l'axe des ordonnées en haut du système de coordonnées logiques utilisé par le circuit clavier, tant en mode de positionnement absolu que relatif. Dans ce mode, le déplacement de la souris vers l'utilisateur se traduit par un décalage positif et un déplacement dans le sens opposé par un décalage négatif. Ce mode est le mode PAR DEFAUT au reset ou à la mise sous tension du système.

## 9.12. REPRISE

Reprend l'envoi des données du circuit clavier vers le processeur. Etant donné que toute commande reçue par le circuit clavier alors qu'il se trouve en mode PAUSE constitue un ordre de REPRISE implicite, cette commande peut être considérée comme inutile. Si le clavier n'est pas en mode PAUSE, cette commande est tout bonnement ignorée.

## 9.13. DESACTIVATION DE LA SOURIS

0x12

Plus aucun compte-rendu d'événement souris n'est envoyé par le circuit clavier au processeur (et la scrutation peut être inactive). Toute commande d'entrée dans un mode souris valide provoque la reprise des envois de descripteurs de position souris. (Les commandes d'entrée dans un mode valides sont : FIXE LE POSITIONNEMENT RELATIF DE LA SOURIS, FIXE LE POSITIONNEMENT ABSOLU DE LA SOURIS et FIXE LA SOURIS EN MODE CLAVIER).

N.B.: Si les boutons de la souris ont été définis comme devant être traités comme des touches clavier, cette commande les AFFECTE également (c'est-à-dire qu'ils deviennent inactifs).

## 9.14. PAUSE

0x13

Stoppe l'envoi de données du clavier vers le processeur jusqu'à ce qu'une autre commande soit envoyée au clavier. L'activité du clavier est toujours scrutée et les codes de scrutation de même que les codes ASCII sont empilés (jusqu'au maximum supporté par le microcontrôleur, c'est-à-dire 8 octets) afin d'être envoyés au processeur lorsque celui-ci demandera la reprise des envois. Si l'on se trouve en mode EVENEMENT MANETTE DE JEU, les événements manette sont également empilés.

Les actions sur la souris sont prises en compte tant que l'on se trouve en mode PAUSE. Si le clavier est en mode POSITIONNEMENT RELATIF de la souris, les événements souris sont cumulés en fonction des seuils de décalage maxima, afin de produire le minimum de blocs descripteurs possibles (les seuils de décalage maxima s'étendent de -128 à +127 unités de déplacement). L'appui ou le relâchement d'un bouton de la souris amène tout événement cumulé à être stocké dans un bloc descripteur, si l'on se trouve en mode de POSITIONNEMENT RELATIF de la souris.

Etant donné que le tampon de stockage d'événements du circuit clavier n'est que de 8 octets, il est recommandé d'user de ce mode avec discernement, les communications entre le clavier et le processeur ne devant pas être figées plus de quelques millisecondes si l'on ne désire pas perdre d'événement.

Les communications sont stoppées seulement après l'envoi en cours, si un envoi est en cours. Le mode PAUSE ne débutera qu'une fois le bloc descripteur complétement envoyé.

Lorsque le circuit clavier se trouve en mode SURVEILLANCE DE MANETTE ou SURVEILLANCE DE BOUTON DE FEU, la commande PAUSE stoppe également ce mode de compte-rendu permanent, c'est-à-dire que les événements ne sont plus empilés pour transmission.





## 9.15. FIXE LE MODE EVENEMENT MANETTE DE JEU

0x14

Entre en mode compte-rendu d'événement manette de jeu. Il s'agit du mode PAR DEFAUT. Chaque ouverture ou fermeture de contact d'une manette de jeu provoque l'envoi d'un descripteur d'événement manette de jeu.

## 9.16. FIXE LE MODE DEMANDE MANETTE DE JEU

0x15

Désactive le mode EVENEMENT MANETTE DE JEU. Le processeur doit envoyer au circuit une commande de DEMANDE D'ETAT MANETTE DE JEU pour connaître l'état d'une manette.

## 9.17. DEMANDE D'ETAT MANETTE DE JEU

0x16

Renvoie un enregistrement décrivant l'état courant des manettes de jeu. Cette commande est valide que l'on se trouve en mode EVENEMENT MANETTE DE JEU ou en mode DEMANDE MANETTE DE JEU.

## 9.18. SURVEILLANCE MANETTE DE JEU

0x17
vitesse temps d'échantillonnage en centièmes de seconde

Blocs retournés (blocs de 2 octets retournés aussi longtemps que l'on est dans ce mode) :


Dans ce mode, le clavier intelligent ne fait que trois choses: a) communiquer sur la ligne série, b) mettre à jour l'horloge-calendrier, c) surveiller les manettes de jeu. Le taux placé en paramètre définit la fréquence d'échantillonnage.

N.B. : Le programmeur ne peut pas fixer cette fréquence à un taux supérieur à ce que permet la vitesse de communication série pour l'envoi de blocs de deux octets (NDT: la fréquence minimale d'un centième de seconde est admise par le circuit clavier!).

## 9.19. SURVEILLANCE DU BOUTON DE FEU

0x18

| | |
|---|---|
| Valeur retournée: %bbbb bbbb | (tant que l'on est dans ce mode) état du bouton de feu de la manette 1. (huit informations par octet, le premier échantillonnage étant donné dans le bit fort (bit 7)) |

Dans ce mode, le clavier intelligent ne fait que trois choses: a) communiquer sur la ligne série, b) mettre à jour l'horloge-calendrier, c) surveiller le bouton de feu de la manette 1. Le bouton de feu est scruté à une fréquence telle que 8 échantillonnages sont effectués pour un octet envoyé (c'est-à-dire que la vitesse de scrutation est de 8/10° de la vitesse de transmission [NDT: soit une vitesse de scrutation de 6340 interrogations par secondes]). L'intervalle d'échantillonnage reste aussi constant que possible.

## 9.20. FIXE LA MANETTE DE JEU EN MODE CLAVIER

0x19

| | |
|---|---|
| RX | temps (en dizièmes de seconde) nécessaire pour produire un décalage horizontal |
| RY | temps (en dizièmes de seconde) nécessaire pour produire un décalage vertical |
| TX | temps (en dizièmes de seconde) de fermeture du contact avant relâchement produisant un décalage horizontal du curseur |
| TY | temps (en dizièmes de seconde) de fermeture du contact avant relâchement produisant un décalage vertical du curseur |
| VX | temps (en dizièmes de seconde) de fermeture du contact après relâchement produisant un décalage horizontal du curseur |
| VY | temps (en dizièmes de seconde) de fermeture du contact après relâchement produisant un décalage vertical du curseur |

Dans ce mode, la manette de jeu 0 est scrutée et traitée comme s'il s'agissait des touches curseur. A la fermeture de contact, un code d'appui et de relâchement de touche curseur est généré. Puis après un délai (RX ou RY selon le cas), ces codes sont renvoyés selon une fréquence définie par TX et TY. Après réouverture de contact, des codes curseur sont générés selon une fréquence définie par VX et VY. Ceci autorise une gestion de la manette avec auto-répétition.

On notera qu'en plaçant RX et/ou RY à 0, il est possible de désactiver cette auto-répétition. Les valeurs de TX et TY n'ont plus de sens et la génération de codes s'effectue via VX et VY).





## 9.21. DESACTIVATION DES MANETTES DE JEU

0x1A

Désactive la génération d'événements manettes de jeu (la scrutation des manettes peut être désactivée). Toute entrée dans un mode de fonctionnement manettes de jeu provoque la reprise des compte-rendus d'événements manette de jeu (les modes de fonctionnement manette sont MODE EVENEMENT MANETTE DE JEU, MODE DEMANDE MANETTE, SURVEILLANCE MANETTE DE JEU, SURVEILLANCE BOUTON DE FEU ET MANETTES EN MODE CLAVIER).

## 9.22. MISE A JOUR DE L'HORLOGE-CALENDRIER

0x1B

| | |
|---|---|
| AA | année (deux derniers chiffres significatifs) |
| MM | mois |
| JJ | jour |
| hh | heure |
| mm | minute |
| ss | seconde |

Toutes les données concernant la date et l'heure doivent être envoyées vers le clavier au format BCD (binaire codé décimal).

Tout chiffre qui ne rentre pas dans le format BCD ne sera pas pris en compte et ne provoquera pas de modification du champ correspondant de la date ou de l'heure. Cela permet de ne modifier que quelques-unes des caractéristiques de la date ou de l'heure.

## 9.23. DEMANDE DE LA DATE ET DE L'HEURE

0x1C

Valeurs retournées :

| | |
|---|---|
| 0xFC | en-tête d'événement date et heure |
| AA | année (2 chiffres inférieurs) |
| MM | mois |
| JJ | jour |
| hh | heure |
| mm | minute |
| ss | seconde |

Toutes les données relatives à la date et l'heure sont retournées au format BCD (binaire codé décimal).

## 9.24. CHARGEMENT DE DONNEES EN MEMOIRE

```
0x20
adr_sup      adresse de la mémoire vive du contrôleur
adr_inf      où charger les données
nombre       nombre d'octets à charger (0 à 128)
{données}
```

Cette commande permet au processeur de placer des valeurs arbitraires dans la mémoire vive du contrôleur. L'intervalle de temps séparant l'envoi de chaque donnée doit être d'au minimum 20 millisecondes.

## 9.25. LECTURE DE DONNEES EN MEMOIRE

```
0x21
adr_sup      adresse de la mémoire du contrôleur à
adr_inf      partir de laquelle lire les données.
             Valeurs retournées :
                  0xF6      en-tête d'état
                  0x20      code d'accès mémoire
                  donnée    6 octets de données lus en
                  donnée    mémoire à partir de l'adresse
                  donnée    envoyée en paramètre
                  donnée
                  donnée
                  donnée
```

Cette commande permet au processeur de lire le contenu des mémoires du contrôleur. Elle permet de lire aussi bien le contenu de la mémoire morte que de la mémoire vive.

## 9.26. EXECUTION DE ROUTINE CONTROLEUR

```
0x22
adr_sup      adresse de la routine en mémoire morte du
adr_inf      contrôleur que l'on désire faire exécuter.
```

Cette commande permet au processeur d'ordonner l'exécution d'un sous-programme résident dans la mémoire morte du contrôleur clavier.

N.D.T.: Les fonctions 0x20 et 0x22 sont d'un usage périlleux car elles nécessitent une bonne connaissance de la topographie mémoire du 6301, contrôleur clavier. Des informations sur le contenu de cette mémoire et les registres de travail du 6301 peuvent être obtenus dans l'ouvrage "Au coeur de l'Atari ST" (cf. bibliographie en annexe).





## 9.27. DEMANDES D'ETAT

Les commandes de demande d'état sont constituées de l'octet de commande de positionnement dans cet état avec le bit 7 placé à 1 (OU logique avec 0x80). Exemple :

```
0x88 (ou 0x89 ou 0x8A)      demande de mode souris

     Valeurs retournées :
          0xF6      en-tête de réponse d'état
          mode      0x08   si RELATIF
                    0x09   si ABSOLU
                    0x0A   si CLAVIER
          param1    0      si mode RELATIF
                    Xsup   si mode ABSOLU
                    Xdelta si mode CLAVIER
          param2    0      si mode RELATIF
                    Xinf   si mode ABSOLU
                    Ydelta si mode CLAVIER
          param3    0      si mode RELATIF
                    Ysup   si mode ABSOLU
                    0      si mode CLAVIER
          param4    0      si mode RELATIF
                    Yinf   si mode ABSOLU
                    0      si mode CLAVIER
          0                bourrage
          0
```

Les commandes de DEMANDE D'ETAT interrogent le clavier sur le mode courant ou les paramètres associés à un mode donné. Tous les compte-rendus d'état sont formatés dans des blocs de 8 octets. Les réponses à des demandes d'état sont fournies de telle sorte qu'elles puissent être stockées (après réception de l'octet $F6 d'en-tête de réponse état) et renvoyées par la suite sous forme de commandes de positionnement d'état par le processeur afin de restaurer un état donné. Les octets nuls de remplissage seront alors traités comme des commandes inopérantes par le circuit clavier.

