

mémento

**CLEFS
POUR
ATARI ST
Système de base**

**Daniel Martin
et Guy Herzet**

Editions 

CLEFS POUR ATARI ST

Systeme de base



...
...
...
...
...



...
...

Connaissez-vous la collection Atari ST chez P.S.I. ?

- C sur Atari ST — Claude Nowakowski
- Clefs pour Atari ST, 2. Gem — Daniel Martin
- Clefs pour C — François Piette
- Trois étapes vers l'intelligence artificielle pour Atari ST — René Descamps et Augustin Garcia-Ampudia
- 102 programmes pour Atari ST — Jacques Deconchat
- Création et animation graphique sur Atari ST — Gilles Fouchard et Jean-Yves Corre

Pour tout problème rencontré dans les ouvrages P.S.I.
vous pouvez nous contacter au numéro ci-dessous :

Numéro Vert/Appel Gratuit en France

05 21 22 01

(Composer tous les chiffres, même en région parisienne)

La loi du 11 mars 1957 n'autorisant, aux termes des alinéas 2 et 3 de l'article 41, d'une part, que les « copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective », et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, « toute représentation ou reproduction intégrale, ou partielle, faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause, est illicite » (alinéa 1^{er} de l'article 40).

Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contrefaçon sanctionnée par les articles 425 et suivants du Code Pénal.

© Editions P.S.I., une société du Groupe

5 place du Colonel Fabien, 75491 Paris Cedex 10

1987



ISBN : 2-86595-351-3

CLEFS POUR ATARI ST

Systeme de base

**Daniel Martin
et
Guy Herzet**



**Éditions PSI
1987**

Présentation des auteurs :

Daniel Martin : agrégé en mathématique physique. Après quelques mois au Ministère de l'Education Nationale, attiré par la micro-informatique, il entre comme Computer Manager chez Tandy Corporation. Un bref passage chez Apple aux Pays-Bas puis cinq ans d'ingénierie système chez Intertechnique, il est actuellement chef de projet à la Sesa, société française de services. Il est l'auteur d'une série de "Livre du ..." éditée par BCM/PSI et des "Clefs pour Amstrad" et "Clefs pour PC et compatibles".

Guy Herzet : ingénieur industriel polyvalent. Ingénieur de maintenance chez Burroughs pendant un an, employé au service d'informatique et d'électronique d'un grand quotidien belge, il est, depuis 1984, ingénieur au laboratoire de recherche et de micro-informatique de l'Institut National des Industries EXtractives (INIEX). Il a collaboré avec Daniel Martin à l'élaboration du "Clefs pour PC et compatibles".

PRÉSENTATION

L'ATARI ST constitue une machine fantastique. Premier micro complètement architecturé autour d'un vrai processeur 16/32 bits, il est disponible en configuration complète, pour un prix situé sous la barre psychologique des 10 000 FF.

En outre, le système d'exploitation, livré en standard et basé sur la manipulation des fenêtres et des icônes, était auparavant réservé à des machines valant plus de trois fois le prix du ST et ne possédant pas les possibilités des modes graphiques multicolores de l'ATARI.

Ce manuel constitue le premier tome des "Clefs pour Atari ST". Il traite de tous les modèles actuels de la gamme ST du 260 au 1040.

Le présent volume analyse les caractéristiques générales du système, le langage Basic et le langage Logo, le système d'exploitation GEMDOS et son BIOS, la programmation du processeur 68000 et des différents circuits. Il comporte aussi les brochages des différents circuits et des connecteurs.

Le second volume portera sur le GEM AES, le GEM VDI, le langage C et les différents utilitaires du système.

Au niveau des logiciels de base, le système ST est toujours à la recherche de son équilibre. L'examen du Basic ne nous laisse aucun doute quant à l'apparition d'importantes modifications dans des versions ultérieures. Dès lors, soyez assurés que nous mettrons tout en oeuvre en vue de maintenir ce manuel à jour suivant les différentes versions qui pourraient apparaître sur le marché dans les prochains mois.

Atari ST est une marque déposée d'Atari Corp.
Nous remercions Atari pour l'aide apportée à la réalisation de cet ouvrage.

GEM est une marque déposée de Digital Research.

SOMMAIRE

PRESENTATION	5
CHAPITRE I - STRUCTURE INTERNE	11
Généralités	11
Schéma général du ST	12
Configuration générale	13
Différentes interfaces	14
Système graphique	18
Configuration mémoire	19
Table des adresses de communication avec les périphériques	20
Table des interruptions	25
Support des cartouches ROM	27
Gestionnaire intelligent du clavier	28
Table des commandes du gestionnaire clavier intelligent	31
CHAPITRE II - LANGAGE BASIC	37
Généralités	37
Caractéristiques générales du Basic	38
Constantes, variables et opérateurs	39
Commandes et instructions Basic	42
Fonctions Basic	63
Messages d'erreur	71
Editeur	76
Basic et langage machine	78
Structure interne des variables	82
Format interne des valeurs et VARPTR	83
Basic et GEM	87
CHAPITRE III - LANGAGE LOGO	91
Généralités	91
Commandes et mots clés (primitives)	92
CLEFS POUR ATARI ST	7

Commandes Logo utilisant les caractères de contrôle	117
Messages d'erreur Logo	119
Table des adresses des mots clés Logo classés par mots clés	121
Table des adresses des mots clés Logo classés par adresses	124

CHAPITRE IV - LANGAGE MACHINE 127

Généralités	127
Architecture interne du MC 68000	129
Jeu d'instructions	135
Table des codes opératoires triés par code	151
Table des codes opératoires triés par mnémonique	160

CHAPITRE V - GEMDOS 167

Généralités	167
Spécifications des fichiers du GEMDOS	169
Principaux types de fichiers	170
Commandes internes	171
Programmes BATCH	177
Messages d'erreur de l'interpréteur de commandes	178
Architecture interne	179
Initialisation du système	180
CCP et programmes externes	184
Format d'un fichier exécutable	186
HANDLER périphériques	188
Structure du disque	189
Table d'allocation de fichier F.A.T.	190
Structure des répertoires	191
Attributs des fichiers	194
File Control Bloc (FCB)	195
Table des fonctions du BDOS	198
Fonctions du BIOS	220
Codes d'erreurs internes des fonctions du BIOS	221
Table des fonctions du BIOS standard	222
Table des fonctions du BIOS étendu	226
Variables systèmes internes	238

CHAPITRE VI - CIRCUITS 243

MFP 68901	243
Générateur sonore AY3-8910	251
Contrôleur de disque WD 1772	259
ACIA 6850	265
Circuit DMA	268
Circuit GLUE	269
Circuit MMU	269
Circuit SHIFTER	269

CHAPITRE VII - BROCHAGE ET CONNECTEURS **271**

Brochage du 68000	271
Brochage du MFP 68901	273
Brochage du circuit AY3-8910	275
Brochage du FDC WDI772	277
Brochage de l'ACIA 6850	279
Brochage du circuit DMA	280
Brochage du circuit GLUE	282
Brochage du circuit MMU	284
Brochage du circuit SHIFTER	286
Connecteurs - Alimentation	288
Connecteurs - sortie MIDI	288
Connecteurs - entrée MIDI	289
Connecteurs - Moniteur	290
Connecteurs - Imprimante	291
Connecteurs - Sortie RS-232 (MODEM)	292
Connecteur pour lecteur de disquettes	293
Connecteur pour disque dur	294
Connecteur pour cartouches	295
Connecteur pour souris ou manette de jeux	296
Connecteur pour manette de jeux	296

CHAPITRE VIII - TRUCS ET ASTUCES **297**

Instruction DEF FN et fonction FN	297
Empêcher le traçage des fenêtres par le Basic	300
Simulation des fonctions HEX\$, BIN\$ et &B	301
Manipulations de l'écran en Basic	303
Trucs en Logo :	
Traduction des primitives en français	305
Production de bruits divers	305
Appel du GEM en Basic	306
Appels du BIOS et du BDOS :	
Lecture d'un caractère	307
Hardcopy	308
Appel du PSG	308

**ANNEXE - TABLE DES CODES CLAVIER RETOURNÉS
PAR LE GESTIONNAIRE CLAVIER** **311**

INDEX **313**

CONSEILS DE LECTURE **321**

GENERALITES

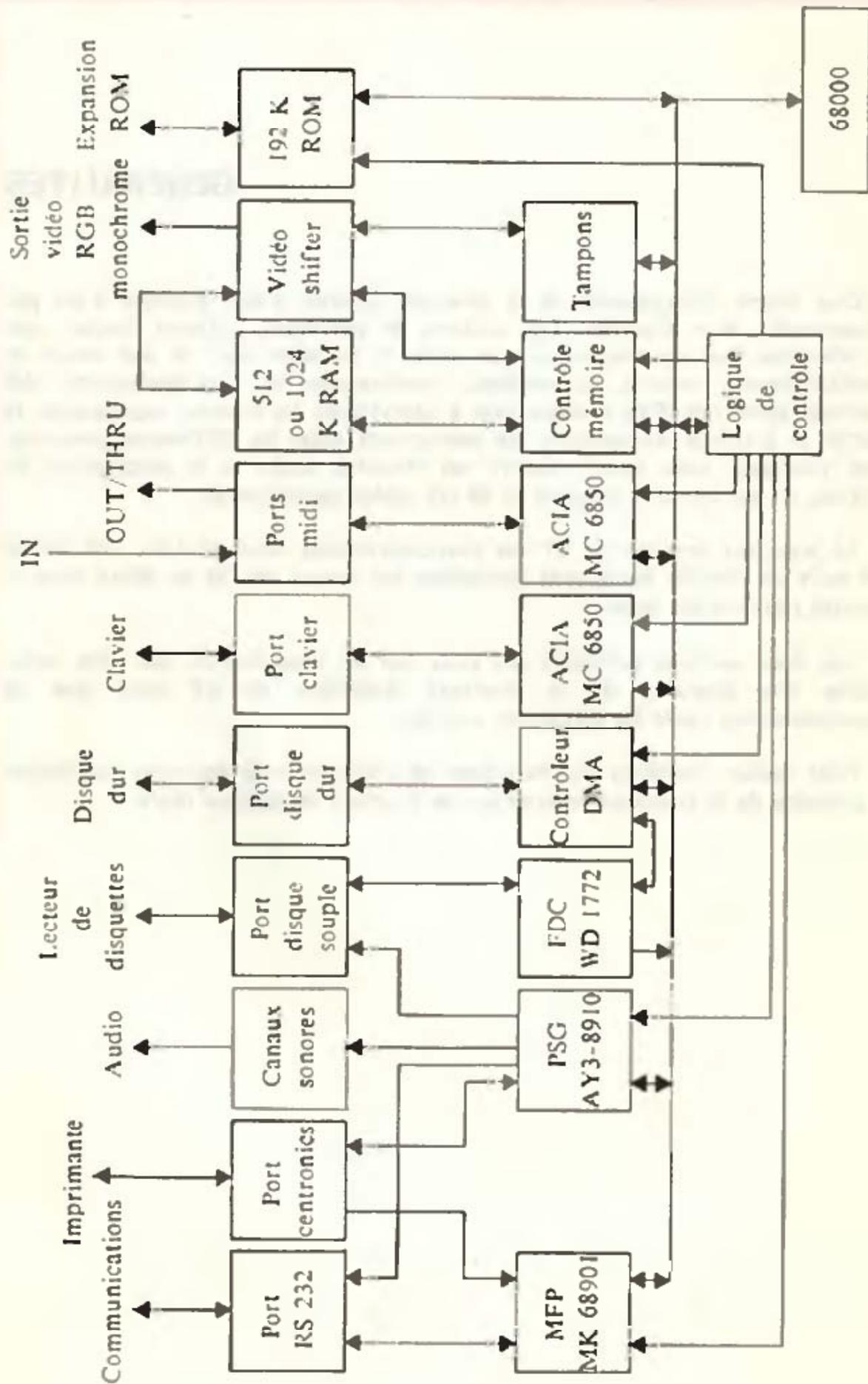
Une bonne connaissance de la structure interne d'une machine n'est pas indispensable pour l'utiliser. Des millions de personnes utilisent chaque jour un téléviseur ou une lessiveuse sans avoir la moindre idée de son mode de fonctionnement interne. Cependant, comprendre le fonctionnement des fonctions primaires d'un système aide à démystifier les couches supérieures du logiciel et à mieux comprendre les interactions entre les différents processus. C'est pourquoi nous avons réservé un chapitre entier à la description du système, de ses adresses internes et de ses tables particulières.

La structure interne du ST est particulièrement bien étudiée, elle utilise une série de circuits hautement spécialisés qui seront décrits en détail dans le chapitre qui leur est réservé.

Les deux sections suivantes ont pour but de vous donner une idée, aussi précise que possible, de la structure matérielle du ST ainsi que de l'interconnexion entre les différents circuits.

Pour mieux visualiser l'architecture du système, reportez-vous au schéma de principe de la structure interne qui se trouve à la page suivante.

SCHEMA GENERAL DU ST



CONFIGURATION GENERALE

Le coeur du système ST est bien entendu le processeur 16/32 bits 68000 auquel un chapitre entier est consacré. Le 68000 a une structure interne de 32 bits et un bus de communication externe de 16 bits. Le bus d'adresses est de 24 bits, ce qui permet d'adresser, sans interruption ou segmentation, 16 mégaoctets de mémoire centrale (2 exposant 24). Le 68000, piloté dans le ST par une horloge à 8 mégahertz, permet de très bonnes performances au niveau de la vitesse d'exécution des instructions.

Le 68000 est en contact avec une mémoire centrale modifiable (RAM) de 512 ou 1024 K, suivant les modèles. Cette mémoire est constituée de 16 ou de 32 circuits de 256 K fois un bit.

Le 68000 est aussi en contact avec une mémoire à lecture seule (ROM) de 192 K sur les systèmes sortis d'usine ces derniers mois, et de 32 K sur les anciens systèmes. Cette mémoire morte contient le minimum vital pour démarrer le système dans les premiers modèles et une grande partie du système d'exploitation dans les modèles plus récents.

Enfin, le 68000 est en contact avec une série de circuits spécialisés ou de coprocesseurs, dont voici une description sommaire.

Le circuit le plus complexe après le CPU (68000) est sans doute le MFP (Multi Fonction Périphérique) 68901, il gère toutes les interruptions du système ainsi que les transmissions RS232 et différents compteurs (TIMER).

Le contrôle du disque souple est assuré par un circuit WD 1772 de Western Digital.

L'AY3-8910 de General Instrument se charge de la production sonore sur trois voies et huit octaves. Il est en outre équipé de deux ports d'entrées/sorties qui permettent de contrôler l'imprimante parallèle, ainsi que différents signaux utiles au disque souple et à l'interface série RS232.

Le système comporte aussi deux ACIA 6850 de Motorola qui sont utilisés, l'un pour la gestion de l'interface MIDI, et l'autre pour la gestion du clavier intelligent.

En plus de ces circuits classiques et couramment commercialisés, ATARI a développé spécialement pour le ST, quatre circuits particuliers.

Le premier est un contrôleur de DMA (Accès Direct Mémoire) qui est fourni pour faciliter l'interfaçage d'un disque dur. Le DMA permet une vitesse de transfert entre le système et le disque dur de l'ordre de 8 mégaoctets par seconde. Le DMA est aussi utilisé par le FDC WD 1772 pour adresser le disque souple.

Le deuxième circuit est le MMU (Memory Management Unit) qui permet la gestion aisée de la mémoire centrale.

Le troisième circuit appelé SHIFTER contrôle la production du signal vidéo et l'interfaçage avec la partie de la mémoire centrale qui sert de VIDEORAM.

Le dernier circuit est purement anecdotique, il porte le doux nom de GLUE CHIPS (circuit de colle), et remplace simplement une série de circuits TTL classiques qui occuperait trop de place sur le circuit imprimé de la machine.

DIFFERENTES INTERFACES

Le système ST supporte sept types d'interfaces que nous allons décrire sommairement.

Interface vidéo

L'interface vidéo permet d'utiliser un moniteur monochrome haute résolution et un moniteur couleur RGB ou un téléviseur équipé d'une prise PERITEL. La reconnaissance par le système de la présence d'un moniteur monochrome est due à la présence d'un pontage sur une broche prévue à cet effet, à l'intérieur de la prise de connexion du moniteur. Ce pontage est transmis sur le PORT du MFP et le logiciel du ST peut ainsi en prendre connaissance.

Le mode de synchronisation (interne ou externe) et la fréquence de référence (50 ou 60 hertz) sont sélectionnés par l'intermédiaire du registre de synchronisation du SHIFTER.

Deux autovecteurs d'interruption sont générés pour permettre au logiciel de se synchroniser avec les BLANKING (disparition temporaire du signal de balayage) verticaux et horizontaux. Cette particularité permet de produire des défilements horizontaux et verticaux de l'image sans parasites.

Interface parallèle type CENTRONICS

L'interface type CENTRONICS permet la connexion de la plupart des imprimantes parallèles du marché. L'interface est gérée en grande partie par le PSG. Le port B du PSG sert à la transmission des données, un bit du port A permet d'activer le STROBE. Seul le BUSY (imprimante occupée) est pris en charge par le MFP.

L'interface permet d'envoyer environ 4 000 octets par seconde, ce qui est largement supérieur aux possibilités des imprimantes actuelles.

Interface série RS232

L'interface série est gérée presque entièrement par le MFP. Elle permet n'importe quel format de donnée (5, 6, 7 ou 8 bits, parité paire, impaire ou nulle, un ou deux bits d'arrêt) et ce, à une vitesse comprise entre 50 et 19 200 bauds. La génération de l'horloge nécessaire à la stabilisation de la vitesse est prise en charge par le TIMER D du MFP. Seuls deux signaux de gestion de protocole (RTS et DTR) sont gérés par le PORT A du PSG, les autres (CTS, DCD et RI) sont pris en charge par le MFP.

Interface disque souple

Le système ST est capable de gérer deux disques souples. Les disques souples sont gérés par le FDC WD 1772 associé au circuit de DMA. Ils peuvent être simple ou double face et doivent comporter 80 pistes. On peut utiliser indifféremment des disques de 3 pouces et demi ou de 5 pouces un quart.

Les commandes sont envoyées au FDC en écrivant dans le registre de contrôle du DMA pour sélectionner le registre de commande du FDC, et en écrivant, ensuite, la commande dans le registre de contrôle du FDC.

La séquence complète peut se résumer de la façon suivante :

- sélection du lecteur 0 ou 1 (PSG PORT A) ;
- sélection de la face 0 ou 1 (PSG PORT A) ;
- chargement de l'adresse BASE DMA ;
- effacement du compte rendu d'état (passage du mode écriture au mode lecture) par le registre de contrôle de mode DMA ;
- sélection du mode lecture ou écriture (registre contrôle mode DMA) ;
- sélection du registre compteur secteur du DMA (par le registre de contrôle de mode DMA) ;
- chargement du registre compteur secteur ;
- sélection du registre interne de commande du FDC ;
- exécution de la commande d'écriture ou de lecture ;
- le DMA reste actif jusqu'à l'issue de la commande ;
- test du registre d'état du DMA (compte rendu d'erreur).

Remarque : la détection de l'insertion ou de l'éjection de la disquette n'est pas prise en compte par le matériel.

Interface disque dur

Le disque dur est également géré par le contrôleur de DMA. Cependant, l'utilisation d'un disque dur nécessite la présence d'un contrôleur particulier d'ATARI (AIDI) qui est fourni avec l'unité de disque.

L'interface spécialisée supporte un jeu minimum d'instructions dérivé des normes SCSI. Les données sont transmises par un bloc de 6 octets dont voici la description :

- octet 0 : B0 à B4 contiennent le code opération et B5 à B7 contiennent le numéro d'unité ;
- octet 1 : B0 à B4 contiennent la partie haute de l'adresse du bloc et B5 à B7 contiennent le numéro de DEVICE ;
- octet 2 : partie médiane de l'adresse du bloc ;
- octet 3 : partie basse de l'adresse du bloc ;
- octet 4 : compteur de bloc ;
- octet 5 : octet de contrôle.

DIFFERENTES INTERFACES

Les commandes possibles sont :

- 00 : test de l'unité.
- 05 : vérification d'une piste.
- 06 : formatage d'une piste.
- 08 : lecture.
- 0A : écriture.
- 0B : positionnement.
- 0D : motif de correction.
- 15 : sélection du mode.
- 1A : mode SENSE.

La séquence d'une opération de lecture ou d'écriture est semblable à celle du disque souple.

Interface clavier

Le clavier possède son propre processeur (HD6301VI) qui prend en charge la gestion des touches, du témoin de verrouillage majuscules et de la souris.

Ce processeur est interfacé au système central par l'intermédiaire d'un ACIA 6850. La vitesse de transfert entre le système central et le coprocesseur clavier est d'environ 8 000 octets par seconde.

L'interface clavier peut interpréter une série de commandes qui sont décrites à la fin de ce chapitre.

Interface MIDI

L'interface MIDI permet d'interfacer le système ST avec un système musical performant et spécialisé (séquenceur, orgue, synthé. batterie électronique...). Le système MIDI s'est imposé comme système de transmission standard parmi les plus grandes marques d'appareils musicaux (YAMAHA, ROLAND, KORG...).

Le système MIDI est un système de transmission série. La vitesse de transfert est fixée à 31.25 Kbauds. L'interface MIDI est prise en charge par le second ACIA du système ST. Il supporte l'entrée et la sortie MIDI (IN - OUT) ainsi que la transparence (THRU).

Le BUS MIDI permet l'utilisation de 16 canaux dans trois modes distincts :

- le mode OMNI (mode par défaut) adresse toutes les unités en même temps
- le mode POLY adresse chaque unité séparément ;
- le mode MONO adresse chaque voie séparément.

DIFFERENTES INTERFACES

Les informations sont transmises via cinq types différents de formats de données.

Ces cinq types sont :

SYSTEM RESET
SYSTEM EXCLUSIVE
SYSTEM REAL TIME
SYSTEM COMMON
CHANNEL

L'interface MIDI peut être pilotée comme l'écran ou l'imprimante au travers des appels de fonctions BIOS (voir *DOS*).

SYSTEME GRAPHIQUE

Le système graphique est basé sur l'interaction profonde du VIDEO SHIFTER et de la partie de la mémoire centrale qui sert de VIDEORAM (mémoire vidéo).

Vidéoram

La vidéoram est configurée comme n plans logiques composés de mots de 16 bits pour un plan physique de près de 32 Ko (7D00H ou 32 000 octets). Ce plan peut se trouver n'importe où en mémoire centrale pour autant que la première adresse soit située à une adresse multiple de 256. L'adresse de départ est placée dans le registre adresse de base du SHIFTER (16 bits) qui est lu et placé dans le registre compteur adresse vidéo et incrémenté.

Configuration

La mémoire peut être configurée en :

- 1 plan logique : 640 x 400 monochrome
- 2 plans logiques : 640 x 200 en 4 couleurs
- 4 plans logiques : 320 x 200 en 16 couleurs

Le mode de division des plans logiques est géré par le registre SHIFT MODE.

En outre, 16 mots permettent de définir les palettes pour chaque couleur. Chacun des mots possède 3 bits significatifs par couleur primaire (rouge, vert et bleu), ce qui représente 9 bits et 512 possibilités (2 exposant 9).

Le nombre de mots de palettes utilisés dépend du mode. En mode 16 couleurs, les 16 mots sont utilisés ; en mode 4 couleurs, les 4 premiers mots sont utilisés ; en mode monochrome, seul le bit 0 du premier mot est utilisé pour indiquer l'état normal ou l'inversion vidéo.

CONFIGURATION MEMOIRE

Les deux premiers Ko de la mémoire du ST sont réservés pour la table des vecteurs d'exception et la pile superviseur du 68000.

La partie haute de la mémoire est utilisée pour communiquer avec les circuits périphériques.

Une ROM de quatre mots est mise en recouvrement avec les huit premiers octets de la mémoire RAM pour configurer le registre compteur programme et le pointeur de pile, lors d'une initialisation.

L'écriture, dans la zone réservée au superviseur ou dans la zone de communication avec les périphériques en dehors du mode SUPERVISEUR, produit une erreur sur le bus.

00 0000	ROM	Initialisation : Pile SUPERVISEUR
00 0004	ROM	Initialisation : Compteur programme
00 0008	RAM	Début de la RAM Table des vecteurs d'exception
02 0000	RAM	Fin RAM pour les systèmes 128 K
08 0000	RAM	Fin RAM pour les systèmes 512 K
10 0000	RAM	Fin RAM pour les systèmes 1 méga
20 0000	RAM	Fin RAM pour les systèmes 2 méga
FA 0000	ROM	ROM sur connecteur d'extension
FC 0000	ROM	ROM BOOT (192K)
FE FFFF	ROM	Fin de la ROM
FF 8000	I/O	Registres de configuration
FF 8200	I/O	Registres du SHIFTER vidéo
FF 8400	I/O	Réservé
FF 8600	I/O	Registres DMA et FDC
FF 8800	I/O	Registres du PSG
FF FA00	I/O	Registres duMFP
FF FC00	I/O	Registres des ACIAs

TABLE DES ADRESSES DE COMMUNICATION AVEC LES PERIPHERIQUES

Les adresses de communication entre le processeur et les circuits périphériques se trouvent de FF0000 à FFFFFFFF.

FF8200 à FF82FF sont utilisés par le SHIFTER vidéo

FF8600 à FF86FF sont utilisés par le contrôleur DMA

FF8800 à FF88FF sont utilisés par le PSG

FFFA00 à FFFAFF sont utilisés par le MFP

FFFC00 à FFFCFF sont utilisés par les ACIAs

Remarques :

- L signifie Lecture et E Ecriture. L/E indique que l'adresse est utilisable à la fois en lecture et en écriture.
- LONG indique la longueur en octet.
- x indique que l'état du bit n'a pas d'importance.

Adresse	Long	L/E	Fonction
FF8001	1	L/E	Configuration de la mémoire exprimée sur les quatre bits de droite. Les deux bits les moins significatifs sont utilisés pour le BANK mémoire 1, les deux autres sont utilisés pour indiquer la configuration du BANK mémoire xxxxMMMM MM 00 128 K 01 512 K 10 2 M 11 Non utilisé
FF8201	1	L/E	Adresse haute de la base de la mémoire vidéo. (voir SHIFTER).
FF8203	1	L/E	Adresse basse de la base de la mémoire vidéo.
FF8205	1	L	Valeur haute du compteur d'adresse vidéo (Adresse sur 24 bits).
FF8207	1	L	Valeur médiane du compteur d'adresse vidéo.
FF8209	1	L	Valeur basse du compteur d'adresse vidéo.
FF820A	1	L/E	Mode de synchronisation (deux bits de droite) xxxxxxFS └─┬─┘ Synchro interne ou externe └─┬─┘ Fréquence 50 ou 60 hertz

TABLE DES ADRESSES DE COMMUNICATION AVEC LES PERIPHERIQUES

Adresse	Long	L/E	Fonction
FF8240	2	L/E	<p>Palette de la couleur 0 (bord). Chaque couleur de base est codée sur 3 bits xxxxRRRxxVVVxBBB</p> <div style="margin-left: 40px;"> <p>Intensité du bleu Intensité du vert Intensité du rouge</p> </div> <p>Lors d'une utilisation monochrome, seul le bit le moins significatif est utilisé pour indiquer l'état normal ou inversé de l'écran.</p>
FF8242	2	L/E	Palette de la couleur 1.
FF8244	2	L/E	Palette de la couleur 2.
FF8246	2	L/E	Palette de la couleur 3.
FF8248	2	L/E	Palette de la couleur 4.
FF824A	2	L/E	Palette de la couleur 5.
FF824C	2	L/E	Palette de la couleur 6.
FF824E	2	L/E	Palette de la couleur 7.
FF8250	2	L/E	Palette de la couleur 8.
FF8252	2	L/E	Palette de la couleur 9.
FF8254	2	L/E	Palette de la couleur 10.
FF8256	2	L/E	Palette de la couleur 11.
FF8258	2	L/E	Palette de la couleur 12.
FF825A	2	L/E	Palette de la couleur 13.
FF825C	2	L/E	Palette de la couleur 14.
FF825E	2	L/E	Palette de la couleur 15.
FF8260	2	L/E	<p>Mode graphique. xxxxxxxxxxxxxxxxMM</p> <ul style="list-style-type: none"> 00 Mode 320 x 200 16 couleurs 01 Mode 640 x 200 4 couleurs 10 Mode 640 x 400 monochrome 11 Non utilisé