N.D.T.: Pour traiter les réponses à une demande d'état, il est nécessaire de détourner le vecteur réponse d'état, dont on obtient l'emplacement par Kbdvbase (Trap 14, fonction $22), en ajoutant $C à l'adresse de retour de cette fonction (voir bios étendu).

En plaçant sa propre routine dans ce vecteur, on reçoit dans A0 l'adresse de départ du bloc d'octets réponse de la demande d'état et dans A1 l'adresse de fin plus un de cette réponse.

D'une façon générale, toutes les réponses du clavier passent par l'un des vecteurs pointés indirectement par Kbdvbase et nécessitent donc un traitement particulier par des routines utilisateur.

Les commandes valides de DEMANDE D'ETAT sont les suivantes:

| | |
|---|---|
| 0x87 | mode d'action des boutons de la souris |
| 0x88<br>0x89<br>0x8A | mode de positionnement de la souris<br>(plus précisément mode de traitement des<br>déplacements de la souris par le circuit clavier) |
| 0x8B | seuil de décalage de la souris |
| 0x8C | échelle de décalage de la souris (en ABSOLU) |
| 0x8F<br>0x90 | demande du type d'origine verticale de la souris<br>(renvoie 0x0F si en bas, 0x10 si en haut) |
| 0x92 | activité de la souris (réponse : 0x00 si active,<br>0x12 si inactive) |
| 0x94<br>0x95<br>0x99 | demande du mode manette de jeu |
| 0x9A | activité des manettes de jeu (réponse : 0x00 si<br>actives, 0x1A si désactivées) |

Il est de la responsabilité du programmeur de n'avoir qu'une réponse à une demande en même temps (autrement dit, il faut traiter la réponse à une demande avant d'en faire une seconde, les réponses n'étant pas empilées).

Les commandes de DEMANDE D'ETAT ne sont pas prises en compte si le clavier intelligent se trouve en mode SURVEILLANCE MANETTE DE JEU ou en mode SURVEILLANCE BOUTON DE FEU.





## 10. Appendice A -- Codes de scrutation clavier

Les codes de scrutation clavier retourné par le clavier ont été choisi afin de simplifier la mise en oeuvre du GEM.

N.D.T.: Le traducteur s'est permis de fournir les codes clavier correspondant à la matrice de clavier française en sus des codes correspondant à la matrice de clavier anglaise.

| Valeur hexa. | Clavier anglais | Clavier français |
|---|---|---|
| 00 | (inutilisée) | (inutilisée) |
| 01 | Esc | Esc |
| 02 | 1 | 1 |
| 03 | 2 | 2 |
| 04 | 3 | 3 |
| 05 | 4 | 4 |
| 06 | 5 | 5 |
| 07 | 6 | 6 |
| 08 | 7 | 7 |
| 09 | 8 | 8 |
| 0A | 9 | 9 |
| 0B | 0 | 0 |
| 0C | - | ) |
| 0D | == | - |
| 0E | Backspace | Backspace |
| 0F | Tab | Tab |
| 10 | Q | A |
| 11 | W | Z |
| 12 | E | E |
| 13 | R | R |
| 14 | T | T |
| 15 | Y | Y |
| 16 | U | U |
| 17 | I | I |
| 18 | O | O |
| 19 | P | P |
| 1A | [ | [ |
| 1B | ] | ] |
| 1C | Return | Return |
| 1D | Control | Control |
| 1E | A | Q |
| 1F | S | S |
| 20 | D | D |
| 21 | F | F |
| 22 | G | G |
| 23 | H | H |
| 24 | J | J |
| 25 | K | K |
| 26 | L | L |
| 27 | ; | M |
| 28 | ' | ù |
| 29 | ` | ` |
| 2A | Shift (gauche) | Shift (gauche) |
| 2B | \ | \ |

| | | |
|---|---|---|
| 2C | Z | W |
| 2D | X | X |
| 2E | C | C |
| 2F | V | V |
| 30 | B | B |
| 31 | N | N |
| 32 | M | , |
| 33 | , | ; |
| 34 | . | : |
| 35 | / | = |
| 36 | Shift (droit) | Shift (droit) |
| 37 | (inutilisée) | (inutilisée) |
| 38 | Alternate | Alternate |
| 39 | Barre d'espacement | Barre d'espacement |
| 3A | CapsLock | CapsLock |
| 3B | F1 | F1 |
| 3C | F2 | F2 |
| 3D | F3 | F3 |
| 3E | F4 | F4 |
| 3F | F5 | F5 |
| 40 | F6 | F6 |
| 41 | F7 | F7 |
| 42 | F8 | F8 |
| 43 | F9 | F9 |
| 44 | F10 | F10 |
| 45 | (inutilisée) | (inutilisée) |
| 46 | (inutilisée) | (inutilisée) |
| 47 | Home | Home |
| 48 | Flèche vers le haut | Flèche vers le haut |
| 49 | (inutilisée) | (inutilisée) |
| 4A | Pavé numérique - | Pavé numérique - |
| 4B | Flèche vers la gauche | Flèche vers la gauche |
| 4C | (inutilisée) | (inutilisée) |
| 4D | Flèche vers la droite | Flèche vers la droite |
| 4E | Pavé numérique + | Pavé numérique + |
| 4F | (inutilisée) | (inutilisée) |
| 50 | Flèche vers le bas | Flèche vers le bas |
| 51 | (inutilisée) | (inutilisée) |
| 52 | Insert | Insert |
| 53 | Delete | Delete |
| 54 | (inutilisée) | (inutilisée) |
| 55 | (inutilisée) | (inutilisée) |
| 56 | (inutilisée) | (inutilisée) |
| 57 | (inutilisée) | (inutilisée) |
| 58 | (inutilisée) | (inutilisée) |
| 59 | (inutilisée) | (inutilisée) |
| 5A | (inutilisée) | (inutilisée) |
| 5B | (inutilisée) | (inutilisée) |
| 5C | (inutilisée) | (inutilisée) |
| 5D | (inutilisée) | (inutilisée) |
| 5E | (inutilisée) | (inutilisée) |
| 5F | (inutilisée) | (inutilisée) |
| 60 | touche ISO (inutilisée) | < |
| 61 | Undo | Undo |





| | | |
|---|---|---|
| 62 | Help | Help |
| 63 | Pavé numérique ( | Pavé numérique ( |
| 64 | Pavé numérique ) | Pavé numérique ) |
| 65 | Pavé numérique / | Pavé numérique / |
| 66 | Pavé numérique * | Pavé numérique * |
| 67 | Pavé numérique 7 | Pavé numérique 7 |
| 68 | Pavé numérique 8 | Pavé numérique 8 |
| 69 | Pavé numérique 9 | Pavé numérique 9 |
| 6A | Pavé numérique 4 | Pavé numérique 4 |
| 6B | Pavé numérique 5 | Pavé numérique 5 |
| 6C | Pavé numérique 6 | Pavé numérique 6 |
| 6D | Pavé numérique 1 | Pavé numérique 1 |
| 6E | Pavé numérique 2 | Pavé numérique 2 |
| 6F | Pavé numérique 3 | Pavé numérique 3 |
| 70 | Pavé numérique 0 | Pavé numérique 0 |
| 71 | Pavé numérique . | Pavé numérique . |
| 72 | Pavé numérique Enter | Pavé numérique Enter |

N.D.T. : Rappelons que les codes hexadécimaux 0x74 et 0x75 sont utilisés par le contrôleur clavier pour la simulation des appuis de boutons de feu des manettes de jeu.

# L E S  S P E C I F I C A T I O N S  M I D I

**(Version 1.0)**

Norme MIDI (Musical Instrument Digital Interface)

Spécification 1.0

IMA The International MIDI Association
11857 Hartsook St., North Hollywood, California 91607, USA,
(818)505-8964

## 1. INTRODUCTION

MIDI permet aux synthétiseurs, séquenceurs, ordinateurs, boites à rythmes, etc.. d'être interconnectés à travers une interface standard.

Chaque instrument équipé de prises MIDI contient normalement un émetteur et un récepteur. Certains instruments peuvent ne contenir qu'un émetteur ou qu'un récepteur. L'émetteur génère les messages au format MIDI et les transmet par l'intermédiaire d'un UART (Universal Asynchronous Receiver-Transmitter) et d'un amplificateur de ligne. Le récepteur reçoit les messages au format MIDI et exécute les commandes correspondantes. Il comprend un optocoupleur, un UART et les circuits nécessaires à l'exécution des commandes.

Cette spécification définit le format des données et les circuits du standard MIDI.

CONVENTIONS

Les octets d'état ("status") et de donnée des tableaux 1 à 6 sont fournis en binaire.
Les nombres suivis de "H" sont en hexadécimal.
Tous les autres nombres sont en décimal.

## 2. CIRCUITS MIDI

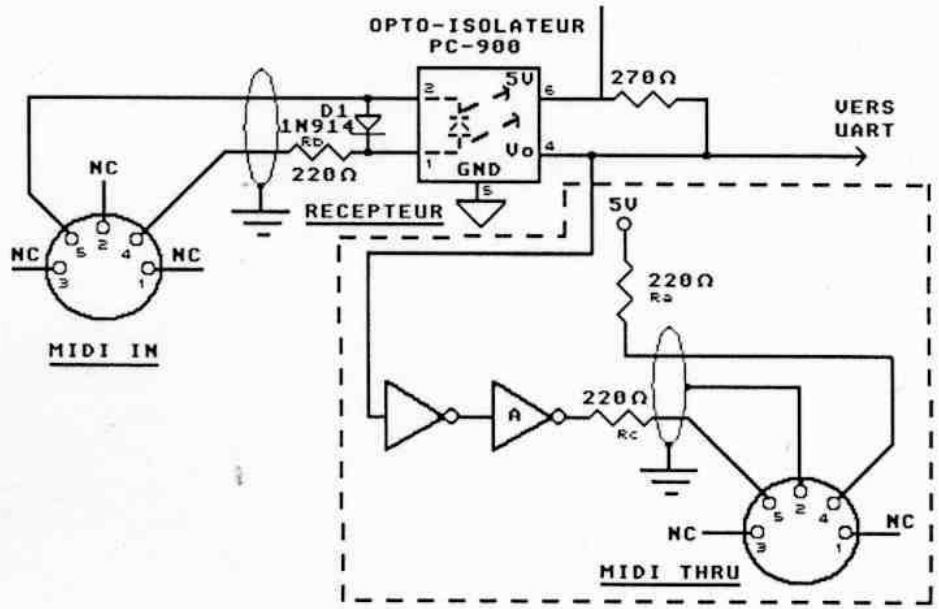
L'interface fonctionne en asynchrone à 31,25 Kbaud(+/- 1%).
Le format de transmission est constitué de mots de 10 bits: un bit de start, huit bits de données(D0 à D7) et un bit de stop. Soit un temps d'émission de 320 microsecondes pour chaque octet de donnée.