TABLE DES ADRESSES DE COMMUNICATION AVEC LES PERIPHERIQUES

Adresse	Long	L/E	Fonction
FF8604	2	L/E	Mot d'accès du contrôleur de disque WDI772. XXXXXXXXMMMMMMMM
FF8606	2	L	Etat du DMA. XXXXXXXXXXXXXXXXIZE <ul style="list-style-type: none"> └─ Erreur └─ Compteur de secteur à 0 └─ Demande de donnée inactive
FF8606	2	E	Mode contrôle. XXXXXXXXCCCCCCCCCx <ul style="list-style-type: none"> └─ A0-CA1 └─ AI-90 └─ Sélection registre HDC/FDC └─ Sélection registre compteur de secteurs └─ Non utilisé └─ Autorisation DMA └─ FDC / HDC └─ Ecriture/lecture
FF8609	1	L/E	Partie haute du compteur et Base DMA (adresse 24 bits).
FF860B	1	L/E	Partie médiane du compteur et base DMA.
FF860D	1	L/E	Partie basse du compteur et base DMA.
FF8800	1	L	Lecture du port B du PSG. DDDDDDDD
FF8800	1	E	Sélection du registre du PSG (0 à 15). xxxxRRRR
FF8802	1	E	Ecriture sur les PORTS A et B du PSG. DDDDDDDD PORT A <ul style="list-style-type: none"> └─ Sélection face disque └─ Sélection disque A └─ Sélection disque B └─ RTS RS232 └─ DTR RS232 └─ STROBE imprimante // └─ Sortie a usage général └─ Réserve

TABLE DES ADRESSES DE COMMUNICATION AVEC LES PERIPHERIQUES

Adresse	Long	L/E	Fonction
FFFA00	1	L/E	MFP 68901 I/O à usage général (GPIP). (voir description du circuit) xxxxxxxxDDDDDDDD
FFFA03	1	L/E	MFP registre AER.
FFFA05	1	L/E	MFP Sens des données (DDR).
FFFA07	1	L/E	MFP Autorisation d'interruption A (IERA).
FFFA09	1	L/E	MFP Autorisation d'interruption B (IERB).
FFFA0B	1	L/E	MFP Pending interruption A (IPRA).
FFFA0D	1	L/E	MFP Pending interruption B (IPRB).
FFFA0F	1	L/E	MFP Interruption en service A (ISRA)
FFFA11	1	L/E	MFP Interruption en service B (ISRB).
FFFA13	1	L/E	MFP Masque d'interruption A (IMRA).
FFFA15	1	L/E	MFP Masque d'interruption B (IMRB).
FFFA17	1	L/E	MFP Vecteur (VR).
FFFA19	1	L/E	MFP Contrôle du TIMER A (TACR).
FFFA1B	1	L/E	MFP Contrôle du TIMER B (TBCR).
FFFA1D	1	L/E	MFP Contrôle des TIMERS C et D (TCDCR).
FFFA1F	1	L/E	MFP Données du TIMER A (TADR).
FFFA21	1	L/E	MFP Données du TIMER B (TBDR).
FFFA23	1	L/E	MFP Données du TIMER C (TCDR).
FFFA25	1	L/E	MFP Données du TIMER D (TDDR).

TABLE DES ADRESSES DE COMMUNICATION AVEC LES PERIPHERIQUES

<i>Adresse</i>	<i>Long</i>	<i>L/E</i>	<i>Fonction</i>
FFFA27	1	L/E	MFP Caractère de synchronisation (SCR).
FFFA29	1	L/E	MFP Contrôle de l'USART (UCR).
FFFA2B	1	L/E	MFP Etat de la réception (RSR).
FFFA2D	1	L/E	MFP Etat de l'émission (TSR).
FFFA2F	1	L/E	MFP Données de l'USART (UDR).
FFFC00	1	L/E	Contrôle de l'ACIA 6850 du clavier.
FFFC02	1	L/E	Données de l'ACIA 6850 du clavier.
FFFC04	1	L/E	Contrôle de l'ACIA 6850 MIDI.
FFFC06	1	L/E	Données de l'ACIA 6850 MIDI.

TABLES DES INTERRUPTIONS

Autovecteurs d'interruptions du 68000

<i>Priorité</i>	<i>Définition</i>	
7 (haut) 6 5 4 3 2 1 (basse)	NMI (Interruption non masquable) MK68901 MFP BLANKING vertical (synchronisation) BLANKING horizontal (synchronisation)	

Interruption du MFP MK68901

<i>Priorité</i>	<i>Définition</i>	<i>Adresse</i>
15 (haute)	Détection du moniteur monochrome	013C
31	RS232 broche RI	0138
31	Horloge système et BUSY	0134
12	RS232 tampon de réception plein	0130
11	RS232 Erreur en réception	012C
10	RS232 tampon d'émission vide	0128
09	RS232 erreur en émission	0124
08	Compteur BLANKING horizontal	0120
07	Contrôleur de disque	011C
06	CLAVIER et MIDI	0118
05	TIMER C	0114
04	RS232 TIMER D	0110
03	GPU opération accomplie	010C
02	RS232 CTS	0108
01	RS232 DCD	0104
00 (basse)	BUSY centronic	0100

Il est bon de remarquer que la détection de présence du moniteur monochrome a la plus haute priorité. Ainsi, la déconnexion de l'écran produit une initialisation du système.

Vecteurs TRAP

Le ST utilise quatre des seize vecteurs TRAP du 68000. Les douze autres sont disponibles pour les applications.

TABLES DES INTERRUPTIONS

<i>TRAP</i>	<i>Utilisation</i>
0	Inutilisé
1	Interface avec GEMDOS
2	Extension du DOS (GEM, GSX)
3-12	Inutilisés
13	BIOS
14	Extension du BIOS ATARI
15	Inutilisé

SUPPORT DES CARTOUCHES ROM

Le système ST permet d'utiliser des cartouches ROM d'extension. Ces cartouches peuvent contenir 128 Ko de mémoire. Elles commencent en FA0000 et se terminent en FBFFFF. L'adresse FA0000 permet au système de vérifier si une cartouche est utilisée.

Il existe deux catégories de programmes sur cartouches :

- les applications qui sont reconnues par GEM et traitées après l'initialisation ;
- les cartouches de diagnostic qui sont utilisés par les services techniques et qui prennent le contrôle du système dès l'initialisation.

Les cartouches de diagnostic contiennent FA52255F à l'adresse FA0000. Le système saute à l'adresse FA0004 sans initialiser quoi que ce soit.

Les cartouches d'application doivent contenir ABCDEF42 à l'adresse FA0000. Un en-tête d'application doit suivre cette série d'octets à l'adresse FA0004.

Un en-tête d'application se compose au maximum de 32 octets définis comme suit :

- 4 octets qui contiennent l'adresse du prochain en-tête éventuel ;
- 4 octets qui pointent sur l'adresse d'initialisation éventuelle de l'application
- 4 octets qui pointent sur le point d'entrée de l'application ;
- 4 octets qui pointent sur la définition éventuelle d'icônes pour l'application
- 4 octets qui indiquent la taille du segment BBS pour l'application ;
- 12 octets maximum qui contiennent le nom de l'application (8 pour le nom, un pour le point et trois pour l'extension du nom).

GESTIONNAIRE INTELLIGENT DU CLAVIER

Comme nous l'avons déjà signalé lors de la description des interfaces, le clavier possède son propre processeur qui dialogue avec le système central à travers un ACIA. Le gestionnaire intelligent s'occupe de la gestion des touches, de la gestion de la souris et des manettes de jeu ainsi que de la maintenance de la date interne, à la seconde près.

Analysons en détail les diverses fonctions du gestionnaire clavier intelligent :

Clavier proprement dit

Le gestionnaire du clavier retourne toujours un code sur 8 bits correspondant à la touche enfoncée. Ce code est totalement indépendant de la gravure des touches et est donc identique, quel que soit le pays d'utilisation. Les codes générés se trouvent dans le *tableau figurant ci-après*. Le code existe toujours, même si, pour certaines configurations de clavier, le constructeur ne disposait pas de touches à un endroit précis.

Le compte rendu de relâchement de la touche est obtenu en positionnant le bit 7 de l'octet fourni à 1.

Les codes F6 à FF ont un usage particulier défini comme suit :

F6	Compte rendu d'état.
F7	Enregistrement de la position absolue de la souris.
F8-FB	Enregistrement de la position relative de la souris.
FC	Date interne.
FD	Etat des manettes de jeu.
FE	Gestion de la manette 0.
FF	Gestion de la manette 1.

Remarque : les deux touches SHIFT, la touche ENTER et la touche RETURN fournissent des codes différents.

Souris

Le gestionnaire de la souris permet d'interfacer une souris qui peut fournir jusqu'à 80 transitions d'état par centimètre de déplacement. Autrement dit, la souris peut envoyer 80 impulsions lorsqu'on la déplace d'un seul centimètre. Sa résolution est donc de l'ordre du dixième de millimètre. La vitesse de suivi peut atteindre 25 cm par seconde, soit à peu près 2 000 transitions par seconde.

Le gestionnaire intelligent peut fournir un état de la souris sous trois formes différentes : le positionnement relatif, le positionnement absolu et la conversion directe en codes de déplacement de curseur, semblables à ceux des flèches du clavier.

Positionnement relatif

En positionnement relatif, le gestionnaire retourne un compte rendu dès qu'une modification d'état de la souris se produit (bouton enfoncé ou déplacement). Dans ce mode, tous les bits sont envoyés au système maître sans considération d'une position neutre préalable éventuelle.

Le contenu des trois octets est défini comme suit :

- Octet 1 : 111110RL
 - └─Etat du bouton gauche
 - └─Etat du bouton droit
- Octet 2 : Déplacement sur l'axe horizontal (delta x) exprimé en entier complément à deux.
- Octet 3 : Déplacement sur l'axe vertical (delta y) exprimé en entier complément à deux.

Remarque : si le déplacement entre deux lectures est supérieur à 127 ou inférieur à -128, le déplacement est fourni sous la forme de plusieurs séries de trois octets.

Positionnement absolu

Le positionnement absolu fonctionne de façon similaire au positionnement relatif. Cependant, les valeurs de delta x et de delta y fournies sont calculées par rapport à une position absolue et un facteur d'échelle. Ces derniers peuvent être programmés préalablement par des commandes internes.

Conversion code clavier

Le déplacement de la souris peut être converti en un nombre de codes de déplacement de curseur. La correspondance entre le nombre de transitions de la souris et l'envoi d'un code de déplacement est programmable de façon indépendante pour chaque axe. Le déplacement de la souris produit donc une série de codes correspondant aux flèches, chaque code étant directement suivi de son code de relâchement (bits 7 à 1).

Manettes de jeux

Le gestionnaire clavier permet d'interroger les manettes de jeux selon cinq modes différents :

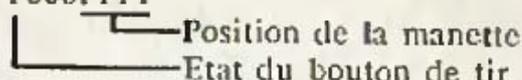
Rapport de déplacement

Dans ce premier mode, le gestionnaire fournit un compte rendu de deux octets chaque fois que la position de la manette est modifiée.

Octet 1 : 1111111J --- J est le numéro de la manette (0 ou 1).

GESTIONNAIRE INTELLIGENT DU CLAVIER

Octet 2 : T000PPPP



Interrogation de la manette

Dans ce mode, le gestionnaire fournit un compte rendu de trois octets qui fournit la position de la manette :

Octet 1 : 11111101 - FD en-tête du rapport.
Octet 2 : T000PPPP - Rapport de la manette 0 (défini ci-dessus).
Octet 3 : T000PPPP - Rapport de la manette 1 (défini ci-dessus).

Surveillance de la manette

Dans ce mode, chaque communication avec le clavier produit un compte rendu de l'état de la manette. La vitesse de lecture est programmable.

Surveillance du bouton de tir

Dans ce mode, seul l'état du bouton de tir est fourni au système et ce, à une vitesse définie préalablement.

Conversion en code clavier

Dans ce mode, la position de la manette est convertie en code de déplacement du curseur.

Horloge

Le gestionnaire du clavier maintient l'horloge interne du système. Il existe des commandes permettant de positionner et d'interroger l'horloge. La résolution de l'horloge est fixée à une seconde.

Compte rendu d'état interne du gestionnaire

L'état interne du gestionnaire et ses différents paramètres peuvent être fournis en envoyant une demande de rapport d'état.

Initialisation

A l'initialisation, le gestionnaire intelligent du clavier exécute un autotest pour détecter une erreur matérielle interne éventuelle. Si l'autotest ne détecte pas d'erreur, le gestionnaire retourne un code hexadécimal Fx au système central. x représente le numéro de version du gestionnaire (actuellement 0).

A l'initialisation, certaines valeurs par défaut sont positionnées :

- souris en haut et à gauche de l'écran ;
- manette 1 sélectionnée ;
- les boutons de tir sont assignés à la souris.

Initialisation

Code : 80 01 (hexa)

La commande d'initialisation positionne tous les paramètres à leurs valeurs par défaut et exécute l'autotest. Si le test est correct, le gestionnaire fournit F0 au système. L'initialisation n'affecte pas l'horloge.

Gestion de la souris

Positionnement du mode d'action du bouton de la souris

Code : 00 à 07

Cette commande positionne la manière dont le gestionnaire doit traiter une action sur le bouton de la souris. Par défaut à l'initialisation, c'est le mode 00 qui est choisi.

00	Mode normal.
01	La pression produit un compte rendu de position absolue de la souris.
02	Le relâchement produit un compte rendu de position absolue de la souris.
04	Le bouton agit comme une touche du clavier.
05 06 07	Sont identiques à 04.

Positionnement de la souris en mode relatif

Code : 08

Cette commande est sélectionnée par défaut à l'initialisation.

Positionnement de la souris en mode absolu

Code : 09 XMAXH XMAXB YMAXH YMAXB

Cette commande comporte cinq octets. XMAXH constitue la partie haute de la valeur maximale que peut prendre le nombre de transitions sur l'axe X. XMAXB constitue la partie basse de la même valeur. YMAXH et YMAXB sont identiques à XMAXH et XMAXB pour le nombre de transitions sur l'axe Y.

Positionnement de la souris en mode clavier

Code : 0A deltaX deltaY

Cette commande comporte trois octets. deltaX exprime le nombre de transitions nécessaires pour produire un code de déplacement horizontal (gauche ou droit). deltaY exprime la même chose pour un code de déplacement vertical.

TABLES DES COMMANDES DU GESTIONNAIRE CLAVIER INTELLIGENT

Positionnement du seuil de déclenchement

Code : 0B seuilX seuilY

Cette commande comporte trois octets. seuilX exprime le nombre de transitions nécessaires sur l'axe X avant qu'un évènement se produise. seuilY exprime la même chose pour l'axe Y. Le seuil ne modifie absolument pas la résolution des données. Cette commande est valide uniquement en mode relatif. Par défaut, le seuil est positionné à 1.

Positionnement de l'échelle en mode absolu

Code : 0C echX echY

Cette commande positionne le facteur d'échelle de la souris en mode absolu. echX et echY expriment respectivement le nombre de transitions nécessaire pour produire la modification du compte rendu de positionnement absolu sur l'axe X et sur l'axe Y.

Interrogation de la position de la souris

Code : 0D

Cette commande est valide en mode positionnement absolu quel que soit l'état du bouton d'action. Elle fournit un compte rendu sous la forme de 6 octets définis comme suit :

Octet 1 :F7	En-tête de rapport de positionnement
Octet 2 :0000xxxx	Etat des boutons
	┌─┐ Droit enfoncé depuis la dernière interrogation
	├─┐ Droit relâché depuis la dernière interrogation
	└─┐ Gauche enfoncé depuis la dernière interrogation
	└─┐ Gauche relâché depuis la dernière interrogation
Octet 3 :XH	Partie haute de la position horizontale
Octet 4 :XB	Partie basse de la position horizontale
Octet 5 :YH	Partie haute de la position verticale
Octet 6 :YB	Partie basse de la position verticale

Chargement de la position de la souris

Code : 0E 00 XH XB YH YB

Cette commande permet de programmer de façon interne la position absolue de la souris.

Positionnement du 0 vertical en bas

Code : 0F

Cette commande permet de positionner l'origine de l'axe vertical Y en bas du système de coordonnées. Cette option inverse le signe des déplacements verticaux.

Positionnement du 0 vertical en haut

Code : 10

Cette commande permet de positionner l'origine de l'axe vertical Y en haut du système de coordonnées. C'est la position choisie par défaut à l'initialisation.

Reprise

Code : 11

Cette commande permet de reprendre après une pause (voir 14). Si le système n'est pas en mode pause, la commande est simplement ignorée.

Suspension de l'action de la souris

Code : 12

Cette commande empêche la gestion de la souris.

Pause

Code : 14

Cette commande suspend l'émission de données vers le système central jusqu'à l'arrivée d'une autre commande valide. Les déplacements de la souris peuvent être accumulés pendant que le système est en mode pause. La pause commence son action à l'issue du déroulement de la commande en cours d'exécution. Suite aux limitations de mémoire du gestionnaire clavier, cette commande doit être utilisée avec parcimonie.

Gestion de la manette de jeux

Positionne la manette en mode rapport de positionnement

Code : 14

Ce mode qui fournit au système toute modification de l'état de la manette est sélectionné par défaut lors de l'initialisation.

Positionne la manette en mode interrogation

Code : 15

Interrogation de la manette

Code : 16

Cette commande fournit une suite de trois octets déjà définie lors de la description des modes de la manette.

TABLES DES COMMANDES DU GESTIONNAIRE CLAVIER INTELLIGENT

Chaque valeur invalide est purement et simplement ignorée et ne modifie pas l'ancienne valeur.

Les valeurs sont codées en BCD (4 bits par chiffre).

Interrogation de l'horloge

Code : 1C

Cette commande retourne sept octets. Les six derniers sont en BCD.

Octet 1 = FC en-tête

Octet 2 = année

Octet 3 = mois

Octet 4 = jour

Octet 5 = heures

Octet 6 = minutes

Octet 7 = secondes

Ecriture dans la mémoire du contrôleur

Code : 20 ADRH ADRB NO O1 O2 O3 ...

Cette commande permet d'écrire une suite d'octets dans la mémoire du contrôleur. ADRH et ADRB expriment la valeur haute et la valeur basse de l'adresse de chargement ; NO exprime le nombre d'octets à écrire, O1 à On sont les octets à écrire.

Lecture de la mémoire du contrôleur

Code : 21 ADRH ADRB

Cette commande fournit huit octets qui sont constitués d'un en-tête de deux octets (F6 20) et de six valeurs consécutives lues. ADRH et ADRB sont les valeurs haute et basse de l'adresse où commence la lecture du premier des six octets.

Exécution d'une routine interne du contrôleur

Code : 22 ADRH ADRB

Cette commande exécute une routine interne du contrôleur située à l'adresse contenue dans ADRH et ADRB.

Rapport d'état

Code : 80+ le code de la commande dont on désire un compte rendu.

Cette commande fournit un compte rendu de la commande désirée.

Exemple : si on désire un compte rendu de la commande de positionnement de la manette en mode interrogation, il faut envoyer 80+15, soit 95, au gestionnaire.

TABLES DES COMMANDES DU GESTIONNAIRE CLAVIER INTELLIGENT

En retour, cette commande fournit huit octets définis comme suit :

- Octet 1 : F6 En-tête
- Octet 2 : numéro de la commande (15 par exemple)
- Octet 3 : paramètre 1
- Octet 4 : paramètre 2
- Octet 5 : paramètre 3
- Octet 6 : paramètre 4
- Octet 7 : paramètre 5
- Octet 8 : paramètre 6

Les paramètres équivalent aux paramètres passés en entrée lors de l'appel de la fonction. Si la fonction utilise moins de six paramètres, les paramètres excédentaires valent 0 et sont ignorés par le système lors du renvoi éventuel des paramètres.

Les commandes utilisables sont 87, 88, 89, 8A, 8B, 8C, 8F, 90, 92, 94, 95, 99 et 9A.

GENERALITES

Le Basic livré avec le ST à l'heure actuelle ne figure pas parmi les plus franches réussites du système.

Entièrement écrit en langage C, il utilise un vieux MATHPACK (ensemble de routines mathématiques) développé en assembleur par MOTOROLA. Cet ensemble de routines mathématiques est malheureusement peu précis et rend quasi impossible tout développement sérieux en Basic. Et pour couronner le tout, ce piètre interpréteur occupe près de 160 K de mémoire centrale, ce qui laissera, au pauvre possesseur d'un système sans ROM équipé de 512 K de mémoire centrale, la bagatelle de 9 K.

Les caractéristiques du Basic livré sont fortement inspirées de celles du GWBASIC de Microsoft disponible sur les PC de type IBM. Cependant, quelques absences de fonctions et d'instructions ainsi que de nombreuses erreurs internes (BUGS) empêchent une compatibilité totale entre les deux langages. La plupart des erreurs rencontrées, lors de l'élaboration de cet ouvrage, sont signalées en cours de description des mots clés. En outre, la structure même de l'interpréteur empêche l'édition classique dans une clef de la table des instructions internes et du format de stockage des programmes qui sont absolument inexploitable.

Pour ne pas être complètement négatif, signalons cependant que l'interfaçage avec GEM, au moyen d'appel de fonctions spécialisées, constitue une franche réussite et que l'on peut apprécier le système de gestion des fenêtres.

En résumé, nous devons reconnaître que, dans l'état actuel des choses et dans l'attente d'un Basic convenable, ce langage s'avère inutilisable pour un développement sérieux. Nous lui préférons donc le Pascal, le Modula2, le Fortran 77, le Forth, le Logo (par ailleurs excellent), l'assembleur ou mieux encore le langage C. Tous ces langages sont à l'heure actuelle disponibles et parfaitement satisfaisants.

CARACTERISTIQUES GENERALES DU BASIC

Constantes

Entière : -32768 à 32767 inclus
Simple précision : +- 5.421012 E-20 à +-5.764606 E18
Double précision : +- 5.421011 E-20 à +-5.764606 E18
Chaîne de caractères : 255 caractères

Numéro de ligne : 0 à 65529 inclus

Longueur d'une ligne : 255 caractères

Occupation de la mémoire

Variable entière : 10 octets minimum, 8 + 2 pour la valeur de la variable. Si le nom de la variable comporte plus d'un caractère, ajoutez un octet par caractère supplémentaire.

Variable simple précision : 12 octets minimum, 8 + 4 pour la valeur de la variable.

Variable double précision : 16 octets minimum, 8 + 8 pour la valeur de la variable.

Variable alphanumérique : 14 octets minimum.

Variable tableau : 24 octets minimum.

Taille maximum du nom d'une variable : 31 caractères.

Nombre de GOSUB imbriqués possible : 16.

CONSTANTES, VARIABLES ET OPERATEURS

Constantes

Les constantes Basic sont de quatre types :

- le type alphanumérique : limité à 255 caractères ;
- le type numérique entier : compris entre -32768 et +32767 ;
- le type numérique simple précision : compris entre $\pm 5.421012 \text{ E}-20$ et $\pm 5.764606 \text{ E}+18$. Exprimé sur 6 chiffres significatifs ;
- le type numérique double précision : compris entre les mêmes valeurs que le type simple précision. Exprimé sur 15 chiffres significatifs.

On peut exprimer les constantes numériques en hexadécimal en les faisant précéder de "&H", ou en octal en les faisant précéder de "&O".

Le point d'exclamation (!) permet de forcer le format simple précision et le dièse (#) permet de forcer le format double précision.

Variables

Comme les constantes, les variables sont de quatre types. En plus de la définition implicite possible grâce aux instructions DEFSTR, DEFINI, DEFSNG et DEFDBL, il est possible de définir le type de chaque variable en la faisant suivre par un signe défini comme suit :

- \$ pour définir le type alphanumérique ;
- % pour définir le type entier ;
- ! pour définir le type simple précision ;
- # pour définir le type double précision.

En outre, les variables peuvent être dimensionnées en faisant suivre leur nom de la liste de leurs dimensions entre parenthèses.

Opérateurs

Les opérateurs Basic sont les petits mots qui n'ont pas vraiment leur place parmi les instructions et les fonctions. Il s'agit principalement des opérateurs arithmétiques, relationnels et logiques.

- + Opérateur d'addition arithmétique.
- Opérateur de soustraction arithmétique.

CONSTANTES, VARIABLES ET OPERATEURS

- * Opérateur de multiplication arithmétique.
- / Opérateur de division arithmétique.
- ^ Opérateur d'exponentiation (élévation à une puissance).
- \ Opérateur de division entière.
- MOD Opérateur de modulo (reste de la division par).
- = Opérateur relationnel "égal".
- <> Opérateur relationnel "non égal".
- < Opérateur relationnel "plus petit".
- > Opérateur relationnel "plus grand".
- <= Opérateur relationnel "plus petit ou égal".
- >= Opérateur relationnel "plus grand ou égal".
- AND Opérateur logique ET.
- EQV Opérateur logique d'équivalence. C'est l'opérateur inverse du OU EXCLUSIF.
- IMP Opérateur logique d'implication.
- NOT Opérateur logique de négation ou inversion logique.
- OR Opérateur logique OU (un ou l'autre ou les deux).
- XOR Opérateur logique OU EXCLUSIF (un ou l'autre mais pas les deux).
- + Opérateur de concaténation de chaînes.

Abréviations utilisées

- adr : adresse située en mémoire.
- iuc : nombre représentant la valeur de l'incrément.
- n : nombre entier.
- nfich : nom de fichier.
- nl : numéro de ligne d'un programme Basic.
- np : numéro de port entrée/sortie.
- nufich : numéro du tampon sous lequel un fichier a été ouvert.
- var : variable quelconque.
- var\$: variable alphanumérique (chaîne de caractères).

CONSTANTES, VARIABLES ET OPERATEURS

<code>varnum</code>	: variable numérique (nombre).
<code>vart</code>	: variable tableau.
<code>x,y</code>	: coordonnées d'un point.
sélection de <code>nl</code>	: sélection de numéros de lignes. Cette sélection peut se faire de plusieurs façons: <code>nl</code> : le numéro de ligne <code>nl</code> . <code>-nl</code> : <code>nl</code> et toutes les lignes dont le numéro est plus petit que <code>nl</code> . <code>n11 - n12</code> : <code>n11</code> , <code>n12</code> et toutes les lignes dont le numéro est compris entre <code>n11</code> et <code>n12</code> . <code>n11, n12, n13,....</code> liste de tous les numéros de lignes à considérer.

Consignes

Les options se trouvant entre crochets [] sont facultatives.

COMMANDES ET INSTRUCTIONS BASIC

- AUTO** **AUTO** [nl] [,inc]
Numérote automatiquement les lignes d'un programme en commençant par le numéro de ligne (nl) spécifié et avec l'incrément (inc). Par défaut, le numéro de ligne et l'incrément valent 10. Ctrl-G permet de sortir du mode AUTO. AUTO n'est pas autorisé en mode EDIT. AUTO doit s'utiliser en mode commande.
- BLOAD** **BLOAD** "nfich" [,adr]
Permet de charger les programmes en langage machine et les variables ainsi que leur contenu précédemment sauvés au moyen de BSAVE. Si aucune adresse n'est spécifiée, le fichier chargé sera implanté à l'adresse spécifiée lors du BSAVE. Comme il est possible de charger des fichiers à n'importe quelle adresse, l'utilisateur doit veiller à ne pas entrer en conflit avec la partie mémoire utilisée par le Basic ou par ses programmes.
- BREAK** **BREAK** [sélection de nl]
Introduite avant le lancement d'un programme Basic, cette commande a pour effet d'en stopper l'exécution après chaque ligne. Un RETURN ou un CONT permettent de passer à l'exécution de la ligne suivante. Si des numéros de lignes sont spécifiés, l'exécution du programme ne stoppera qu'à ces dernières. A chaque arrêt, la ligne de programme et les sorties sont affichées à l'écran.
- BSAVE** **BSAVE** "nfich", adr, adr
Permet de sauver sur fichier une partie de la mémoire centrale (programmes en langage machine, données et images écran).
- CALL** **CALL** varnum [(var,var,var,...)]
Permet de transférer le contrôle à une routine en langage machine. L'adresse mémoire de la routine doit être contenue dans la variable numérique. Une série de paramètres (variables), entre parenthèses et séparés par des virgules, peuvent être transmis à la routine.
- CHAIN** **CHAIN** nfich [,nl] [,ALL]
CHAIN MERGE nfich [,nl] [,DELETE sélection de nl]
CHAIN permet de transférer le contrôle à un autre programme en lui passant les variables. Le nouveau programme remplace l'ancien dans la mémoire centrale. CHAIN MERGE ajoute le nouveau programme à l'ancien en remplaçant les lignes portant le même numéro par celle du nouveau programme. Le numéro de ligne facultatif après le nom du fichier permet de spécifier où commencer l'exécution du nouveau programme. Si ce numéro n'est pas indiqué, l'exécution démarre à la première ligne exécutable. L'option ALL permet de spécifier que toutes les variables doivent

COMMANDES ET INSTRUCTIONS BASIC

être transmises au nouveau programme. Si ALL n'est pas spécifié, il faudra utiliser la commande COMMON pour indiquer quelles variables sont communes aux deux programmes et seront donc passées au nouveau. L'option DELETE de CHAIN MERGE permet de supprimer certaines lignes de l'ancien programme avant d'effectuer le mélange.