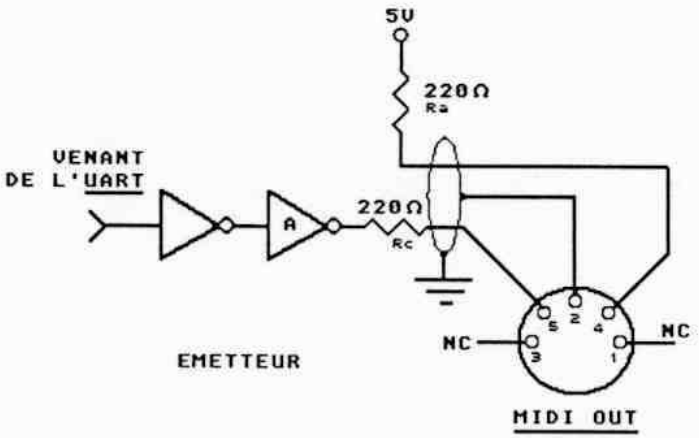
Schéma:
La figure 1 donne le modèle de la boucle de courant 5mA utilisée. Le passage du courant correspond à un niveau logique 0 sur la broche 5.
Une sortie ne peut alimenter qu'une seule entrée.
Chaque entrée doit etre isolée électriquement à l'aide d'un optocoupleur nécessitant moins de 5mA pour fonctionner.
Les optocoupleurs HP 6N138 et Sharp PC-900 conviennent, mais des coupleurs plus rapides peuvent être utilisés. Leur temps de montée et de descente doivent être inférieurs à 2 microsecondes.





SCHEMA MATERIEL STANDARD DE L'INTERFACE MIDI:




Connecteurs: Socle femelle DIN 5 broches (180 degrés) repérés "MIDI IN" et "MIDI OUT".

Remarque: Les broches 1 et 3 des prises MIDI IN et MIDI OUT sont inutilisées.

N.D.T.: Sur le ST, les broches 1 et 3 de la prise MIDI OUT servent de MIDI THRU. Il est donc important de veiller à ce que les broches 1 et 4 ainsi que les broches 3 et 5 ne soient pas reliées à l'intérieur des prises (ce qui peut arriver avec des cables du commerce).

La longeur des cables utilisés pour le raccordement des instruments ne dépassera pas 15 mètres. On utilisera de la paire torsadée blindée, reliée à chaque extrémité à une prise DIN 5 broches mâle.

Une sortie "MIDI THRU" peut être proposée. Elle donne une recopie des données arrivant sur la prise "MIDI IN".

Si vous souhaitez raccorder plus de trois instruments en série, il est nécessaire d'utiliser des optocoupleurs rapides afin de réduire les retards apportés aux temps de montée et de descente du signal.





## 3. FORMAT DES DONNEES

Le protocole de communication MIDI est réalisé à l'aide de "messages" multi-octets composé d'un octet d'état("status") suivi par un ou deux octets de données, à l'exception des messages exclusifs et temps réel.

TYPES DE MESSAGES:

Les messages sont divisés en deux catégories principales:

Les MESSAGES CANAUX

Ces messages sont adressés spécialement à l'un des 16 canaux à l'aide des quatres bits de poids faible de l'octet d'état. Ils seront reçus par tous les instruments positionnés sur ce canal.

Deux types de message canaux doivent être distingués:

Les messages voix qui servent à contrôler les paramètres musicaux (note on, note off, changement de programme(timbre), pitch, after-touch...).

Les messages modes qui définissent le mode de réaction de l'instrument au message voix (omni, poly, mono).

Les MESSAGES SYSTEME:

Ils sont destinés à tous les instruments raccordés et ne contiennent pas d'indication du numéro de canal.

Il existe trois types de messages système:

Les messages communs (sélection de partition, pointeur de position partition, fin de mode système excusif...).

Les messages temps réel:
Ils ne contiennent qu'un octet d'état (pas d'octet de donnée).Ils peuvent être envoyés à tout instant, même entre les octets d'un message dont l'état est différent. Dans ce cas, le message temps réel est ignoré ou mis en attente jusqu'à ce que processus réception soit revenu à son état initial.

Les messages exclusifs:
Ils sont réservés aux constructeurs pour le contrôle de fonctions spécifiques à leurs instruments.
Ils peuvent contenir un nombre quelconque d'octets de données et sont terminés par un EOX (End of Exclusive) ou n'importe quel autre octet d'état. Ces messages comprennent un code d'identification du constructeur(ID). Si le récepteur ne reconnaît pas le code ID, il ignorera les données du message exclusif. Ainsi chaque utilisateur a accès à toutes les fonctionalités de son instrument.

Les constucteurs doivent publier le format de leur code exclusif et sont seuls habilités à en modifier le contenu.

TYPES DE DONNEES:

Octets d'état:

Ils servent à identifier le type de message et la destination des octets de données qu'ils précèdent. Leur bit de poids fort est toujours à 1.

Sauf pour les messages temps réel, un récepteur sera toujours configuré par rapport au dernier octet d'état reçu, même si celui-ci intervient avant la fin du message précédent.

Etat courant:

Lorsqu'un octet d'état, concernant un message "voix" ou un message "mode", est reçu et traité, le récepteur maintient cet état tant qu'il n'a pas reçu un état différent. Il est ainsi possible d'omettre la répétition d'un même octet d'état, dans la mesure ou le nombre des octets de données est conforme.

Le message sous état courant, se résume alors aux octets de données envoyés dans l'ordre. Ceci permet de réduire la taille des données, en particulier si l'on utilise des "note on" avec vélocité à zéro au lieu de "notes off".

L'état courant sera valide jusqu'à l'arrivée d'un octet d'état différent. L'arrivée de messages temps réel n'interrompt que momentanément l'état courant.

Etat non implémentés:

Les état non implémentés dans un récepteur ainsi que les données s'y rapportant, ne seront pas pris en compte par le récepteur.

Etat non utilisés:

Toutes précautions doivent être prises pour éviter l'émission de ces messages pendant les phases d'arrêt et de mise en route. Ces octets seront ignorés à leur réception.

Octets de données:

Placés directement après un octet d'état, un ou deux octets de données contiennent les valeurs spécifiques à la fonction sélectionnée par l'octet d'état. Les octets de données sont codés sur les sept bits de poids faible de l'octet (le bit de poids fort reste à zéro).

Le nombre d'octets de données et les valeurs utilisées sont indiqués dans les tableaux placés en annexe.

Selon le type d'état, un récepteur attendra le nombre d'octets requis avant d'exécuter la fonction et il ne tiendra pas compte d'octets de données qui ne seraient pas précédés d'un octet d'état correct (à l'exception d'état courant vu plus haut).





MODES CANAUX

Les synthétiseurs contiennent des générateurs sonores appelés "voix". Dans le protocole MIDI, la relation entre les 16 canaux utilisables et l'affectation des voix doit être définie.

Plusieurs messages canaux sont destinés à cet usage: omni on / omni off, poly / mono (voir tableau 3).

omni on / omni off:

En mode omni on, le récepteur reconnaîtra tous les messages "voix" contenus dans tous les canaux (en ignorant leur numéro), tandis qu'en mode "omni off", il tiendra compte du numéro de canal contenu dans l'octet d'état, n'acceptant ainsi que les messages voix qui lui sont destinés.

poly / mono:

Exclusifs l'un de l'autre, ces messages affectent les voix du synthétiseur pour les faire réagir aux messages voix de façon polyphonique (plusieurs notes à la fois) ou de façon monophonique (une seule note à la fois).

Ces deux fonctions donnent lieu, pour un récepteur calé sur le canal N, à quatre combinaisons possibles:

| | |
|---|---|
| omni on - poly: | messages voix reçus sur tous les canaux et affectés aux voix de manière polyphonique. |
| omni on - mono: | messages voix reçus sur tous les canaux et affectés à une voix monophonique. |
| omni off - poly: | messages voix reçus sur le canal N et affectés aux voix de manière polyphonique. |
| omni off - mono: | messages voix reçus à partir du canal N jusqu'à N + M - 1 et affectés monophoniquement aux voix 1 à M. Le nombre de voix est précisé par le troisième octet du message "mono mode"(voir tableau 3). |

Les quatres combinaisons s'appliquent également aux émetteurs:

Les émetteurs dont on ne peut choisir le canal émettent normalement sur le canal 1 (N=0).

| | |
|---|---|
| omni on - poly: | tous les messages voix sont trasmis sur le canal N. |
| omni on - mono: | les messages voix sont envoyés, pour une voix, sur le canal N. |
| omni off - poly: | les messages voix sont envoyés, pour toutes les voix, sur le canal N. |
| omni off - mono: | Les messages voix, pour les voix 1 à M sont respectivement transmis aux canaux N à N+M-1. |

Un émetteur ou un récepteur MIDI ne peut fonctionner que dans un seul mode à la fois. L'émetteur et le récepteur sont en

général sur le même mode. Si le mode ne peut etre appliqué au récepteur, celui-ci peut ignorer le message et ses données ou se commuter sur un mode de remplacement (en général omni on - poly).

Seuls les messages modes envoyés sur le canal assigné au récepteur, en fonction du mode courant, seront reconnus par le récepteur.

Un récepteur MIDI peut_etre assigné par défaut à un ou plusieurs canaux MIDI par une commande utilisateur. Ainsi, un synthétiseur à huit voix peut-être assigné au canal 1 à la mise sous tension. L'utilisateur peut alors commuter l'instrument pour le configurer comme deux synthétiseurs à quatre voix, chacun assigné à son propre canal. Des messages modes séparés seront alors envoyés à chacun des synthétiseurs comme s'ils étaient des instruments physiquement distincts.

A la mise sous tension, tous les instruments devraient être par défaut en mode "omni on - poly". Sauf pour les états de "note on" et "note off", tous les messages voix devraient être inhibés. Les émissions non conformes devraient être évitées.





TABLEAU 1

Sommaire des octets d'état:

| Etat D7---D0 | nombre d'octets de données | description |
|---|---|---|
| **Message voix** | | |
| 1000nnnn | 2 | événement 'note off' |
| 1001nnnn | 2 | événement 'note on' (vélocité=0 => note off) |
| 1010nnnn | 2 | pression/relâchement touche polyphonique |
| 1011nnnn | 2 | changement contrôle |
| 1100nnnn | 1 | changement programme |
| 1101nnnn | 1 | pression/relâchement touche canal |
| 1110nnnn | 2 | changement molette de réglage |
| **Messages canal** | | |
| 1011nnnn | 2 | sélection du mode canal |
| **Messages système** | | |
| 11110000 | * * * * * | système exclusif |
| 11110sss | 0 à 2 | message commun système |
| 11111ttt | 0 | système temps réel |

NOTES:

nnnn : no. de canal - 1
(0000 pour le canal 1,
0001 pour le canal 2,
.....
1111 pour le canal 16)

| | |
|---|---|
| * * * * *<br>iiiiiii | 0iiiiiii, données, ..., EOX<br>identification |
| sss | 1 à 7 |
| ttt | 0 à 7 |

TABLEAU 2

Messages voix

| état | données | description |
|---|---|---|
| 1000nnnn | | 'note off' (voir note 1) |
| | 0kkkkkkk | 'note pitch' |
| | 0vvvvvvv | vélocité 'note off' |
| 1001nnnn | | 'note on' (voir note 1) |
| | 0kkkkkkk | 'note pitch' |
| | 0vvvvvvv | vélocité |
| | | (note off si vélocité = 0) |
| 1010nnnn | | relâchement de touche polyphonique |
| | 0kkkkkkk | 'note pitch' |
| | 0vvvvvvv | valeur de la pression |
| 1011nnnn | | changement contrôle |
| | 0ccccccc | numéro du contrôle |
| | 0wwwwwww | valeur du contrôle |
| 1100nnnn | | changement de programme |
| | 0ppppppp | valeur de programme |
| 1101nnnn | | relâchement touche canal |
| | 0vvvvvvv | valeur de la pression |
| 1110nnnn | | changement molette(voir note 2) |
| | 0vvvvvvv | octet de poids faible |
| | 0vvvvvvv | octet de poids fort |

remarque:
Il n'est pas nécessaire de renvoyer l'octet d'état tant que celui-ci ne change pas de valeur.

notes:

nnnn — numéro de canal (voir codage tableau 1).

kkkkkkk — hauteur de la note suivant la gamme chromatique. 128 valeurs(0 -> 127), 60 correspond au do du milieu du clavier piano.

vvvvvvv — vélocité du clavier sur 128 valeurs(0 -> 127), valeur fixe de 64 pour les claviers non dynamiques.

note 1 — pour chaque message "note on", il doit être envoyé ultérieurement, un message "note off" de même hauteur sur le même canal.





| | |
|---|---|
| ccccccc | numéro de contrôle: |
| 0 | continuous controller 0 octet fort |
| 1 | continuous controller 1 " (molette de modulation) |
| 2 | continuous controller 2 octet fort |
| 3 | continuous controller 3 octet fort |
| 4->31 | continuous controller 4->31 octet fort |
| 32 | continuous controller 0 octet faible |
| 33 | continuous controller 1 " (molette de modulation) |
| 34 | continuous controller 2 octet faible |
| 35 | continuous controller 3 octet faible |
| 36->63 | continuous controller 4->31 octet faible |
| 64->95 | interrupteurs (on/off) |
| 96->121 | inutilisé |
| 122->127 | réservé pour les messages canaux(voir tableau 3). |

Les contrôleurs ne sont pas spécifiquements affectés, chaque constructeur peut choisir la correspondance entre les numéros logiques et les contrôleurs physiques. La table d'allocation des contrôleurs doit être décrite dans le manuel d'utilisation.