- CIRCLE** CIRCLE x,y,rayon [,angle début, angle fin]
Trace un cercle dont le centre a pour coordonnées (x,y) et pour rayon, le rayon spécifié. Si on préfère ne dessiner qu'un arc de cercle, il faut spécifier les angle de début et angle de fin en dixièmes de degrés.
Exemple : CIRCLE 50,70,40,900,1800 dessinera un arc de cercle de centre (50,70), de rayon 40 et ayant pour limites 90° et 180°.
- CLEAR** CLEAR
Efface toutes les valeurs des variables de la mémoire centrale en y laissant le programme courant. A l'heure actuelle, cette commande ne fonctionne pas correctement avec les variables indicées.
- CLEARW** CLEARW n
Efface la fenêtre Basic numéro n. Les quatre fenêtres Basic sont:
0 - EDIT : fenêtre d'édition.
1 - LIST : fenêtre contenant le listing du programme.
2 - OUTPUT : fenêtre dans laquelle se font les sorties
 (résultats d'exécution de programme).
3 - COMMAND : fenêtre de commandes.
- CLOSE** CLOSE [[#]nufich [,[#]nufich,...]]
Referme le fichier ouvert sous le numéro spécifié et libère le tampon correspondant.
Lorsqu'aucun numéro de tampon n'est précisé, CLOSE ferme tous les fichiers ouverts et libère tous les tampons occupés.
Le signe # précédant le numéro de tampon est optionnel.
Les numéros de tampons doivent être compris entre 2 et 15, 15 étant le nombre maximum de fichiers que l'on peut ouvrir simultanément.
- CLOSEW** CLOSEW [n]
Ferme la fenêtre Basic numéro n. Si n n'est pas spécifié, CLOSEW fermera toutes les fenêtres Basic. Quatre fenêtres Basic sont à notre disposition, leur description se trouve à l'instruction CLEARW.
- COLOR** COLOR A,B,C [,D,E]
Etablit les couleurs du texte (A), de remplissage (B) et de dessin (C) ainsi que le motif de remplissage (D = style, E = index).
Les différentes couleurs disponibles sont les suivantes:

COMMANDES ET INSTRUCTIONS BASIC

<i>Couleur</i>	<i>N°</i>	<i>16 c.</i>	<i>4 c.</i>	<i>haute rés.</i>
BLANC	0	X	X	X
NOIR	1	X	X	X
ROUGE	2	X	X	
VERT	3	X	X	
BLEU	4	X		
BLEU MARINE	5	X		
BRUN	6	X		
VERT FONCE	7	X		
GRIS	8	X		
GRIS FONCE	9	X		
BLEU CLAIR	10	X		
BLEU VERT	11	X		
POURPRE CLAIR	12	X		
POURPRE FONCE	13	X		
JAUNE CLAIR	14	X		
JAUNE FONCE	15	X		

Les différents motifs de remplissage sont sélectionnés au moyen des paramètres style et index.

Index 0 - tout blanc, peu importe la valeur de style.

1 - tout noir, peu importe la valeur de style.

2 - 24 types de grillagés différents sélectionnés par le paramètre style.

3 - 12 types de hachures différentes sélectionnées par le paramètre style.

4 - modèle de remplissage représentant le logo d'Atari. La valeur du paramètre style n'a aucune influence sur ce modèle.

COMMON COMMON var1, var2, var3,....

Utilisé avant d'effectuer un CHAIN, COMMON permet de spécifier quelles variables doivent être passées au nouveau programme.

CONT CONT

Permet de continuer l'exécution d'un programme après un BREAK, un STOP ou un Ctrl C.

DATA DATA donnée1, donnée2, donnée3,....

Permet de stocker dans un programme une liste de constantes lues par l'instruction READ.

COMMANDES ET INSTRUCTIONS BASIC

- DEFDBL** DEFDBL var1-var2 DEFDBL var1,var,var3,...
Permet de définir un ensemble de variables dans la plage var1-var2 ou suivant la liste var1,var2,... comme étant du type double précision.
- DEF FN** DEF FN f[(x,...)]=expression
Permet de définir une fonction utilisateur. f représente le nom donné à la fonction, (x,...), ses paramètres formels et expression, son expression générale. (voir *Trucs et astuces*).
- DEFINT** DEFINT var1-var2 DEFINT var1,var2,...
Permet de définir un ensemble de variables dans la plage var1-var2 ou suivant la liste var1,var2,... comme étant de type entier (compris entre -32768 et +32767).
- DEF SEG** DEF SEG n
Définit l'offset pour les instructions PEEK et POKE. 3 cas sont possibles:
- 1- $n < 0$: PEEK (adr) fournira le contenu exprimé sur 8 bits de l'adresse adr+n. De même, POKE octet, adr introduira l'octet spécifié à l'adresse adr + n.
 - 2- $n = 0$: PEEK (adr) fournira le contenu exprimé sur 16 bits des adresses adr et adr+2 (256 fois le contenu de adr + le contenu de adr+1). Bien entendu, POKE réagit de la même façon.
 - 3- $n <= 0$ et adr spécifiée en double précision, PEEK fournira un entier exprimé sur 32 bits et qui correspond au contenu des adresses adr, adr+1, adr+2 et adr+3. POKE réagit de la même façon.
- DEFSNG** DEFSNG var1-var2 DEFSNG var1,var2,...
Définition d'un ensemble de variables dans la plage var1-var2 ou suivant la liste var1,var2,... comme étant de type SIMPLE PRECISION.
- DEFSTR** DEFSTR var1-var2 DEFSTR var1,var2,...
Définition d'un ensemble de variables dans la plage var1-var2 ou suivant la liste var1,var2,... comme étant de type ALPHANUMERIQUE (chaines de caractères).
- DEFUSR** DEFUSRn = adr
Définition standard d'un point d'entrée d'une routine en langage machine (à utiliser conjointement avec la fonction USR). Cette fonction a été implantée pour assurer une compatibilité avec le Basic MICROSOFT. Hélas, elle ne fonctionne pas correctement. En outre, il faut lui préférer l'instruction CALL.

COMMANDES ET INSTRUCTIONS BASIC

- DELETE** DELETE [sélection de nl]
Permet de supprimer de la mémoire centrale les lignes de programme dont les numéros sont spécifiés.
Le 1° exemple: suppression d'une ligne.
Le 2° exemple: suppression de toutes les lignes dont les numéros sont plus petits ou égaux à nl.
Le 3° exemple: suppression de toutes les lignes dont les numéros sont compris entre nl1 et nl2.
Le 4° exemple: suppression de toutes les lignes dont les numéros sont spécifiés dans la liste nl1,nl2,.....
- DIM** DIM var(n) [,var(m)..]
DIM var(n1,...,nX) [,var(n1,..)]
Dimensionnement d'un tableau (var) de 1 à X dimensions. Par défaut, une variable est automatiquement dimensionnée à 10 (var(10)). Plusieurs variables peuvent être dimensionnées ensembles. OPTION BASE permet de déterminer si l'élément 0 doit être considéré ou non.
- DIR** DIR [lecteur:] [/chemin/] [nfich.type]
Fournit une liste des fichiers présents sur le disque spécifié. Le résultat apparaît sur la fenêtre de commandes.
DIR fournira la liste de tous les fichiers présents sur le disque courant et DIR A:, sur le lecteur A.
DIR B:BAS.PRG fournira la liste de tous les programmes Basic présents sur le lecteur B.
DIR A:BAS.* fournira la liste des fichiers BAS de tous types présents sur le lecteur A.
DIR B:*.PRG fournira la liste de tous les fichiers de type .PRG présents sur le lecteur B.
DIR A:*. * fournira la liste de tous les fichiers de tous les types présents sur le lecteur A.
DIR A:BAS.PR? fournira la liste des fichiers BAS et dont l'extension commence par PR.
- EDIT** EDIT [nl] ED [nl]
Introduit le programme Basic dans l'éditeur à partir de la ligne numéro nl ou de la première ligne si nl n'est pas spécifié.
- ELLIPSE** ELLIPSE x,y, rayon horizontal, rayon vertical[, angle début,angle fin]
Trace une ellipse dont le centre a pour coordonnées (x,y) et pour rayons, les rayons horizontal et vertical spécifiés. Pour ne dessiner qu'une partie de l'ellipse, il faut spécifier les angle de début et angle de fin en dixièmes de degrés.

COMMANDES ET INSTRUCTIONS BASIC

Exemple :

```
ELLIPSE 50,70,40,60,900,1800
```

tracera un arc d'ellipse dont le point central a pour coordonnées (50,70), le rayon horizontal, une longueur de 40, le rayon vertical, une longueur de 60. Les limites de cet arc seront comprises entre 90° et 180°.

END

END

Introduite dans un programme, cette instruction a pour effet de stopper l'exécution du programme, de fermer les fichiers et de retourner au niveau de commandes.

Exemple :

```
10 INPUT A
20 IF A=6 THEN END
30 PRINT....
```

ERA

ERA [lecteur:] nfich

Efface le fichier disque dont le nom est précisé. Si le lecteur n'est pas précisé, le lecteur courant est considéré.

ERASE

ERASE vart, vart, vart,.....

Permet de libérer la place mémoire retenue par la commande DIM. Elimine les variables dimensionnées spécifiées de la mémoire centrale.

ERROR

ERROR n

Simule l'arrivée de l'erreur numéro n.

FIELD

FIELD #nufich, liste de champs

Alloue et découpe l'espace occupé par les champs variables d'un fichier direct. nufich représente le nombre sous lequel le fichier a été ouvert (*tampon*, voir *OPEN*). La liste des champs décrits se présente sous la forme : nombre AS variable alphanumérique. Le nombre représente le nombre d'octets réservés à la variable alphanumérique.

Exemple :

```
10 OPEN "R", #2, "ADRESSES", 80
20 FIELD #2, 20 AS N$, 15 AS PR$, 30 AS RU$, 5 AS
CPS$, 10 AS V$
```

Un fichier d'adresses est ouvert et permet 20 caractères pour le nom (N\$), 15 pour le prénom (PR\$), 30 pour la rue (RU\$), 5 pour le code postal (CPS) et 10 pour la ville (V\$).

FILL

FILL x,y [,couleur]

Permet de remplir des figures jusqu'à la couleur désignée ou au moyen du motif prévu lors de l'instruction *COLOR* (voir *COLOR*). x et y représentent les coordonnées d'un point situé à l'intérieur de la surface à remplir.

COMMANDES ET INSTRUCTION BASIC

Exemple :

```
10 COLOR 1,2,1
20 CIRCLE 100,100,70
30 FILL 100,100
40 COLOR 1,1,1,4,4
50 FILL 100,100
```

FOLLOW FOLLOW var [,var]

Permet de suivre l'évolution de la valeur d'une variable au cours de l'exécution d'un programme. Chaque fois que la valeur d'une variable suivant FOLLOW change de valeur, son nom, sa nouvelle valeur et le numéro de ligne, où cela se passe, apparaissent.

La commande UNFOLLOW permet de stopper l'effet de FOLLOW.

FOR FOR varnum = X to Y [STEP Z]

.....
NEXT varnum

Introduit une boucle. Toutes les instructions, comprises entre un FOR et le NEXT correspondant, seront répétées pour toutes les valeurs de var allant de X à Y par pas de Z (ou de 1 si Z n'est pas précisé).

Exemple :

```
10 FOR I=5 TO 100 STEP 5
20 A = I * I
30 PRINT I;" au carré = ";A
40 NEXT I
RUN
5 au carré = 25
10 au carré = 100
```

.....
100 au carré = 10000

REM : plusieurs boucles peuvent se trouver imbriquées mais elles ne peuvent jamais se chevaucher:

Exemples corrects :

```
10 FOR I = A TO B
20 FOR J = C TO D
30 FOR K = E TO F
.....
90 NEXT K
100 NEXT J
110 NEXT I
```

```
10 FOR I = A TO B
20 FOR J = C TO D
30 FOR K = E TO F
.....
90 NEXT K,J,I
```

COMMANDES ET INSTRUCTIONS BASIC

Exemple incorrect :

```
10 FOR I = A TO B
20 FOR J = C TO D
30 FOR K = E TO F
90 NEXT I
100 NEXT J
110 NEXT K
.....
```

FULLW FULLW n
Permet d'allouer tout l'écran à la fenêtre Basic numéro n (0 = édition, 1 = List, 2 = sorties, 3 = commandes).

GEMSYS GEMSYS (n)
Cette instruction est particulière au système ST. Elle permet d'appeler une routine interne du GEM AES. L'utilisation complète de cette instruction est décrite dans une section particulière de ce chapitre.

GET GET [#]nufich [,n]
Lit un enregistrement spécifié par le nombre n depuis le fichier direct ouvert sous le numéro nufich. Si n n'est pas précisé, c'est l'enregistrement qui suit le dernier enregistrement lu ou écrit qui est accédé. Voir OPEN pour l'exemple.

GOSUB GOSUB nl GOSUB LABEL
Envoie le contrôle à la sous-routine située à la ligne numéro nl ou à la ligne commençant par le label désigné.

Exemple :

```
10 GOSUB 1000
20 PRINT "au revoir"
30 END
1000 PRINT "début de la sous-routine"
1010 GOSUB MERCI
1020 PRINT "fin"
1030 RETURN
2000 MERCI: PRINT "encore vous"
2010 RETURN
RUN
début de la sous-routine
encore vous
fin
au revoir
```

GOTO GOTO nl GOTO LABEL
Effectue un saut à la ligne portant le numéro nl ou à la ligne commençant par le label désigné.

COMMANDES ET INSTRUCTIONS BASIC

Exemple :

```
10 PRINT "bonjour"
20 GOTO 100
30 PRINT "voici un bête programme"
100 PRINT "on est à 100"
110 GOTO JOLI
120 PRINT "ceci est un programme idiot"
130 JOLI
140 PRINT "voulez-vous des fleurs ?"
150 END
RUN
bonjour
on est à 100
voulez-vous des fleurs ?
```

GOTO LABEL n'est pas particulièrement fiable.

GOTOXY GOTOXY colonne,ligne

L'instruction GOTOXY permet de positionner le curseur à un endroit précis sur l'écran en précisant la colonne (0 à 79 en monochrome ou 0 à 39 en couleur) et la ligne (0 à 24) où l'affichage doit avoir lieu. Des conflits entre la gestion des fenêtres et cette instruction produisent parfois des résultats aberrants. Cette instruction n'est donc pas fiable.

IF IF condition THEN instruction [ELSE instruction]

Effectue l'instruction suivant le THEN si la condition spécifiée est vraie. Si la condition est fautive et que la forme ELSE est utilisée, alors, l'instruction suivant ELSE est effectuée. S'il n'y a pas de ELSE et que la condition est fautive, la ligne suivante sera effectuée.

Exemple :

```
10 INPUT A
20 IF A > 10 THEN PRINT "c'est beaucoup" ELSE
PRINT "c'est peu"
30 END
```

INPUT INPUT var [,var,...]

INPUT "blabla";var [,var,...]

Lit les données en provenance du clavier et les affecte aux variables spécifiées.