Les contrôleurs dont la valeur est proportionnelle au déplacement sont codés sur deux octets (MSB et LSB). Si la précision ne nécessite qu'un codage sur 7 bits, seul l'octet de poids fort (MSB) est transmis. Sinon la valeur est codée sur 14 bits et l'on envoie l'octet de poids fort, puis l'octet de poids faible (LSB). Si seul le LSB a changé de valeur, il peut être renvoyé sans le MSB.

wwwwwww — position du contrôleur(MSB):
sur 128 valeurs(0 -> 127) pour les contrôleurs proportionnels,
sur deux valeurs(0, 127) pour les pédales (les valeurs intermédiaires sont ignorées).

ppppppp — numéro de programme (0 -> 127).

note 2 — La sensibilité de la molette de hauteur (pitch bender) est choisie par le récepteur. La position centrale de la molette (pas de variation de hauteur) correspond à la valeur 2000H et sera transmise sous la forme: EnH-00H-40H.

TABLEAU 3

Messages modes

| état | données | description |
|---|---|---|
| 1011nnnn | | message mode |
| | 0ccccccc | |
| | 0vvvvvvv | |
| nnnn | numéro de canal(voir codage tableau 1). | |
| ccccccc | vvvvvvv | |
| 122 | 0 | contrôle local off |
| 122 | 127 | contrôle local on |
| 123 | 0 | toutes notes off |
| 124 | 0 | omni mode off (toutes notes off) |
| 125 | 0 | omni mode on (toutes notes off) |
| 126 | M | mono mode on (poly mode off, toutes notes off) |
| 127 | 0 | poly mode on (mono mode off, toutes notes off) |

contrôle local: cette commande interne (local off) peut être utilisée pour interrompre la transmission des données entre le clavier d'un synthétiseur et ses générateurs sonores. Le clavier sort sur la prise MIDI OUT et les générateurs peuvent être pilotés par la prise MIDI IN. La commande inverse (local on), rétablit la connexion.

Les messages 123 à 127 génèrent un message "all notes off", interrompant les voix du canal correspondant dans l'état. Excepté le message 123 ("toutes notes off"), ces messages doivent être réservés à des emplois particuliers et en aucun cas servir à couper les notes en cours.

Par conséquent, puisqu'à chaque commande "note on" doit correspondre une commande "note off", les commandes "toutes notes off"(123->127) pourront être ignorées par les récepteurs ne disposant pas de notes tenues.

M compris entre 1 et 16, précise le nombre de canaux pour lesquels des messages voix monophoniques vont être envoyés. On utilisera alors les canaux à partir du canal de base(nnnn), jusqu'au canal nnnn + M - 1, avec un maximum de 16. Si M = 0, le récepteur reconnaîtra, en mode mono, un nombre de canaux alignés à partir de nnnn, sur le nombre de voix dont il dispose.





TABLEAU 4

Messages communs

| état | données | description |
|---|---|---|
| 11110001 | | inutilisé |
| 11110010 | | pointeur de position partition |
| | 01111111 | octet faible |
| | 0hhhhhhh | octet fort |
| 11110011 | | sélection de partition |
| | 0sssssss | numéro de partition |
| 11110100 | | inutilisé |
| 11110101 | | inutilisé |
| 11110110 | sans | demande d'accord |
| 11110111 | sans | EOX(end of system exclusive) |

notes:

pointeur de position de partition: Ce pointeur est un registre interne qui contient le nombre de battements MIDI(un battement correspond à six impulsions d'horloge MIDI) depuis le début du morceau.

Le registre, mis à zéro par la commande START au début de l'exécution du morceau, est incrémenté toutes les six impulsions d'horloge jusqu'à l'apparition de la commande STOP. La commande CONTINUE relance l'incrémentation.

Les deux octets de données permettent de remplir le registre et de positionner le pointeur à une position quelconque du morceau.

sélection de partition: précise quel morceau ou séquence va être joué à la réception de la commande START (message temps réel).

demande d'accord: demande aux synthétiseurs analogiques d'accorder leurs oscillateurs sur leur fréquence de référence.

EOX: drapeau utilisé pour signaler la fin de transmission de données en système exclusif(voir tableau 6).

TABLEAU 5

Messages temps réel

| état | données | description |
|---|---|---|
| 11111000 | | cadencement horloge |
| 11111001 | | inutilisé |
| 11111010 | | début ("start") |
| 11111011 | | poursuite ("continue") |
| 11111100 | | arrêt ("stop") |
| 11111101 | | inutilisé |
| 11111110 | | détection active ("active sensing") |
| 11111111 | | réinitialisation système ("reset") |

notes:

Les messages temps réel permettent de synchroniser simultanément tous les instruments raccordés à un instrument maître (séquenceur ordinateur ou boite à rythme).

Chaque message, composé d'un octet, peut être envoyé à un instant quelconque et transiter entre deux octets d'un autre type de message.

cadencement horloge (F8H): cet octet sert d'horloge au système pour la synchronisation des instruments. Il est envoyé 24 fois pendant le temps correspondant à la durée d'une noire.

début (FAH): cet octet est envoyé après appui sur la touche PLAY de l'instrument maître. Il caractérise le début du morceau ou de la séquence.

poursuite(FBH): cet octet est envoyé après appui sur la touche CONTINUE de l'instrument maître. Il redémarre, sur l'octet d'horloge suivant, la séquence interrompue.

arrêt (FCH): cet octet est envoyé après appui sur la touche STOP de l'instrument maître. Il stoppe l'éxécution de la séquence.

détection active (FEH): cet octet, dont l'utilisation est optionnelle, permet de détecter l'éventuelle déconnexion d'un des instruments reliés à l'instrument maitre. Après un premier envoi, qui sensibilise les récepteurs, il doit être envoyé de façon régulière, au moins toutes les 300 millisecondes.

Si un récepteur ne reçoit aucun octet pendant cet intervalle de temps, il conclut à une coupure de ligne, arrête les notes en





cours et revient à sa configuration normale.

réinitialisation système(FFH): à la réception de cet octet, tous les éléments du système reprennent la configuration qu'ils avaient à la mise sous tension. Cet octet ne doit pas être envoyé à la mise sous tension, il est recommandé de limiter son utilisation à une commande manuelle de reset.

TABLEAU 6

Messages "exclusifs"

| état | données | description |
|---|---|---|
| 11110000 | | début de message exclusif. |
| | 0iiiiiii | identification du constructeur. |
| | (0*******) | nombre quelconque d'octets dont |
| | . | le bit de poids fort est à zéro. |
| | . | |
| | . | |
| | (0*******) | |
| | 11110111 | EOX (fin de message exclusif). |

notes:

iiiiii numéro d'identification du constructeur(0->7FH).

| | |
|---|---|
| 01H | Sequential Circuits, Inc. |
| 02H | Big Briar |
| 03H | Octave/Plateau |
| 04H | Moog Music |
| 05H | Passeport designs |
| 06H | Lexicon |
| 20H | Bon Tempi |
| 21H | S.I.E.L. |
| 40H | Kawai |
| 41H | Roland |
| 42H | Korg |
| 43H | Yamaha |

Aucun octet d'état ou de donnée ne pourra être intercalé dans le message exclusif, à l'exception des messages "temps réel".

# Manuel d'utilisation du 'BLITTER'

## (Processeur de transfert de blocs de bits)

**ATARI Corporation**

**Sunnyvale, California**

17 Juin 1987.

Cette documentation se limite à une description fonctionnelle du 'BLITTER' de l'ATARI ST. Il ne constitue ni une notice d'installation du circuit, ni un manuel de programmation de ce circuit. Pour plus d'informations, référez-vous à l'Annexe C en fin de ce chapitre.

## 1. INTRODUCTION

Le processeur de transfert de blocs de bits de l'Atari ST ("Bit-Block Transfer Processor", BLITTER) constitue la traduction matérielle de l'algorithme de transfert de blocs de bits. Cet algorithme peut être décrit comme la méthode de copie d'un bloc de bits source vers un bloc de bits destination à travers une opération logique. La primitive de transfert de blocs peut servir à des opérations comme:

* le remplissage de zone,
* la rotation par découpage récursif,
* le zoom ou la réduction,
* le tracage de ligne par l'algorithme de Bresenham,
* les transformations de texte (ex: gras, italique, souligné)
* le déroulement de texte ("scrolling"),
* le remplissage selon une matrice,

et toutes les fonctions de copie de blocs de mémoire [C.1].

L'essence du transfert de blocs a été définie de façon formelle et pour la première fois par Newman et Sproull dans leur description de la fonction de copie de zones ("RasterOp", C.2). Selon cette définition de base, l'opération de copie effectuait le transfert sur une base de bit à bit et se limitait à un jeu réduit de combinaisons booléennes. Des améliorations de cette fonction comme le transfert de bits en parallèle ou l'introduction d'une matrice demi-teinte ("half-tone") lors du transfert étaient simplement proposées au lecteur en exercices.

Afin d'améliorer les fonctionnalités et d'accroître les performances de l'algorithme d'origine, les caractéristiques décrites plus haut ont été ajoutées à la définition de la fonction copie de zones et implémentées dans le circuit "RasterOp" (C.3). Ce circuit manquait toutefois de la bi-dimensionnalité de la fonction de départ et ses performances se ressentaient du goulot d'étranglement constitué par la lecture et l'écriture des données de la source, de la destination et de la matrice demi-teinte (il ne pouvait accéder à la mémoire par DMA).

Tandis que des progrès étaient accomplis au niveau de l'accélération des fonctions du circuit "RasterOp", la définition formelle de la copie de zones était par ailleurs redéfinie et fournissait la base de la primitive de copie par boucle de blocs de bits dans le noyau du langage graphique Smalltalk-80 (C.4). Du fait de son interface utilisateur très ouverte, la primitive de





transfert de blocs se révéla toutefois peu efficace, des optimisations étant indispensables pour les cas simples, ce qui allait contre la vocation même de globalité de la fonction. Il devenait clair qu'une solution matérielle était indispensable si l'on désirait améliorer les performances de la primitive de copie par boucle sans réduire ses fonctionnalités.

Le 'Blitter' de l'Atari ST est une solution matérielle aux problèmes de performances rencontrés par la fonction de transfert de blocs. Le 'Blitter' est un périphérique DMA (accès direct à la mémoire) qui réalise l'étendue complète des fonctions de copie de blocs de bits en y ajoutant quelques possibilités mineures. Des incréments ou des décréments sur un ou plusieurs mots sont possibles pour des transferts vers la mémoire d'affichage du ST. Un masque de destination, qui pourra être constitué de bits à 1 (ce qui supprime son effet), autorise un niveau supplémentaire de trame. Le reste de cette documentation est directement orienté vers la description fonctionnelle du 'Blitter' de l'Atari ST.

## 2. TRANSFERTS DE BLOCS DE BITS

Comme indiqué précédemment, un transfert de bloc de bits peut être considéré comme une procédure de déplacement de données alignées sur un bit d'une source vers une destination, avec application d'une opération logique lors du transfert. Il existe seize régles de combinaisons logiques applicables au transfert de la source vers la destination. On notera que ces seize régles constituent l'ensemble des opérations logiques possibles lors du transfert. La table ci-dessous fournit les caractéristiques de ces opérations logiques:

OPERATIONS LOGIQUES

| (~s&~d) bit fort | (~s&d) | (s&~d) | (s&d) bit faible | OP | Régle logique |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | tous à zéro |
| 0 | 0 | 0 | 1 | 1 | (source) ET (destination) |
| 0 | 0 | 1 | 0 | 2 | (source) ET (NON destination) |
| 0 | 0 | 1 | 1 | 3 | source |
| 0 | 1 | 0 | 0 | 4 | (NON source) ET (destination) |
| 0 | 1 | 0 | 1 | 5 | destination |
| 0 | 1 | 1 | 0 | 6 | (source) EOU (destination) |
| 0 | 1 | 1 | 1 | 7 | (source) OU (destination) |
| 1 | 0 | 0 | 0 | 8 | (NON source) ET (NON destination) |
| 1 | 0 | 0 | 1 | 9 | (NON source) EOU (destination) |
| 1 | 0 | 1 | 0 | A | (NON destination) |
| 1 | 0 | 1 | 1 | B | (source) OU (NON destination) |
| 1 | 1 | 0 | 0 | C | (NON source) |
| 1 | 1 | 0 | 1 | D | (NON source) OU (destination) |
| 1 | 1 | 1 | 0 | E | (NON source) OU (NON destination) |
| 1 | 1 | 1 | 1 | F | tous à un |

Les paramètres de dimensions et de positions des blocs ainsi que les caractéristiques du transfert doivent être initialisés avant le transfert du bloc. Ces paramètres incluent l'affichage restreint ("clipping"), l'oblique, les masques finaux, et le recouvrement.

Affichage restreint ("clipping"): Les dimensions et positions des blocs source et destination sont ajustées pour correspondre au rectangle d'affichage restreint sélectionné. Dans la mesure où les blocs source et destination ont une taille identique, les dimensions du bloc destination sont réduites à celles du bloc source une fois appliqué à ce dernier l'affichage restreint (et vice-versa). On notera que le transfert n'a pas lieu d'être si le bloc résultant est nul.

Oblique: Le décalage de ligne horizontale réalisant l'oblique est calculé lors du transfert de la source vers la destination.





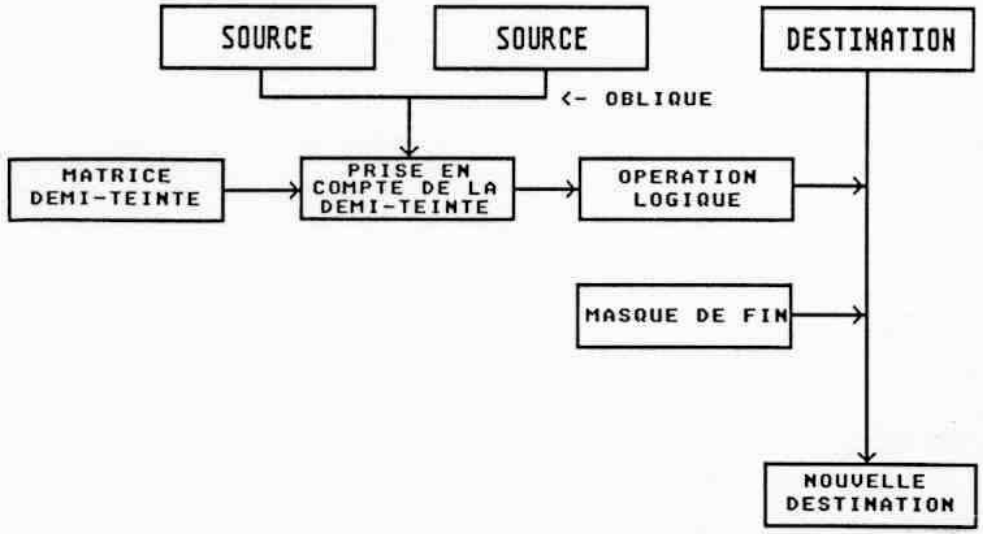
Masques finaux: Les masques partiels des mots de début et de fin sont déterminés. Ces masques sont fusionnés si la destination a une largeur d'un seul mot.

Recouvrement: Les positions des blocs sont comparées pour tester le recouvrement éventuel de ces blocs et éviter la destruction d'une partie du bloc source lors du transfert.

Lors de transferts sans recouvrement, la direction de transfert du bloc source est sans importance et débutera par défaut au sommet en haut à gauche pour se terminer au sommet en bas à droite. Pour des transferts avec recouvrement, la direction de transfert correspond également à la diagonale coin haut gauche vers coin bas droite si l'adresse de la source est supérieure ou égale à l'adresse de la destination. Dans le cas contraire, c'est-à-dire si l'adresse de la source est inférieure à l'adresse de la destination, alors le transfert de données s'effectue à partir du sommet en bas à droite vers le sommet en haut à gauche.

Une fois les paramètres de transfert positionnés, l'opération de transfert de bloc peut débuter. Cette opération s'effectue selon l'opération logique fixée (la demi-teinte et le code de prise en compte de la demi-teinte [HOP] seront étudiés en 4.5).

TRANSFERT DE BLOC DE BITS



## 3. DESCRIPTION FONCTIONNELLE

Veuillez vous référer au diagramme précédent de transfert de bloc de bits pour la compréhension de ce chapitre. Pour percevoir les composants de base d'un transfert, examinons d'abord le cas le plus simple possible de transfert. Nous voulons remplir un bloc de mémoires soit uniquement avec des zéros, soit seulement avec des uns (OP = 0 ou OP = F). Dans ce cas élémentaire, seuls le bloc d'opération logique, qui génère les zéros et les uns, et le bloc de masque final servent au transfert. Si le masque final est uniquement composé de 1, le 'BLITTER' écrira simplement un mot (de 0 ou de 1 selon OP) puis un autre, etc., à partir de l'adresse de la destination sans même prendre connaissance de l'ancien contenu de la destination.

Au fur et à mesure de l'écriture, l'adresse de la destination sera modifiée en accord avec les valeurs des registres d'INCREMENT HORIZONTAL DE DESTINATION, d'INCREMENT VERTICAL DE DESTINATION, de TAILLE HORIZONTALE et de TAILLE VERTICALE. Ces registres définissent la taille et la forme du bloc à transférer. Les registres de TAILLE HORIZONTALE et de TAILLE VERTICALE fournissent les dimensions du bloc. Le registre de TAILLE HORIZONTALE spécifie le nombre d'écritures de mots nécessaires pour la mise à jour d'une ligne horizontale. Le registre de TAILLE VERTICALE spécifie le nombre de lignes horizontales du bloc. L'INCREMENT HORIZONTAL DE DESTINATION est un mot signé (16 bits, en complément à 2) qui est ajouté à l'adresse de destination afin d'obtenir l'adresse du mot suivant de la destination. En fin d'écriture de ligne, l'INCREMENT VERTICAL DE DESTINATION est ajouté à l'adresse de la destination afin de pointer le premier mot de la ligne suivante.

Le masque final détermine quels bits de la destination vont être mis à jour. Les bits de la destination correspondant à des 1 dans le masque final sont mis à jour. Les bits de la destination correspondant à des 0 dans le masque final restent inchangés. On notera que même si certains bits de la destination demeurent inchangés, une séquence de lecture-modification-écriture est nécessaire. Afin d'améliorer les performances, seule la lecture sera effectuée. On distingue trois MASQUES FINAUX numérotés de 1 à 3. Le MASQUE FINAL 1 sert uniquement à l'écriture de la première ligne horizontale. Le MASQUE FINAL 3 sert uniquement à l'écriture de la dernière ligne horizontale. Le MASQUE FINAL 2 sert pour toutes les autres lignes.

Maintenant considérons un cas plus complexe, supposons que nous voulions effectuer un EOU ("XOR") du bloc de destination avec une matrice demi-teinte de 16x16 bits. D'abord nous devons charger la mémoire demi-teinte ("Halftone RAM") avec la matrice demi-teinte. La sélection de demi-teinte s'effectue à partir du registre HOP tandis que l'opération logique EOU est fixée par l'opérateur logique OP. Le registre de NUMERO DE LIGNE sert à spécifier lequel des 16 mots de la matrice demi-teinte servira pour la ligne courante. Ce registre doit être incrémenté ou décrémenté en fin de chaque ligne conformément au signe du





registre d'INCREMENT VERTICAL DE LA DESTINATION. Placez les registres d'INCREMENT HORIZONTAL et VERTICAL DE LA DESTINATION ainsi que les registres de TAILLE HORIZONTALE et VERTICALE à leurs valeurs avant le transfert. Cette procédure peut être utilisée quel que soit l'opérateur logique choisi. Elle est également utilisable avec un bloc source au lieu d'une matrice demi-teinte ou en effectuant un ET logique entre le bloc source et la matrice demi-teinte par modification du registre HOP. Un bloc source doit avoir la même taille que le bloc destination mais peut avoir des incréments et une adresse différentes, lesquels sont fixés par les registres d'INCREMENT HORIZONTAL et VERTICAL DE SOURCE et par le registre d'ADRESSE SOURCE.

En conclusion, considérons le cas où les blocs source et destination ne sont pas alignés sur un mot. Dans ce cas, il se peut que l'on doive lire les deux premiers mots de la source dans le tampon source 32 bits et utiliser les seuls 16 bits coïncidant avec les bits de la destination, selon le contenu du registre OBLIQUE. Lorsque le mot suivant de la source est lu, les 16 bits faibles du tampon source sont copiés dans les 16 bits forts et ces 16 bits faibles sont remplacés par le nouveau mot. Ce processus est inversé lorsque la source est lue de la droite vers la gauche (INCREMENT HORIZONTAL DE SOURCE négatif).

Etant donné que peuvent se présenter des cas où il s'avère nécessaire d'effectuer une lecture supplémentaire de la source en début de ligne afin de rafraîchir le tampon de source et d'autres où cela n'est pas indispensable du fait du registre de masque final, un bit de contrôle a été fourni pour permettre cette lecture supplémentaire. Le bit FXSR du registre OBLIQUE indique, lorsqu'il est placé à 1, qu'une lecture supplémentaire de la source est nécessaire en début de ligne afin de rafraîchir le tampon de source. Cette lecture peut ne pas être nécessaire avec certaines combinaisons de masques finaux et d'obliques. Si cette lecture est supprimée, le transfert du mot bas vers le mot haut du tampon source se produira normalement. Dans ce cas, un cycle de lecture-modification-écriture sera effectué sur la destination en fin d'écriture de chaque ligne horizontale sans prise en compte de la valeur du registre de MASQUE FINAL correspondant.