Exemple :

```
10 INPUT "combien en voulez-vous ";A
20 INPUT "quels sont vos nom et prénom ";BS,CS
30 PRINT A:PRINT BS:PRINT CS
```

REM: on est obligé de répondre à un INPUT. En effet, si on répond par un Enter, un message d'erreur apparaît. Le séparateur entre le texte de l'input et le nom de la variable peut

COMMANDES ET INSTRUCTIONS BASIC

être soit une virgule, soit un point-virgule. Dans le cas d'un point-virgule, un ? et un espace seront ajoutés entre le texte et la réponse.

```
10 INPUT "combien";A
20 INPUT "et encore",B
RUN
combien? 440
et encore56
```

- INPUT#** INPUT # nufich, var1, var2, ...
Lit les enregistrements en provenance d'un fichier séquentiel. Les différents enregistrements lus seront affectés aux variables spécifiées.
Voir OPEN pour l'exemple (exemple 2, lecture de fichier séquentiel).
- KILL** KILL "nfich"
Permet de supprimer un fichier disque pour autant qu'il soit fermé. A l'inverse d'ERASE, KILL peut être utilisé à l'intérieur d'un programme Basic.
- LET** LET var = expression
Affecte à la variable le résultat de l'expression située à droite du signe égal.
LET A = 500 * 3
On peut bien entendu se contenter d'écrire :
A\$ = 500 * 3
LET est utilisé dans les cas où l'on désire qu'un programme reste compatible avec d'anciens programmes.
- LINEF** LINEF x1,y1,x2,y2
Trace une ligne du point (x1,y1) au point (x2,y2) en respectant le mode ligne décrit par l'instruction COLOR.
REM : l'absence insolite d'une fonction de traçage de point (PSET) oblige l'utilisateur à employer :
LINEF x1,y1,x1,y1
ou alors, l'instruction CIRCLE avec un rayon de 1.
- LINE INPUT** LINE INPUT "chaîne", var\$
LINE INPUT #nufich, var\$
Lit une ligne entière dans une variable alphanumérique même si cette ligne contient des séparateurs (virgule, point, ...). La première forme lit le clavier et permet l'affichage préalable d'une chaîne de caractères. La seconde forme effectue une lecture du contenu du fichier séquentiel ouvert sous le numéro n.
- LIST** LIST [sélection de n]
Inscrit tout le programme ou les lignes de programme spécifiées sur la fenêtre LIST.

COMMANDES ET INSTRUCTIONS BASIC

Exemple :

LIST 10	listera la ligne 10.
LIST 10-100	listera les lignes dont les numéros sont compris entre 10 et 100.
LIST 10,30,50,60	listera les lignes précisées.
LIST - 80	listera les lignes dont les numéros sont plus petits ou égaux à 80.
LIST 10-30, 70,100	

LLIST LLIST [sélection de nl]
Comme LIST mais les lignes de programmes seront imprimées par l'imprimante.

LOAD LOAD nfich
Charge un programme Basic en mémoire centrale. LOAD suppose une extension .BAS même si elle n'est pas spécifiée. Le chargement d'un nouveau programme a pour effet de supprimer celui qui se trouvait dans la mémoire centrale auparavant. La commande OLD fonctionne de la même façon que LOAD.

LPRINT LPRINT [liste d'expressions]
LPRINT USING format ; liste d'expressions
Commande permettant une impression directe.
Les expressions de la liste d'expressions suivant la commande LPRINT USING doivent être séparées par des virgules.

Exemple :

```
LPRINT DS; " vaut "; N  
LPRINT USING FS; AS, N, M
```

LSET LSET var\$ = var\$
Permet de placer une variable dans le buffer d'un fichier en justifiant son contenu à gauche.
Voir OPEN pour l'exemple.

MERGE MERGE nfich
Insère un programme Basic en provenance du disque dans le programme présent en mémoire centrale. Lorsqu'une ligne du programme disque porte le même numéro qu'une ligne du programme de la mémoire centrale, elle la remplace. *Voir CHAIN.*

MID\$ MID\$ (var\$1, départ, longueur, var\$2)
Remplace une partie de var\$2 (expression alphanumérique) par une autre, contenue dans var\$1.

Exemple :

```
10 AS = "ABCDEFGHijkl"  
20 RS = "ZZZZ"  
30 MID$ (AS,2,4) = RS
```

COMMANDES ET INSTRUCTIONS BASIC

```
40 PRINT AS  
RUN  
AZZZZFGHIJKL
```

Si la longueur de var\$2 est inférieure à la longueur prévue, la longueur considérée sera celle de var\$2 :

```
30 MID$(AS,2,7) = RS  
RUN  
AZZZZFGHIJKL
```

Si la longueur de var\$2 est supérieure à la longueur prévue, seule une partie de var\$2 sera introduite dans var\$1 :

```
30 MID$(AS,2,2) = RS  
RUN  
AZZDEFGHIJKL
```

- NAME** NAME "nfich1" AS "nfich2"
Permet de changer le nom d'un fichier. nfich1 s'appellera désormais nfich2. Le fichier que l'on désire renommer doit exister et le nouveau nom que l'on désire lui attribuer ne peut pas être le nom d'un autre fichier existant, sinon, une erreur apparaît.
- NEXT** NEXT varnum [,varnum...]
Termine une boucle FOR. Voir FOR pour l'exemple.
- NEW** Commande qui a pour effet d'effacer le programme présent en mémoire centrale. Si le programme effacé n'a pas été sauvé sur disque avant d'effectuer la commande NEW, il est perdu.
- La commande NEW n'efface pas les valeurs internes des tableaux dimensionnés. Un bug de plus dans le Basic.
- ON ERROR ON ERROR GOTO n1**
Spécifie le numéro de ligne à laquelle l'exécution du programme doit continuer lorsqu'une erreur se produit. Voir RESUME.
- ON...GOTO** ON varnum GOTO n11, n12, n13, ...
ON...GOSUB ON varnum GOSUB n11, n12, n13, ...
Produit un branchement à une des lignes spécifiées dans la liste en fonction de la valeur de la variable varnum.
Si varnum vaut 1, le branchement se fera à n11,
Si varnum vaut 2, le branchement se fera à n12, ...
Si varnum vaut 0 ou un nombre supérieur au nombre de lignes précisées, il n'y aura pas de branchement. Les numéros de lignes peuvent être remplacés par des labels. Le branchement se fera alors à la ligne commençant par le label désigné. Voir GOTO et GOSUB.
- OLD** OLD nfich
Fonctionne comme LOAD.

COMMANDES ET INSTRUCTIONS BASIC

OPEN OPEN "mode", # nufich, "nfich.DAT", n
Ouvre un fichier en vue d'y lire ou d'y écrire.
mode : O : fichier séquentiel dans lequel on va écrire.
I : fichier séquentiel dans lequel on va lire.
R : fichier accès direct (Random) dans lequel on va lire et écrire.
nufich : représente le numéro du buffer qui servira d'intermédiaire entre la mémoire centrale et le disque contenant le fichier. 15 buffers sont à la disposition de l'utilisateur. nufich sera donc un nombre compris entre 1 et 15.
nfich.DAT: l'extension .DAT signifie que le fichier est un fichier contenant des données. Si .DAT n'est pas spécifié, l'ordinateur l'ajoute automatiquement.
n: nombre représentant la taille d'un enregistrement. Dans le cas de fichiers séquentiels, il est ignoré puisque la taille des enregistrements séquentiels est variable. Dans le cas d'un fichier à accès direct (Random), n vaut 128 par défaut.

Exemples :

1 - Manipulation d'un fichier à accès direct

```
10 OPEN "R", #1, "LISTE",35
20 FIELD #1, 20 AS NOM$, 15 AS PRENOM$
30 I = 1
40 INPUT "NOM";A$
50 IF AS="FINI" THEN CLOSE:END
60 INPUT "PRENOM";BS
70 LSET NOM$=A$
80 LSET PRENOM$=BS
90 PUT #1,I
100 I = I+1:GOTO 40
RUN
NOM? BALTHASART
PRENOM? ANNE
NOM? MARTIN
PRENOM? DANIEL
NOM? KORADY
PRENOM? NATHALIE
NOM? MAIGNAN
PRENOM? DIDIER
NOM? FINI
NEW
10 OPEN "R", #1,"LISTE",35
20 FIELD #1, 20 AS NOM$, 15 AS PRENOM$
30 INPUT "NUMERO DE FICIE A LIRE";A
```

Ce premier programme nous permet d'ouvrir un fichier et d'y introduire quelques enregistrements.

Le programme suivant nous permettra de relire nos enregistrements.

Ce NEW nettoie la mémoire.

COMMANDES ET INSTRUCTIONS BASIC

```
40 GET #1, A
50 PRINT NOMS,PRENOMS
60 INPUT "ENCORE (OUI OU NON)";BS
70 IF LEFT$(BS,1) = "O" THEN 30
80 IF LEFT$(BS,1) = "N" THEN CLOSE:END
90 GOTO 60
RUN
NUMERO DE FICIE A LIRE? 2
MARTIN DANIEL
ENCORE (OUI OU NON)? .....
...
```

2 - Manipulation d'un fichier séquentiel

```
10 OPEN "O",1,"MEMBRES"
20 INPUT "NOM";N$
30 IF N$="FINI" THEN CLOSE:END
40 PRINT #1,N$
50 GOTO 20
RUN
NOM? TARTEMPION
NOM? DUPONT
NOM? HADDOCK
NOM? TINTIN
NOM? FINI
NEW
```

Ce premier programme ouvre un fichier séq. en écriture.

```
10 OPEN "I",1,"MEMBRES"
20 WHILE NOT EOF(1)
30 INPUT #1,A$
40 PRINT A$
50 WEND
60 CLOSE:END
RUN
TARTEMPION
DUPONT
HADDOCK
TINTIN
```

Le fichier séquentiel sera ouvert en lecture.

OPENW

OPENW n

Ouvre la fenêtre Basic précédemment fermée au moyen de la commande CLOSEW.

ou : OPENW n,(x1,y1,x2,y2)

Ouvre la fenêtre entre les points de diagonale x1,y1 et x2,y2.

Suite à un défaut de fonctionnement, cette instruction ne peut être utilisée qu'une seule fois.

OPTION BASE OPTION BASE n avec n= 0 ou 1

Détermine l'indice minimum d'une variable tableau. Voir DIM.

COMMANDES ET INSTRUCTIONS BASIC

OUT **OUT** np,n
Envoie le nombre entier n vers le port désigné (np = numéro de port). n peut être compris entre 0 et 255 (1 octet).
Les ports de l'Atari ST sont :
0 = PRINTER (port parallèle imprimante)
1 = AUX (RS232)
2 = CONSOLE (écran)
3 = MIDI (Musical Instrument Digital Interface)
4 = KEYBOARD (clavier)

PCIRCLE **PCIRCLE** x,y,rayon[,angle de départ,angle d'arrivée]
Comme **CIRCLE**, mais les cercles et arcs de cercles dessinés sont pleins. Ce remplissage se fait suivant le motif décrit dans l'instruction **COLOR**.

Exemple :

```
COLOR 1,2,1
PCIRCLE 100,90,50
PCIRCLE 50,50,40,300,900
CLEARW 2
COLOR 1,2,1,2,2
PCIRCLE 100,90,50
```

PELLIPSE **PELLIPSE** x,y, rayon horizontal, rayon vertical
[,angle de départ, angle d'arrivée]
Comme **ELLIPSE**, mais les ellipses et arcs d'ellipses sont remplies avec le motif décrit lors de l'instruction **COLOR**.

Exemple :

```
COLOR 1,2,1,2,2
PELLIPSE 50,80,100,50,900,1800
```

POKE **POKE** adr,n
Inscrit une donnée n à l'adresse mémoire adr précisée. Voir **PEEK** et **DEF SEG**.

PRINT **PRINT** expression
Affiche l'expression désignée à l'écran (fenêtre **Basic 2**, **Output**). Un ? peut être utilisé au lieu du mot **PRINT**. La ponctuation (, ;) utilisée déterminera la position des données sur l'écran.

Démonstration :

```
10 PRINT "DEMONSTRATION"
20 PRINT
30 AS = "LUNDI" : BS = "MARDI" : CS = "MERCREDI"
40 A = 87 : B = 560 : C = 1000
50 PRINT AS;BS;CS
60 PRINT A$,B$,C$
```

COMMANDES ET INSTRUCTIONS BASIC

```
70 PRINT A;B;C
80 PRINT A,B,C
90 END
RUN
DEMONSTRATION
```

```
LUNDIMARDIMERCREDI
LUNDI      MARDI      MERCREDI
87 560 1000
8860 1000
```

PRINT # PRINT #nufich, var\$ [,var\$,...]
Permet l'écriture de données dans un fichier séquentiel ouvert en écriture sous le numéro nufich. Voir *OPEN* pour l'exemple (exemple 2, fichier séquentiel ouvert en écriture).

PRINT USING PRINT USING chaîne ; liste d'expressions ;
PRINT #nufich ,USING chaîne ; liste d'expressions
La première syntaxe imprime sur la console la liste d'expressions suivant un format fixé par la chaîne de caractères qui suit l'using. La seconde syntaxe utilise la même méthode de mise en format mais pour l'écriture dans un fichier séquentiel ouvert sous le numéro nufich. Pour la chaîne, les conventions sont les suivantes :

- ! indique que seul le premier caractère de chaque élément de la liste doit être affiché.
- \n espaces\ indique que n+2 caractères de chaque élément de la liste doivent être imprimés.
- & indique que chaque élément doit être imprimé tel quel.
- # représente un chiffre d'un nombre.
- . représente le point décimal d'un nombre.
- + indique que le signe du nombre à afficher doit être affiché.
- indique que les nombres négatifs doivent être suivis du signe -.
- ** indique qu'il faut remplir les espaces vides d'un nombre avec des astérisques.
- \$\$ indique qu'il faut remplir les espaces vides d'un nombre avec des dollars.
- **\$ indique qu'il faut remplir les espaces vides avec des astérisques et remplacer le dernier espace vide avant le nombre par un dollar.
- ,
- ^ indique qu'il faut utiliser la notation exponentielle pour représenter le nombre.
- % est produit par l'instruction si le nombre à afficher dépasse le format spécifié.

COMMANDES ET INSTRUCTIONS BASIC

- fait imprimer indéfiniment au système des tirets. La seule façon de s'en sortir est d'effectuer une réinitialisation générale. Donc : A NE PAS UTILISER, c'est encore une erreur du Basic Atari !
- PUT** **PUT** [#] nufich , n
Ecrit l'enregistrement de numéro n dans le fichier direct ouvert sous le numéro nufich.
- QUIT** **QUIT**
Ferme tous les fichiers, quitte le Basic ST et retourne au GEM.
- RANDOMIZE** **RANDOMIZE** [n]
Fixe la séquence des nombres pseudo-aléatoires à partir de n, n étant un entier compris entre -32768 et 32767. Si n est omis, ST Basic le demande.
- READ** **READ** var1, var2, var3,...
Lit les données contenues dans les lignes de DATAs et les affecte aux différentes variables précisées.
- REM** **REM** remarques
Introduit une ligne de commentaires.
- RENUM** **RENUM** [n1] [,n2] [,inc]
Permet de renuméroter les lignes d'un programme. Par défaut, les paramètres sont les suivants :
 RENUM 10,10,10
n1 fournit le premier numéro de ligne de la nouvelle numérotation.
n2 fournit le numéro de ligne de l'ancienne numérotation à partir de laquelle il faut commencer à renuméroter.
inc fournit l'incrément de la nouvelle numérotation.