## 4. MODELES DE PROGRAMMATION

Le 'Blitter' contient un ensemble de registres correspondant aux adresses du transfert, aux cadrages des blocs de bits, aux opérations logiques et demi-teinte ainsi qu'aux accès bus. Le temps de positionnement des registres est à peu près constant et relativement important comparativement au temps de transfert de petits blocs, beaucoup plus réduit relativement au temps de transfert de grands blocs.

### 4.1. Carte des registres

Voici la carte des registres programmables du 'Blitter' (notez que les bits inutilisés sont lus comme des 0 et figurent ici sous forme de tirets).

CARTE DES REGISTRES

| | | | |
|---|---|---|---|
| FF8A00 | XXXXXXXX | XXXXXXXX | MEMOIRE DEMI-TEINTE |
| FF8A02 | XXXXXXXX | XXXXXXXX | |
| FF8A04 | XXXXXXXX | XXXXXXXX | |
| .. | .. | .. | |
| FF8A1E | XXXXXXXX | XXXXXXXX | |
| FF8A20 | XXXXXXXX | XXXXXXX- | INCREMENT HORIZONTAL SOURCE |
| FF8A22 | XXXXXXXX | XXXXXXX- | INCREMENT VERTICAL SOURCE |
| FF8A24 | -------- | XXXXXXXX | ADRESSE BLOC SOURCE |
| FF8A26 | XXXXXXXX | XXXXXXX- | |
| FF8A28 | XXXXXXXX | XXXXXXXX | MASQUE FINAL 1 |
| FF8A2A | XXXXXXXX | XXXXXXXX | MASQUE FINAL 2 |
| FF8A2C | XXXXXXXX | XXXXXXXX | MASQUE FINAL 3 |
| FF8A2E | XXXXXXXX | XXXXXXX- | INCREMENT HORIZ. DESTINATION |
| FF8A30 | XXXXXXXX | XXXXXXX- | INCREMENT VERT. DESTINATION |
| FF8A32 | -------- | XXXXXXXX | ADRESSE BLOC DESTINATION |
| FF8A34 | XXXXXXXX | XXXXXXX- | |
| FF8A36 | XXXXXXXX | XXXXXXXX | TAILLE HORIZONTALE |
| FF8A38 | XXXXXXXX | XXXXXXXX | TAILLE VERTICALE |
| FF8A3A | ------XX | | OPERATION DEMI-TEINTE |
| FF8A3B | ----XXXX | | OPERATION LOGIQUE |

FF8A3C |XXX-XXXX|

- |||| : NUMERO DE LIGNE
- ||| : DEBORDEMENT
- || : PARTAGE DU BUS
- | : OCCUPATION DU BUS

FF8A3D |XX--XXXX|

- |||| : OBLIQUE
- || : NFSR
- | : FXSR





### 4.2. Adresses des blocs de bits

Cette section traite des registres définissant les origines des blocs de bits, les incréments d'adresse et les tailles.

ADRESSE BLOC SOURCE
Ce registre de 23 bits contient l'adresse courante du bloc source (seule une adresse paire peut être spécifiée). Il est accessible par le biais d'un adressage sur deux mots ou sur un long mot. La valeur de ce registre correspond toujours à l'adresse du prochain mot de la source devant être traité. Il doit être mis à jour du pas spécifié par les registres d'INCREMENT HORIZONTAL DE LA SOURCE et d'INCREMENT VERTICAL DE LA SOURCE au fur et à mesure du transfert.

INCREMENT HORIZONTAL SOURCE
Ce registre de 15 bits, le bit le plus faible étant ignoré, spécifie le décalage horizontal en octets qui doit être appliqué à l'adresse du bloc source après chaque transfert de mot. Cette valeur est SIGNEE et ajoutée à l'adresse du bloc source après chaque lecture de mot, lorsque la taille horizontale est différente de 1. Si la taille horizontale est égale à 1, ce registre n'est pas pris en compte. Les instructions portant sur un opérande d'un octet ne doivent pas être utilisées pour lire ou écrire ce registre.

INCREMENT VERTICAL SOURCE
Ce registre de 15 bits, le bit le plus faible étant ignoré, spécifie le décalage en octet qui doit être ajouté à l'adresse du premier mot de la source lors d'un changement de ligne. Cette valeur est SIGNEE et ajoutée au registre d'adresse du bloc source une fois arrivé en fin de ligne (donc, lorsque la TAILLE HORIZONTALE est égale à 1). Si le registre de TAILLE HORIZONTALE contient 1, seul ce registre est utilisé. Les instructions portant sur un opérande d'un octet ne doivent pas être utilisées pour lire ou écrire ce registre.

ADRESSE BLOC DESTINATION
Ce registre de 23 bits contient l'adresse courante du bloc DESTINATION (seule une adresse paire peut être spécifiée). Il est accessible par le biais d'une instruction portant sur une taille mot ou long mot. La valeur de ce registre correspond toujours à l'adresse du prochain mot de la DESTINATION devant être traité. Il doit être mis à jour du pas spécifié par les registres d'INCREMENT HORIZONTAL DE LA DESTINATION et d'INCREMENT VERTICAL DE LA DESTINATION au fur et à mesure du transfert.

INCREMENT HORIZONTAL DESTINATION
Ce registre de 15 bits, le bit le plus faible étant ignoré, spécifie le décalage horizontal en octets qui doit être

appliqué à l'adresse du bloc DESTINATION après chaque transfert de mot. Cette valeur est SIGNEE et ajoutée à l'adresse du bloc DESTINATION après chaque lecture de mot, lorsque la taille horizontale est différente de 1. Si la taille horizontale est égale à 1, ce registre n'est pas pris en compte. Les instructions portant sur un opérande d'un octet ne doivent pas être utilisées pour lire ou écrire ce registre.

INCREMENT VERTICAL DESTINATION

Ce registre de 15 bits, le bit le plus faible étant ignoré, spécifie le décalage en octet qui doit être ajouté à l'adresse du premier mot de la DESTINATION lors d'un changement de ligne. Cette valeur est SIGNEE et ajoutée au registre d'adresse du bloc DESTINATION une fois arrivé en fin de ligne (donc, lorsque la TAILLE HORIZONTALE est égale à 1). Si le registre de TAILLE HORIZONTALE contient 1, seul ce registre est utilisé. Les instructions portant sur un opérande d'un octet ne doivent pas être utilisées pour lire ou écrire ce registre.

TAILLE HORIZONTALE

Ce registre de 16 bits spécifie le nombre de mots contenus dans une ligne de la destination. Le nombre minimum est 1 et le maximum est 65536 désigné par 0. Les instructions portant sur un opérande d'un octet ne doivent pas être utilisées pour lire ou écrire ce registre. Ce registre contient toujours le nombre de mots encore à écrire dans la ligne courante, PAS NECESSAIREMENT le nombre placé à l'écriture du registre. Chaque fois qu'un mot du bloc destination est écrit, la valeur de ce registre est décrémentée jusqu'à ce qu'elle atteigne 0, auquel cas le registre est rafraichi avec la valeur de départ.

TAILLE VERTICALE

Ce registre de 16 bits spécifie le nombre de lignes contenues dans le bloc destination. Le nombre minimum est 1 et le maximum est 65536 désigné par 0. Les instructions portant sur un opérande d'un octet ne doivent pas être utilisées pour lire ou écrire ce registre. Ce registre contient toujours le nombre de lignes restant à écrire dans le bloc, PAS NECESSAIREMENT le nombre placé à l'écriture du registre. Chaque fois qu'une ligne du bloc destination est écrite, la valeur de ce registre est décrémentée jusqu'à ce qu'elle atteigne 0, auquel cas le registre est rafraichi avec la valeur de départ.





## 4.3. Les cadrages des blocs de bits

Cette section décrit les registres de spécification des masques finaux, de transfert oblique et de reminiscence des données source.

MASQUES FINAUX 1, 2, 3

Ces registres de 16 bits servent au masquage lors des écritures du bloc destination. Les bits du mot de destination qui correspondent à des 1 dans le masque final seront modifiés. Les bits du mot de destination qui correspondent à des 0 dans le masque final restent inchangés. Le registre MASQUE FINAL courant est déterminé en fonction de la position de la ligne. Le MASQUE FINAL 1 est utilisé uniquement pour la première ligne. Le MASQUE FINAL 3 sert seulement pour la dernière ligne. Le MASQUE FINAL 2 sert pour toutes les autres lignes. Lorsque la ligne est longue d'un seul mot, le MASQUE FINAL 1 est utilisé. Les instructions portant sur un opérande ne doivent pas être utilisées pour lire ou écrire ces registres.

OBLIQUE

Les quatre bits faibles du registre octet, d'adresse $FF8A3D, spécifient l'oblique, c'est-à-dire le décalage à droite devant être appliqué sur les données de la source avant de les combiner avec la matrice demi-teinte et/ou les données de la destination.

FXSR

Abréviation de "Force eXtra Source Read" (Force une lecture supplémentaire de la source). Lorsque ce bit est à 1, une lecture supplémentaire d'un mot de la source est effectuée afin d'initialiser la portion complète de la source à traiter.

NFSR

Abréviation de "No Final Source Read" (Pas de lecture de la source en fin de ligne). Lorsque ce bit est à 1, la lecture de la source lors du dernier mot de chaque ligne n'est pas effectuée. On notera que l'utilisation de ce bit et/ou du précédent nécessite une mise à jour des registres d'INCREMENT VERTICAL DE LA SOURCE et d'ADRESSE DU BLOC SOURCE.

## 4.4. Opérations logiques

Cette section décrit les registres qui spécifient le type de combinaison logique effectué sur les données du bloc source et du bloc destination.

OP

Les quatre bits faibles de ce registre d'un octet, d'adresse $FF8A3B, spécifient la combinaison logique à effectuer entre les bits du bloc source et du bloc destination, en accord avec la table suivante:

OPERATIONS LOGIQUES

| OP | Combinaison logique |
|---|---|
| 0 | tous à 0 |
| 1 | (source) ET (destination) |
| 2 | (source) ET (NON destination) |
| 3 | (source) |
| 4 | (NON source) ET (destination) |
| 5 | (destination) |
| 6 | (source) EOU (destination) |
| 7 | (source) OU (destination) |
| 8 | (NON source) ET (NON destination) |
| 9 | (NON destination) |
| A | (source) OU (NON destination) |
| B | (NON source) |
| C | (NON source) OU (destination) |
| D | (NON source) OU (NON destination) |
| E | (source) ET (destination) |
| F | (source) ET (destination) |

Note: L'opération logique EOU ("XOR") dispose de la table de vérité suivante:

| | | EOU |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |





## 4.5. Opérations demi-teinte

Cette section traite des registres spécifiant les mémoires de matrice demi-teinte, l'index de ligne de matrice et le type de combinaison entre les données de la source et de la matrice.

MEMOIRES MATRICE DEMI-TEINTE

Ces mémoires correspondent au masque de la matrice, soit 16 mots (16x16 bits). Chaque mot correspond à une ligne du bloc destination et il se répète toutes les 16 lignes. Le mot courant est pointé par l'index de ligne de matrice. Ces registres peuvent être lus mais ne peuvent pas être écrits avec des instructions portant sur un octet.

INDEX DE LIGNE DE MATRICE

Les quatre bits faibles de ce registre octet, situé à l'adresse $FF8A3C, permettent d'obtenir le masque de matrice courant. La valeur du registre est égale à l'index relatif plus deux dans la matrice demi-teinte débutant en $FF8A00. Cette valeur est incrémentée ou décrémentée en fin de ligne et remise à jour lorsqu'elle devient nulle. Le signe de l'INCREMENT VERTICAL DE LA DESTINATION définit le sens du pas (incrémentation ou décrémentation).

DEBORDEMENT

Le bit de débordement, lorsqu'il est à 1, provoque l'utilisation des quatre bits faibles des données de la source oblique comme index de l'adresse de la matrice demi-teinte. Remarque: La matrice demi-teinte reste naturellement valide lorsque ce bit est à 1.

TYPE D'OPERATION DEMI-TEINTE (HOP)

Les deux bits faibles de ce registre octet, situé à l'adresse $FF8A3A, spécifient le type de combinaison de la source et de la matrice demi-teinte selon le tableau suivant:

OPERATIONS DEMI-TEINTE

| HOP | Régle de combinaison |
|---|---|
| 0 | tous à 1 |
| 1 | matrice demi-teinte |
| 2 | source |
| 3 | (source) ET (matrice demi-teinte) |

## 4.6. Accès au bus

Cette section décrit les registres de contrôle d'accès au bus et d'état de base du 'Blitter'.

HOG

Le bit HOG, s'il est à 0, provoque un partage équitable de l'accès au bus entre le processeur 68000 et le 'Blitter'. Dans ce mode, chacun dispose de 64 cycles machines, l'autre étant stoppé. Si ce bit est à 1, le processeur 68000 est stoppé jusqu'à ce que le transfert ait pris fin. Dans les deux cas le 'Blitter' cédera le pas aux autres périphériques DMA. L'arbitrage du bus peut permettre au processeur d'exécuter une ou plusieurs instructions même en mode 'hog'. Aussi n'escomptez pas que l'instruction suivant celle qui placera le bit 'BUSY' ne sera exécutée qu'une fois le transfert terminé. Le bit 'BUSY' doit être scruté si l'on désire réaliser ce type de synchronisation.

BUSY

Le bit 'BUSY' est mis à 1 une fois que tous les autres registres ont été initialisés afin de démarrer le transfert. Il restera à 1 tant que le transfert ne sera pas terminé. La ligne d'interruption est une copie conforme de l'état de ce bit. Voir l'annexe A pour des informations complémentaires sur la façon de traiter le bit 'BUSY'.





## ANNEXE A -- EXEMPLE DE PROGRAMMATION

Afin de maintenir une compatibilité logicielle avec les futurs ST d'Atari équipés d'un 'Blitter', les développeurs n'ont besoin que de rester en accord avec les documentations 'Ligne A' et 'VDI'. Les futurs systèmes d'exploitation en ROM utiliseront le 'Blitter' pour améliorer les performances de nombreuses fonctions du VDI et de la ligne A. Cela s'effectuera de façon transparente pour le programme et l'utilisateur. En conséquence, le développeur n'a besoin de mettre en oeuvre aucune routine particulière pour disposer des avantages du 'Blitter'.

Comme règle de conduite, n'effectuez jamais un appel au VDI ou à la ligne A sous interruption, faute de quoi les résultats seront imprévisibles et peut-être catastrophiques au cas où une opération sur le 'Blitter' viendrait interrompre une autre opération sur ce même 'Blitter'.

Le programme listé ci-dessous n'a pas été optimisé, il est seulement fourni à des fins pédagogiques.

```
* (C) 1987 Atari Corporation
*     Tous droits réservés
*
* ADRESSE DE BASE DU BLITTER
*
BLITTER         equ         $FF8A00
*
* DECALAGES DES REGISTRES DU BLITTER
*
Halftone        equ         0       * Matrice demi-teinte *
Src_Xinc        equ         32      * Incrément X source *
Src_Yinc        equ         34      * Incrément Y source *
Src_Addr        equ         36      * Adresse bloc source *
Endmask1        equ         40      * Masque final 1 *
Endmask2        equ         42      * Masque final 2 *
Endmask3        equ         44      * Masque final 3 *
Dst_Xinc        equ         46      * Incrément X destination *
Dst_Yinc        equ         48      * Incrément Y destination *
Dst_Addr        equ         50      * Adresse bloc destination *
X_count         equ         54      * Largeur de bloc *
Y_Count         equ         56      * Hauteur de bloc *
HOP             equ         58      * Type opération demi-teinte *
OP              equ         59      * Type opération logique *
Line_Num        equ         60      * index de ligne dans matrice *
Skew            equ         61      * oblique *
*
```

```
* DRAPEAUX DE REGISTRES DU BLITTER
*
fHOP_Source     equ     1
fHOP_Halftone   equ     0
*
fSkewFXSR       equ     7
fSkewNFSR       equ     6
*
fLineBusy       equ     7
fLineHog        equ     6
fLineSmudge     equ     5
*
* MASQUES DE REGISTRES DU BLITTER
*
mHOP_Source     equ     $02
mHOP_Halftone   equ     $01
*
mSkewFXSR       equ     $80
mSkewNFSR       equ     $40
*
mLineBusy       equ     $80
mLineHog        equ     $40
mLineSmudge     equ     $20
*
*                   DONNEES DES MASQUES FINAUX
*
* Ces tables sont référencées par des instructions d'adressage
* relatives au compteur de programme. Aussi les noms de ces
* tables doivent-ils se trouver dans les 128 octets des
* instructions de référence. Amen.
*
* 0: Destination   1: Source
* << Inverser données du masque de données droite >>
*
lf_endmask
     dc.w       $FFFF
*
rt_endmask
     dc.w       $7FFF
     dc.w       $3FFF
     dc.w       $1FFF
     dc.w       $0FFF
     dc.w       $07FF
     dc.w       $03FF
     dc.w       $01FF
     dc.w       $00FF
     dc.w       $007F
     dc.w       $003F
     dc.w       $001F
     dc.w       $000F
     dc.w       $0007
     dc.w       $0003
     dc.w       $0001
     dc.w       $0000
```





```
*
* Titre:        BLiT_iT
*
* But:          Transférer un bloc de pixels situés à une position
*               arbitraire  X,Y  dans la forme mémoire source vers
*               une  autre  position  arbitraire X,Y dans la forme
*               mémoire  destination  en utilisant le mode REPLACE
*               (opération logique 3).
*               Les  rectangles  source  et destination ne doivent
*               pas se recouvrir.
*
* Entrées:
*               a4:  pointeur  vers un bloc de paramétres d'entrée
*                    de 34 octets.
*
* Note:         Cette  routine  doit  être exécutée en mode super-
*               viseur  car  un accès vers des registres matériels
*               est effectué dans une région protégée de mémoire.
*
*
*    Décalages du Bloc de Paramètres d'entrée
*
SRC_FORM        equ      0 ; Adresse base du bloc source
SRC_NXWD        equ      4 ; Décalage entre mots dans plan source
SRC_NXLN        equ      6 ; Largeur du bloc source
SRC_NXPL        equ      8 ; Décalage entre plans de la source
SRC_XMIN        equ     10 ; X minimum rectangle source
SRC_YMIN        equ     12 ; Y minimum rectangle source
*
DST_FORM        equ     14 ; Adresse base du bloc destination
DST_NXWD        equ     18 ; Décalage entre mots plan destination
DST_NXLN        equ     20 ; Largeur du bloc destination
DST_NXPL        equ     22 ; Décalage entre plans destination
DST_XMIN        equ     24 ; X minimum rectangle destination
DST_YMIN        equ     26 ; Y minimum rectangle destination
*
WIDTH           equ     28 ; Largeur du rectangle à transférer
HEIGHT          equ     30 ; Hauteur du rectangle à transférer
PLANES          equ     32 ; Nombre de plans à transférer
*
BLiT_iT:
*
     lea        BLiTTER,a5      ; a5-> registre bloc BLITTER
*
* Calcule les coordonnées maximales horizontales à partir des
* coordonnées minimales horizontales et de la largeur
*
     move.w     WIDTH(a4),d6
     subq.w     #1,d6           ; d6 <- largeur - 1

     move.w     SRC_XMIN(a4),d0 ; d0 <- X minimum source
     move.w     d0,d1           ; d1 <- X minimal source
     add.w      d6,d1           ;  + largeur du bloc
```

```
    move.w    DST_XMIN(a4),d2  ; d2 <- X minimum destination
    move.w    d2,d3            ; d3 <- X minimal destination
    add.w     d6,d3            ;  + largeur du bloc
*
* Les masques de fin sont déduits de l'origine horizontale de la
* source modulo 16 et de l'origine horizontale de la destination
* modulo 16.
*
    moveq     #$0F,d6          ; d6 <- masque modulo 16

    move.w    d2,d4            ; d4 <- DST_XMIN
    and.w     d6,d4            ; d4 <- DST_XMIN modulo 16
    add.w     d4,d4            ; d4 <- décalage gauche
    move.w    lf_endmask(pc,d4.w),d4 ; d4 <- masque fin gauche

    move.w    d3,d5            ; d5 <- DST_XMAX
    and.w     d6,d5            ; d5 <- DST_XMAX modulo 16
    add.w     d5,d5            ; d5 <- décalage droite
    move.w    rt_endmask(pc,d5.w),d5 ; d5 <- masque fin droite
    not.w     d5               ; d5 <- masque fin droite
*
* La valeur d'oblique est égale à (Xmin destination modulo 16
* - Xmin source modulo 16) && 0x000F. Trois discriminants sont
* utilisés pour déterminer les états des drapeaux FXSR et NFSR:
*
*    bit 0     0: Xmin source mod 16 <= Xmin destination mod 16
*              1: Xmin source mod 16 > Xmin destination mod 16
*
*    bit 1     0: SrcXmax/16-SrcXmin/16 <> DstXmax/16-DstXmin/16
*                   largeur source          largeur destination
*              1: SrcXmax/16-SrcXmin/16 == DstXmax/16-DstXmin/16
*
*    bit 2     0: largeur destination multi-mots
*              1: largeur destination = un seul mot
*
*    Ces drapeaux fournissent le décalage dans la table d'oblique
*    supportant les drapeaux d'état FXSR et NFSR pour des aligne-
*    ments donnés de la source et de la destination.
*

    move.w    d2,d7            ; d7 <- Dst Xmin
    and.w     d6,d7            ; d7 <- Dst Xmin modulo 16
    and.w     d0,d6            ; d6 <- Src Xmin modulo 16
    sub.w     d6,d7            ; d7 <- Dst Xmin modulo 16
                               ;  - Src Xmin modulo 16
    clr.w     d6               ; d6 <- index base table drapeaux
    addx.w    d6,d6            ; d6[bit 0] <- drapeau d'aligne-
                               ; ment dans le mot
    lsr.w     #4,d0            ; d0 = décalage hor. vers srcXmin
    lsr.w     #4,d1            ; d1 = décalage vert.vers srcXmax
    sub.w     d0,d1            ; d1 <- largeur source - 1

    lsr.w     #4,d2            ; d2 <- décalage mot vers dstXmin
    lsr.w     #4,d3            ; d3 <- décalage mot vers dstXmax
```





```
    sub.w     d2,d3            ; d3 <- largeur destination - 1
    bne       set_endmasks     ; deuxième discriminant = un mot
                               ; de destination
*
* Lorsque la destination n'a qu'un mot de large, les masques de
* début et de fin sont fusionnés pour créer le Masque Final 1.
* Les autres masques finaux seront ignorés par le BLiTTER.
*
    and.w     d5,d4            ; d4 <- masque de fin mot simple
    addq.w    #4,d6            ; d6[bit 2]:1 => un mot dest.

set_endmasks:

    move.w    d4,Endmask1(a5)  ; masque final gauche
    move.w    #$FFFF,Endmask2(a5) ; masque final centre
    move.w    d5,Endmask3(a5)  ; masque final droite

    cmp.w     d1,d3            ; dernier discriminant correspond
    bne       set_count        ; égalité largeurs src et dst

    addq.w    #2,d6            ; d6[bit 1]:1 => largeurs égales

set_count:

    move.w    d3,d4
    addq.w    #1,d4            ; d4 <- nombre mots ligne dest.
    move.w    d4,X_count(a5)   ; place valeur dans BLiTTER

* Calcule l'adresse de départ de la Source:
*
*     Adresse du Bloc Source +
*     ( Ymin source * Largeur Bloc source ) +
*    (( Xmin source / 16) * Xinc Source)

    move.l    SRC_FORM(a4),a0  ; a0 <- début bloc source
    move.w    SRC_YMIN(a4),d4  ; d4 <- décalage en lignes SrcYmin
    move.w    SRC_NXLN(a4),d5  ; d5 <- longueur ligne source
    mulu      d5,d4            ; d4 <- décalage en octets jusque
                               ; (0, Ymin)
    add.l     d4,a0            ; a0 -> (0,Ymin)

    move.w    SRC_NXWD(a4),d4  ; d4<- décalage entre mots consé-
    move.w    d4,Src_Xinc(a5)  ;    cutifs dans un plan source

    mulu      d4,d0            ; d0<- décalage mot contenant Xmin
    add.l     d0,a0            ; a0-> 1er mot source(Xmin, Ymin)

* Src_Yinc est le décalage en octets entre de dernier mot d'une
* ligne de la source et le premier mot de la ligne suivante.

    mulu      d4,d1            ; d1<- taille ligne srce en octets
    sub.w     d1,d5            ; d5 <- valeur ajoutée au pointeur
    move.w    d5,Src_Yinc(a5)  ;    de fin de ligne pour pointer
                               ;    le début de ligne suivante
```

```
*
* Calcule l'adresse de départ de la destination:
*
*     Adresse du Bloc destination +
*     ( Ymin destination * Largeur Bloc destination ) +
*    (( Xmin destination / 16) * Xinc destination)

     move.l    DST_FORM(a4),a1  ; a1 <- début bloc destination
     move.w    DST_YMIN(a4),d4  ; d4 <- décalage en lignes DstYmin
     move.w    DST_NXLN(a4),d5  ; d5 <- longueur ligne destination
     mulu      d5,d4            ; d4 <- décalage en octets jusque
                                ; (0, Ymin)
     add.l     d4,a1            ; a1 -> (0,Ymin)

     move.w    DST_NXWD(a4),d4  ; d4<- décalage entre mots consé-
     move.w    d4,Dst_Xinc(a5)  ;    cutifs dans un plan dest.

     mulu      d4,d2            ; d2<- décalage mot contenant Xmin
     add.l     d2,a1            ; a1-> 1er mot dest.(Xmin, Ymin)

* Dst_Yinc est le décalage en octets entre de dernier mot d'une
* ligne de la destination et le premier mot de la ligne suivante.

     mulu      d4,d3            ; d3<- taille ligne DSTe en octets
     sub.w     d3,d5            ; d5 <- valeur ajoutée au pointeur
     move.w    d5,Dst_Yinc(a5)  ;    de fin de ligne pour pointer
                                ;    le début de ligne suivante
*
* Le quartet bas de la différence entre l'alignement de la source
* et de la destination constitue la valeur oblique. Utilisation
* du drapeau d'index d'oblique pour référencer les états de FXSR
* et NFSR dans la table des drapeaux d'oblique
*
     and.b     #$0F,d7          ; d7 <- compte oblique de base
     or.b      skew_flags(pc,d6.w),d7 ; d7 <- drapeaux nécessaires
     move.b    d7,Skew(a5)      ; charge registre Oblique

     move.b    #mHOP_Source,HOP(a5) ; HOP fixé: source seule
     move.b    #3,OP(a5)        ; OP logique = mode REPLACE

     lea       Line_Num(a5),a2  ; registre numéro de ligne
     move.b    #fLineBusy,d2    ; drapeau ligne Busy
     move.w    PLANES(a4),d7    ; d7 <- controleur de plan
     bsr       begin
```





```
*              Le placement des drapeaux Oblique
*
*
* Qualificateurs    Actions  Direction transfert: Gauche->Droite
*
* égal Sx&F>
* larg.Dx&F FXSR NFSR
*
*  0    0    0    1 |..sssssssssssss|sssssssssssss..|
*                    |......ddddddddd|ddddddddddddddd|dd......
*
*  0    1    1    0 |......ssssssssss|ssssssssssssssss|ss......
*                    |..ddddddddddddd|ddddddddddddd..|
*
*  1    0    0    0 |..sssssssssssss|sssssssssssss..|
*                    |...ddddddddddddd|ddddddddddddd.|
*
*  1    1    1    1 |...ssssssssssss|ssssssssssssss.|
*                    |..ddddddddddddd|ddddddddddddd..|
*


skew_flags:

     dc.b      mSkewNFSR        ; Larg. Source < larg. Destination
     dc.b      mSkewFXSR        ; Larg. Source > larg. Destination
     dc.b      0                ; Largeur=décalage droite source
     dc.b      mSkewNFSR+mSkewFXSR ; Largeur=décalage gche source

* Lorsque la largeur de la destination est un simple mot

     dc.b      0                ; largeur source = 0 mot
     dc.b      mSkewFXSR        ; largeur source de deux mots
     dc.b      0                ; pas de drapeau d'oblique si la
     dc.b      0                ; largeur de source et de dest.
                                ; sont égales à un mot.
```

```
next_plane:

     move.l    a0,Src_Addr(a5)  ; pointeur Source pour ce plan
     move.l    a1,Dst_Addr(a5)  ; pointeur Dest. pour ce plan
     move.w    HEIGHT(a4),Y_count(a5) ; compteur de lignes

     move.b    #mLineBusy,(a2)  ; <<< démarrage du BLiTTER >>>

     add.w     SRC_NXPL(a4),a0  ; a0-> début prochain plan srce
     add.w     DST_NXPL(a4),a1  ; a1-> début prochain plan dest.


*
* Le BLITTER  opère  généralement avec le drapeau HOG à 0. Dans ce
* mode,  le BLITTER et l'unité centrale du ST se partagent équita-
* blement  le  bus,  chacun travaillant durant 64 cycles d'horloge
* tandis  que l'autre est stoppé. Ce mode permet aux interruptions
* d'être prises en compte par le 68000 lorsqu'un transfert de bloc
* large  est effectué.  La contrepartie de ce mode opératoire  est
* que les transferts de blocs effectués dans ce mode prennent deux
* fois plus de temps que dans l'autre mode (HOG à 1).
* 90% des performances du mode HOG peuvent cependant être obtenues
* si l'on  adopte  une  procédure  de  redémarrage ultra-rapide du
* blitter.  Lorsque le contrôle est rendu au 68000 par le blitter,
* celui-ci  réinitialise immédiatement le drapeau BUSY, permettant
* au  Blitter  de  redémarrer  après seulement 7 cycles horloge au
* lieu  des 64 cycles normaux. Les interruptions en attente seront
* traitées  avant  que  le code de redémarrage prenne effet. Si le
* drapeau BUSY est réinitialisé lorsque le compteur de ligne est à
* zéro, le drapeau restera à 0, indiquant la fin de l'opération de
* transfert et la non nécessité de redémarrer le Blitter.
*
* (Les  routines de traitement d'interruption doivent obligatoire-
* ment stopper le BLITTER lors de l'exécution de parties critiques
* par  mise  à 0 du drapeau BUSY. L'état antérieur du drapeau BUSY
* devra ensuite être restauré, avant la fin de la routine de trai-
* tement d'interruption.)


restart:

     bset.b    d2,(a2)          ; redémarre le Blitter + test BUSY
     nop                        ; pour laisser du temps aux inter.
     bne       restart          ; redémarrage du Blitter si le
*                               ; drapeau n'était pas à 0.

begin:

     dbra      d7,next_plane    ; plan suivant
     rts

-------------------------------------------------------------------
```