Exemple :

```
10 REM
20 A$="BLABLA"
30 INPUT B
31 INPUT C
44 PRINT A$
45 REM CECI EST UN BETE PROGRAMME
RENUM 31,40,10
LIST
10 REM
20 A$="BLABLA"
30 INPUT B
40 INPUT C
50 PRINT A$
60 REM CECI EST UN BETE PROGRAMME
```

COMMANDES ET INSTRUCTIONS BASIC

REPLACE REPLACE [[nfich] [,sélection de nl]]
Remplace l'ancien contenu d'un fichier par une nouvelle version. Le fichier concerné doit avoir été chargé au préalable au moyen de la commande LOAD ou de la commande OLD. Après modifications, REPLACE remplacera sur le disque l'ancien contenu par le nouveau. Si un nom de fichier est précisé, l'ancien contenu sera copié sous ce nom. Si une liste de numéros de lignes est précisée, seules ces lignes seront recopiées. Cette instruction est à prescrire totalement car un défaut de conception écrase souvent le fichier cible.

RESET RESET
Copie le contenu de la fenêtre Output (2) dans le tampon graphique dans le but soit de le sauver sur disque, soit de le récupérer par la suite par un OPENW 2.

Exemple :

```
10 COLOR 1,2,1,2,2
20 PCIRCLE 110,100, 70
30 RESET
40 CLEARW 2
50 PELLIPSE 100,100,50,70,0,9000
60 FOR I=1 TO 1000:NEXT
70 OPENW 2:END
```

RESTORE RESTORE [nl]
Définit la ligne à partir de laquelle la lecture des DATAs doit commencer ou recommencer. Si aucun numéro de ligne n'est précisé, la lecture commence à la première ligne contenant des DATAs.

RESUME RESUME [nl] RESUME 0 RESUME NEXT
Permet à l'exécution du programme de continuer à partir de la ligne dont le numéro est spécifié après qu'une erreur ait été décelée et corrigée par une commande ON ERROR GOTO. Si le numéro de ligne est omis ou vaut 0, l'exécution est reprise à l'instruction qui a provoqué l'erreur. Si RESUME NEXT est utilisé, l'exécution est reprise à l'instruction qui suit l'instruction qui a provoqué l'erreur.

RETURN RETURN
Provoque le retour à la routine principale à la fin d'une sous-routine appelée par un GOSUB, ou un ON...GOSUB.

RSET RSET var\$ = "chaîne"
Installe une valeur alphanumérique (chaîne) dans un tampon en vue d'une écriture sur disque (PUT). var\$ est la variable présente à la ligne FIELD. La chaîne est cadrée à droite dans le tampon, tandis que LSET la cadre à gauche. Voir OPEN pour l'exemple.

COMMANDES ET INSTRUCTIONS BASIC

TRON de ne prévenir que lors du passage de l'exécution par ces lignes. TROFF désactive TRON.

UNBREAK UNBREAK [sélection de n]

Désactive la commande BREAK soit pour toutes les lignes, soit pour les lignes précisées. Voir BREAK.

UNFOLLOW UNFOLLOW [var] [,var]

Désactive la commande FOLLOW pour toutes les variables ou pour les variables précisées. Voir FOLLOW.

UNTRACE UNTRACE [sélection de n]

Désactive la commande TRACE pour toutes les lignes ou pour les lignes précisées. Voir TRACE.

VDISYS VDISYS [(n)]

Instruction propre à l'Atari ST. Elle permet d'appeler une routine interne du GEM VDI. Sa description complète est donnée dans une section particulière de ce chapitre. Le paramètre n'est pas optionnel. Son absence peut cependant produire des erreurs.

WAIT WAIT np , masque1 [,masque2]

Suspend l'exécution d'un programme jusqu'à ce que la valeur lue sur le port d'entrée / sortie spécifié soit différente de 0. Avant de tester cette valeur, on lui applique une fonction ET logique avec le masque 1, et optionnellement, une fonction OU EXCLUSIF logique avec le masque2.

WAVE WAVE mixer, enveloppe, forme, période, délai

Cette instruction permet de modifier les paramètres internes du circuit PSG constituant le générateur sonore. Une bonne connaissance de son fonctionnement nécessite une étude préalable de la structure interne des registres du PSG. Reportez-vous au chapitre concernant la description des principaux circuits pour comprendre le sens des paramètres qui suivent.

Le paramètre mixer permet de programmer le contenu du registre interne 7 du PSG.

Le paramètre enveloppe agit sur le bit 4 des registres 8 à 10 du PSG. Le bit 0 du paramètre agit sur le bit 4 du registre 8, le bit 1 agit sur le bit 4 du registre 9 et le bit 2 sur le bit 4 du registre 10. Le paramètre forme agit sur le registre 13 du PSG.

Le paramètre période permet de programmer le contenu des registres 11 et 12 du PSG. Ce paramètre est donc un nombre exprimé sur 16 bits.

Le paramètre délai agit sur le temps entre 2 productions sonores.

WEND WEND

Indique la fin d'une boucle WHILE.

COMMANDES ET INSTRUCTIONS BASIC

WHILE WHILE expression logique
corps de la boucle
WEND
Tant que l'expression logique sera vraie, le corps de la boucle compris entre WHILE et WEND sera exécuté.

Exemple le programme suivant :

```
10 I = 1
20 WHILE I < 11
30 PRINT I
40 I = I + 1
50 WEND
```

fera la même chose que :

```
10 FOR I = 1 TO 10
20 PRINT I
30 NEXT I
```

WIDTH WIDTH [LPRINT] n
Positionne la taille en nombre de caractères par ligne (n) de l'écran ou de l'imprimante. n peut prendre toutes les valeurs comprises entre 14 et 255.

Exemple :

```
10 WIDTH 20
20 FOR I = 1 TO 30
30 PRINT "-";
40 NEXT
50 RUN
```


WRITE WRITE [liste d'expressions]
Imprime des données sur l'écran. WRITE insère des virgules entre les différents éléments de la liste et encadre les valeurs alphanumériques entre guillemets.

Exemple :

```
A$ = "COUCOU"
WRITE A$
"COUCOU"
```

WRITE # WRITE #nufich [,liste d'expressions]
Comme WRITE, mais au lieu d'inscrire les données à l'écran, WRITE # les inscrit dans un fichier séquentiel ouvert sous le numéro nufich.

- ABS** **ABS (n)**
Fournit la valeur absolue de l'expression numérique entre parenthèses.
- ASC** **ASC (C\$)**
Fournit le code ASCII du premier caractère contenu dans la chaîne de caractères entre parenthèses.
REM : si le premier caractère est accentué (ASC (é)), le message d'erreur 5 apparaît : fonction call not allowed.
- ATN** **ATN (n)**
Fournit la valeur en radians de l'angle dont la tangente vaut n.
- CDBL** **CDBL (n)**
Convertit un nombre en double précision.
- CHR\$** **CHR\$ (n)**
Fournit le caractère dont le code ASCII est n. n est un nombre entier compris entre 0 et 255. En fait on peut mettre un nombre plus grand, la fonction le convertit automatiquement en un nombre entre 0 et 255 par MOD 256.
- CINT** **CINT (n)**
Convertit une expression numérique en un nombre entier en arrondissant à l'unité supérieure lorsque la partie décimale de l'expression est supérieure ou égale à 0,5.
- CONTRL** **CONTRL**
Variable interne qui fournit l'adresse interne de la table de contrôle d'entrée/sortie pour l'appel du GEM VDI par l'instruction VDISYS.
- COS** **COS (n)**
Fournit la valeur du cosinus d'un angle en supposant que celui-ci (n) est exprimé en radians.
- CSNG** **CSNG (n)**
Convertit un nombre entier en un nombre réel simple précision.
- CVD** **CVD (chaîne de caractères de 8 octets)**
Convertit la chaîne de caractères en un nombre double précision (inverse de MKD\$).
- CVI** **CVI (chaîne de caractères de 2 octets)**
Convertit une chaîne de caractères de 2 octets en un nombre entier (inverse de MKIS).
- CVS** **CVS (chaîne de caractères de 4 octets)**
Convertit la chaîne de caractères en un nombre simple précision (inverse de MKS\$).

FONCTIONS BASIC

- EOF** EOF (nufich)
Indicateur de fin de fichier séquentiel. Prend la valeur -1 (TRUE) lorsque la fin du fichier est atteinte et 0 dans les autres cas.
Voir OPEN pour exemple.
- ERR** ERR
Variable contenant le numéro de la dernière erreur qui s'est produite.
- ERL** ERL
Variable contenant le numéro de la ligne où la dernière erreur s'est produite.
- EXP** EXP (n)
Calcule e exposant n (exponentielle).
- FIX** FIX (n)
Enlève la partie décimale du nombre n sans arrondir à l'entier le plus proche.
- FLOAT** FLOAT (n)
Convertit un nombre entier en un nombre réel simple précision.
- FRE** FRE (0)
Fournit le nombre d'octets restés libres en mémoire centrale.
- GB** GB
Variable interne qui fournit l'adresse interne d'une table de contrôle d'une liste d'adresse pour l'appel du GEM AES par l'instruction GEMSYS. GB doit être impérativement en double précision.
- HEX\$** HEX\$ (n)
Convertit le nombre entier n en un nombre hexadécimal sous forme de chaîne alphanumérique.
!!! Si le nombre entier n est représenté par une variable non entière, HEX\$ ne fonctionne pas !!! encore une erreur du Basic ST !
- INKEYS** Cette fonction est censée scruter le clavier pendant un court instant. Elle est hélas totalement inopérante.
- INP** INP (np)
Lit le contenu (1 octet) du port d'entrée/sortie spécifié. Le numéro du port peut être compris entre 0 et 5. Les ports d'entrées/sorties standards du ST sont:
0 = PRINTER (port parallèle imprimante).
1 = AUX (RS232).

- 2 = CONSOLE (écran-clavier).
- 3 = MIDI (Musical Instrument Digital Interface).
- 4 = KEYBOARD (clavier commande interne inexploitable).

INPUTS var\$ = INPUTS (n) [,[*] nufich]
Lit n caractères depuis le clavier ou depuis le fichier précisé et les retourne sous forme de chaîne de caractères (dans var\$) sans que rien n'apparaisse à l'écran.

Exemple :

```
10 PRINT "MOT DE PASSE"
20 PS = INPUT$(4)
30 IF PS = "ANNE" THEN 100 ELSE PRINT "FAUX":END
100 PRINT "PARFAIT"
```

INSTR INSTR ([n],A\$,B\$)
Fournit la position de la chaîne B\$ dans la chaîne A\$ en commençant à chercher à partir du nième caractère.
Par défaut, n vaut 1.

Exemple :

```
10 A$="LA TARTE AUX POMMES"
20 PRINT INSTR(2,A$,"RT")
RUN
```

INT INT(n)
Convertit une expression numérique quelconque en un nombre entier, en supprimant sa partie décimale, et sans arrondir à l'unité supérieure lorsque la partie décimale est supérieure à 0.5. Cette fonction n'est pas totalement fiable.

INTIN INTIN
Variable interne qui fournit l'adresse interne d'une table de contrôle d'entrée pour l'appel du GEM VDI par l'instruction VDISYS.

INTOUT INTOUT
Variable interne qui fournit l'adresse interne d'une table de contrôle de sortie pour l'appel du GEM VDI par l'instruction VDISYS.

LEFTS LEFTS(A\$,n)
Fournit une chaîne de caractères formée des n caractères situés à gauche de la chaîne A\$.

Exemple :

```
PRINT LEFTS("ATARI",3)
ATA
```

FONCTIONS BASIC

LEN **LEN (A\$)**
Fournit un nombre entier égal au nombre de caractères composant la chaîne de caractères précisée.

Exemple :

```
PRINT LEN ("coucou")  
6
```

LOC **LOC (n)**
Fournit un nombre représentant soit un numéro d'enregistrement, soit un nombre d'octets lus ou écrits depuis un fichier ouvert sous le numéro n.
Utilisé après **GET** ou **PUT** (fichier à accès direct), **LOC** fournit le numéro de l'enregistrement qui a été lu ou écrit le dernier. Utilisé pour un fichier séquentiel, **LOC** fournit le nombre d'octets lus ou écrits depuis l'ouverture du fichier.

LOF **LOF (n)**
Fournit le nombre d'octets présents dans le fichier ouvert sous le numéro n.

LOG **LOG(n)**
Fournit le logarithme en base e de n.

LOG10 **LOG10(n)**
Fournit le logarithme en base 10 de n.

LPOS **LPOS (n)**
Fournit la position de la tête d'imprimante dans le tampon d'imprimante. Elle est égale au nombre de caractères imprimés depuis le dernier retour-chariot. Un retour arrière vaut -1. Si des caractères de contrôle ont été imprimés, **LPOS** ne fournira pas la position réelle de la tête d'imprimante.

MID\$ **MID\$(var\$,n,m)**
Fournit une partie de la chaîne de caractères précisée. Cette partie sera longue de m caractères et commencera au nième caractère de la chaîne spécifiée.

Exemple :

```
PRINT MID$ ("MARTIN",3,2)  
RT
```

MKDS **MKDS(n)**
Convertit le nombre double précision n en une chaîne de caractères de 8 octets. Il est nécessaire de convertir les nombres ASCII en chaînes de caractères afin de pouvoir les introduire dans le tampon d'un fichier à accès direct au moyen des instructions **RSET** et **LSET**. Pour la conversion inverse, voir *CVD*.

MKIS **MKIS(n)**
 Convertit le nombre entier *n* en une chaîne de caractères de 2 octets. *Voir MKD\$. Pour la conversion inverse, voir CVI.*

MKSS **MKSS(n)**
 Convertit le nombre simple précision *n* en une chaîne de caractères de 4 octets. *Voir MKD\$. Pour la conversion inverse, voir CVS.*

OCTS **OCTS(n)**
 Convertit le nombre *n* en une chaîne de caractères représentant un nombre octal (en base 8). Le nombre *n* peut être soit décimal, soit hexadécimal. Il est converti en un nombre entier (INT) avant sa conversion en octal. Cette fonction souffre du même défaut que HEX\$.

PEEK **PEEK (adr)**
 Lit le contenu mémoire situé à l'adresse précisée. *Voir DEF SEG,* dans les instructions et commandes, pour la description des cas possibles suivant l'offset défini.