**Annexe B -- Fonction XBios de configuration du Blitter**

0x40  Blitmode - Fixe/Demande la configuration du Blitter

```
     Synopsis:          int  Blitmode(drapeau)
                        int  drapeau;
```

La fonction 0x40 (64 en décimal) du Bios étendu (trap #14) fixe et demande la configuration actuelle du blitter. Si 'drapeau' est égal à -1 (0xFFFF), aucune opération n'est effectuée et la configuration courante du blitter est retournée. Si 'drapeau' est différent de -1, alors la configuration du blitter est fournie comme suit:

| | |
|---|---|
| bit 0: | 0: mode de transfert logiciel<br>1: mode de transfert matériel (blitter) |
| bits 1 .. 14: | indéfinis, réservés |
| bit 15: | doit être nul |

La configuration antérieure du blitter est retournée dans le mot faible de D0 avec les caractéristiques suivantes:

| | |
|---|---|
| bit 0: | 0: transferts par logiciel<br>1: transferts par blitter |
| bit 1: | 0: pas de Blitter implanté sur le système<br>1: Blitter implanté sur le système |
| bits 2 .. 14: | indéfinis, réservés, peuvent être à zéro ou à un au retour |
| bit 15: | toujours retourné à 0 |

Si l'on tente de positionner le mode de transfert "matériel" alors qu'aucun blitter n'est implanté sur le système, le mode "logiciel" reste positionné.

Les champs réservés sont destinés à de futures possibilités du blitter ou d'autres circuits graphiques. Ils ne doivent pas être pris en compte mais doivent être laissés inchangés car ils sont susceptibles de servir dans l'avenir.

Cet appel fonctionne avec toutes les versions du système d'exploitation sur ROMs (N.D.T.: Avec des ROMs antérieures à Avril 1987, cet appel renvoie 0x40, soit "transfert par logiciel" (bit 0 à 0) et "pas de Blitter implanté" (bit 1)).

EXEMPLE D'APPEL EN LANGAGE C

```
#define Blitmode(a)  xbios(64,a)

int       curmode;

Curmode = Blitmode(-1);    /* demande état courant blitter */
Blitmode(curmode | 1);     /* active le blitter */
travail();                 /* ... fait quelque chose */
Blitmode(curmode);         /* replace état antérieur blitter*/
```

EXEMPLE D'APPEL EN ASSEMBLEUR 68000

```
     move.w    #-1,-(sp)          * demande état courant
     move.w    #$40,-(sp)         * fonction Blitmode (demande)
     trap      #14                * bios étendu
     addq.l    #4,sp              * restaure la pile
     move.w    d0,-(sp)           * sauve état antérieur blitter
     or.w      #1,d0              * active le blitter
     move.w    d0,-(sp)           * fixe activation blitter
     move.w    #$40,-(sp)         * fonction Blitmode (fixe)
     trap      #14                * bios étendu
     addq.l    #4,sp              * restaure la pile
*
*    ... fait quelque chose
*
     move.w    #$40,-(sp)         * restaure l'état antérieur
     trap      #14                * bios étendu
     addq.l    #4,sp              * dépile paramètres
```





## Annexe C -- Références


[1] Rob Pike, Leo Guibas, et Dan Ingalls, 'SIGGRAPH84 Course Notes: Bitmap Graphics', AT&T Bell Laboratories 1984.

[2] William Newman et Robert Sproull, 'Principles of Interactive Computer Graphics', McGraw-Hill 1979, Chapitre 18.

[3] John Atwood, '16160 RasterOp Chip Data Sheet', Silicon Compilers 1984. Voir aussi 'VL16160 RasterOp Graphics/Boolean Operation ALU', VLSI Technology 1986.

[4] Adele Goldberg et David Robson, 'Smalltalk-80: The Language and its Implementation', Addison-Wesley 1983, Chapitre 18.