POS **POS (n)**
 Où *n* est un argument inopérant mais nécessaire. POS fournit la position courante du curseur sur l'écran ou sur l'imprimante.

PTSIN **PTSIN**
 Variable interne qui fournit l'adresse interne d'une table de contrôle d'entrée pour l'appel du GEM VDI par l'instruction VDISYS.

PTSOUT **PTSOUT**
 Variable interne qui fournit l'adresse interne d'une table de contrôle de sortie pour l'appel du GEM VDI par l'instruction VDISYS.

RIGHT\$ **RIGHT(var\$,n)**
 Fournit une chaîne de caractères composée des *n* caractères situés à droite de la chaîne précisée.

Exemple :

```
PRINT RIGHT$("MARTIN",3)
MAR
```

RND **RND ((n))**
 Fournit un nombre aléatoire compris entre 0 et 1 et choisi dans une séquence précédemment définie par RANDOMIZE. Si RND est suivi d'une expression numérique :
 - positive (*n*>0) : il fournira le nombre suivant de la séquence courante ;

FONCTIONS BASIC

- négative ($n < 0$) : il fixera une nouvelle séquence de nombres pseudo-aléatoires et en fournira le premier nombre ;
- nulle ($n = 0$) : il fournira le dernier nombre aléatoire généré.

SGN **SGN(n)**
Fournit 1 si l'expression numérique n est positive, -1 si elle est négative et 0 si elle est nulle.

SIN **SIN(n)**
Fournit la valeur du sinus de l'angle n exprimé en radians.

STR\$ **STR\$(n)**
Fournit une chaîne de caractères composée du nombre spécifié.

Exemple :

```
10 A = 123
20 B$ = STR$(A) + " COUCOU"
30 PRINT B$
RUN
123 COUCOU
```

STRING\$ **STRING\$(n,var\$)** **STRING\$(n,m)**
Fournit une chaîne de caractères composée de n fois la chaîne spécifiée ou de n fois le caractère dont le code ASCII vaut m.

Exemple :

```
PRINT STRING$(10,"*")
*****
```

SPACES **SPACES(n)**
Fournit une chaîne de caractères composée de n caractères d'espacement.

Exemple :

```
PRINT SPACES(10);"COUCOU"
COUCOU
```

SPC **SPC(n)**
Introduit des espaces dans une instruction PRINT, LPRINT ou PRINT#.

Si le nombre d'espaces spécifiés est supérieur au nombre de caractères par ligne (WIDTH) de l'écran ou de l'imprimante, le nombre d'espaces introduits sera égal à n moins le nombre de caractères par ligne ($n \text{ MOD width}$).

Exemple :

```
10 PRINT "ALPHABET"
20 PRINT "A"SPC(4)"B"SPC(2)"C"
RUN
ALPHABET
A    B    C
```

- SQR** **SQR(n)**
Fournit la racine carrée de n.
- SYSDBG** Variable interne dont l'utilisation est actuellement mal définie.
- SYSTAB** **SYSTAB**
Variable interne qui fournit l'adresse mémoire d'une table de paramètres système. Ces paramètres sont destinés à être lus et non à être modifiés. Seul SYSTAB+2 peut en principe être modifié.

<i>Déplacement</i>	<i>Fonction</i>
SYSTAB+0	Résolution graphique 1=Haute, 2=Médium 4=Basse.
SYSTAB+2	Style de la ligne de l'éditeur. Ce paramètre est compris entre 1 et 17.
SYSTAB+4	Interface fenêtre EDIT de l'AES.
SYSTAB+6	Interface fenêtre LIST de l'AES.
SYSTAB+8	Interface fenêtre de sortie de l'AES.
SYSTAB+10	Interface de la fenêtre de commande AES.
SYSTAB+12	Sémaphore ouverture fenêtre EDIT 0=fermé, 1 =ouvert.
SYSTAB+14	Sémaphore ouverture fenêtre LIST.
SYSTAB+16	Sémaphore ouverture fenêtre OUTPUT.
SYSTAB+18	Sémaphore ouverture fenêtre COMMAND.
SYSTAB+20	Adresse du tampon graphique exprimée sur 4 octets.
SYSTAB+24	Sémaphore GEM (0= ON , 1=OFF).

- TAB** **PRINT TAB(n)**
Place le curseur à la nième colonne avant d'imprimer. S'utilise avec PRINT, LPRINT et PRINT#.

Exemple :

```
10 PRINT"123456789"
20 PRINT TAB(3) "AB"
30 PRINT TAB(5) "CDE"
RUN
123456789
  AB
    CDE
```

- VAL** **VAL(var\$)**
Examine la chaîne de caractères et la convertit en un nombre réel. Si la chaîne de caractères commence par une lettre, le nombre retourné sera nul.

FONCTIONS BASIC

Exemple :

```
10 A = VAL("123AZE")
20 B = VAL("A123CDER")
30 PRINT A;" ";B
RUN
123 0
```

VARPTR **VARPTR(var)** **VARPTR(#nufich)**

Fournit l'adresse mémoire où la valeur de la variable spécifiée se trouve stockée. Cette fonction est utilisée pour permettre de passer des variables à des sous-routines écrites en langage machine.

Dans le cas où un numéro de fichier est spécifié, **VARPTR** fournit l'adresse de début du tampon permettant les entrées et sorties du fichier.

Numéro	Message
1	Non utilisé.
2	Something is wrong. Il y a une erreur quelque part.
3	RETURN statement needs matching GOSUB. Une instruction RETURN nécessite un GOSUB préalable.
4	READ statement ran out of data. Une instruction READ a été rencontrée alors qu'il n'y a plus de données à lire.
5	Function call not allowed. La fonction utilisée n'est pas autorisée.
6	Number too large. Nombre trop grand.
7	Not enough memory. Trop peu d'espace mémoire.
8	A statement or a command refers to a nonexistant line. Une instruction ou une commande fait référence à une ligne inexistante.
9	Subscript refers to element outside the array. L'indice du tableau ou du vecteur dépasse la dimension spécifiée.
10	You defined an array more than once. Une instruction DIM est exécutée sur un tableau ou un vecteur déjà dimensionné.
11	You cannot divide by zero. Vous ne pouvez pas diviser par zéro.
12	Statement is illegal in direct mode. Cette instruction ne fonctionne pas en mode direct.
13	Types of values do not match. Les valeurs introduites ne correspondent pas au type de variable décrit. Ce message apparaît par exemple lorsque l'utilisateur essaie d'affecter une chaîne de caractères à une variable numérique.
14	Non utilisé.

MESSAGES D'ERREUR

<i>Numéro</i>	<i>Message</i>
15	String cannot be over 255 characters long. La longueur d'une chaîne de caractères ne peut excéder 255 caractères.
16	Expression is too long or too complex. L'expression est trop longue ou trop compliquée.
17	CONT works only in BREAK mode. La commande CONT ne fonctionne qu'en mode BREAK.
18	Function needs prior definition with DEF FN. Cette fonction nécessite une définition préalable au moyen de DEF FN.
19	Non utilisé.
20	RESUME statement found before error routine entered. Une instruction RESUME a été rencontrée alors qu'aucune routine de traitement d'erreur n'a été utilisée.
21	Non utilisé.
22	Expression has operator with no following operand. L'expression contient un opérateur qui n'est pas suivi d'une opérande.
23	Program line too long. Ligne de programme trop longue.
24-29	Non utilisés.
30	Window number invalid. Numéro de fenêtre invalide.
31	Argument out of range. La valeur de l'argument utilisé est hors des limites.
32	Command cannot be executed from the editor. Cette commande ne peut être exécutée depuis l'éditeur.
33	Line is too complex. Cette ligne est trop complexe.
34-49	Non utilisés.

Numéro	Message
50	FIELD statement caused overflow L'instruction FIELD a provoqué un débordement.
51	Device number invalid. Numéro de périphérique illégal.
52	File number or filename invalid. Numéro ou nom de fichier invalide.
53	File not found on disk drive specified. Fichier non trouvé sur le lecteur de disques spécifié.
54	File mode is not valid. Le mode d'ouverture de fichier n'est pas bon. On a essayé d'ouvrir en mode accès direct un fichier séquentiel.
55	You cannot OPEN or KILL a file already open. Vous ne pouvez ni ouvrir ni supprimer un fichier déjà ouvert.
56	Non utilisé.
57	Disk input/output error. Erreur d'entrée/sortie disque.
58	File exists. Fichier existant.
59-60	Non utilisés.
61	Disk is full. Le disque est plein.
62	You have reached end-of-file. Vous avez atteint la fin du fichier.
63	The record number in PUT or GET is more than 32767 or zero. Le numéro d'enregistrement suivant PUT ou GET est soit supérieur à 32767, soit égal à zéro.
64	Invalid filename. Nom de fichier invalide.
65	Invalid character in program file. Caractère impropre dans un fichier de programme.

MESSAGES D'ERREUR

<i>Numéro</i>	<i>Message</i>
66	Program file has statement with no line number. Le fichier programme comporte une instruction sans numéro de ligne.
67-98	Non utilisés.
99	--Break--
100	Non utilisé.
101	Program has too many lines. Le programme comporte trop de lignes.
102	ON statement is out of range. L'instruction ON est hors limites.
103	Invalid line number. Numéro de ligne invalide.
104	A variable is required. Une variable est requise.
105	Non utilisé.
106	Line number does not exist. Numéro de ligne inexistant.
107	Number too large for an integer. Nombre trop grand pour être un entier.
108	Input data is not valid, restart input from first item. La donnée entrée est impropre, recommencez l'encodage à partir du premier paramètre.
109	Stop.
110	You have nested subroutine calls too deep. Vous avez enchaîné un trop grand nombre de sous-routines.
111	Invalid BLOAD file. Fichier ne peut être chargé par BLOAD.
112-201	Non utilisés.
202	Command not allowed here. Commande non permise ici.

Numéro	Message
203	Line number is required. Un numéro de ligne est requis.
204	FOR statement needs a NEXT or WHILE statement needs a WEND. L'instruction FOR nécessite une instruction NEXT ou l'instruction WHILE nécessite une instruction WEND.
205	NEXT statement needs a FOR or WEND statement needs a WHILE. L'instruction NEXT nécessite un FOR ou l'instruction WEND nécessite un WHILE.
206	A comma is expected. Une virgule est attendue.
207	A parenthesis is expected. Une parenthèse est attendue.
208	Option Base must be 0 ou 1. L'option Base doit être 0 ou 1. L'option base spécifie si les éléments des tableaux se comptent à partir de 0 ou à partir de 1.
209	Statement end is expected. Une instruction de fin est attendue.
210	Too many arguments in your list. Votre liste comporte trop d'arguments.
211	Non utilisé.
212	Cannot redefine variable(s). On ne peut redéfinir la (les) variable(s).
213	Function defined more than once. Fonction définie plus d'une fois.
214	You are trying to jump into a loop. Vous essayez de sauter dans une boucle.
215-220	Non utilisés.
221	System error #X, please restart. Erreur système...Redémarrage nécessaire !!!
222	Program not run. Le programme ne tourne pas.
223	Too many FOR loops. Trop de boucles FOR imbriquées.

EDITEUR

L'éditeur du Basic ST est relativement classique, il est appelable à l'aide de la commande EDIT ou du menu déroulant d'édition du bureau.

Les flèches permettent de se déplacer sans problème dans la fenêtre réservée à l'éditeur.

Les différentes fonctions d'édition sont appelables à l'aide des touches de fonction F1 à F10 ou du menu déroulant d'édition du bureau.

L'éditeur reconnaît deux types de lignes : les lignes valides qui sont présentées avec des caractères normaux et les lignes modifiées ou chargées directement dans l'éditeur et qui ne sont pas validées. Ces dernières sont appelées fantômes et sont représentées sur l'écran en caractères plus légers (sous-intensifiés)

Commandes

Touches de fonction

- F1 - Insertion d'un espace à l'emplacement du curseur.
- F2 - Effacement du caractère situé à l'emplacement du curseur.
- F3 - Insertion d'une ligne au dessus de la position courante du curseur.
- F4 - Effacement de la ligne située à l'emplacement du curseur.
- F5 - Remontée d'une page.
- F6 - Descente d'une page.
- F7 - Chargement du texte du programme courant en mode fantôme.
- F8 - Sauvegarde du texte du tampon éditeur.
- F9 - Réajustement du tampon d'édition en supprimant toutes les lignes en mode fantôme.
- F10 - Sortie de l'éditeur.

Menu déroulant

Le menu déroulant permet d'émuler les dix commandes décrites ci-dessus (touches de fonctions).

En outre, le menu déroulant permet d'activer l'éditeur et ajoute trois fonctions qui utilisent une boîte de dialogue.

La première (HELP EDIT) permet de rappeler l'utilisation des touches fonctions F1-F10.

La deuxième (GOTO LINE) permet d'afficher le texte dans la fenêtre d'édition en commençant à une ligne précisée dans la boîte de dialogue.

La troisième (DELETE LINES) permet d'effacer un ensemble de lignes comprises entre deux numéros précisés dans la boîte de dialogue.

Remarque : Les lignes d'instructions de grande taille présentent un problème certain. Pour éviter des désagréments, nous vous conseillons de taper un blanc en début de chaque ligne écran (une ligne de grande taille est fatalement constituée de plusieurs lignes d'écran).

BASIC ET LANGAGE MACHINE

Le Basic est un langage facile à utiliser et très efficace pour les calculs mathématiques et les programmes de gestion. Mais, lorsqu'une exécution ultra-rapide ou une économie de mémoire est nécessaire, on doit s'adresser au processeur dans sa langue maternelle : le langage machine.

A l'exception de programmes très particuliers comme les jeux d'arcades ou les logiciels intégrés, il est rarement pratique d'écrire un programme complet en langage machine, cette écriture étant longue et fastidieuse.

La meilleure approche consiste à réaliser le programme en Basic et à programmer les sous-routines trop longues au point de vue temps en langage machine.

L'utilisation de sous-programmes en langage machine dans un programme Basic nécessite quelques précautions :

- interdire au Basic et à ses tables de rentrer en conflit avec lui au point de vue de l'emplacement ;
- introduire (charger) le programme en langage machine dans la mémoire ; nous analyserons dans la suite de cette section les différentes façons de procéder ;
- définir l'adresse de départ de la routine ; en général, c'est le premier octet de la routine, mais cette règle n'est pas absolue ;
- exécuter la routine ; c'est le but de l'instruction ou de l'ensemble DEFUSRn et USRn.

Instructions CALL, DEFUSR et USR

DEFUSR et USR : syntaxe : DEFUSRn = adr : X = USRn(Y)

Où n représente un nombre compris entre 0 et 9. On peut donc définir dix appels de fonction. La fonction USR lance effectivement l'appel. L'argument Y permet de passer une et une seule valeur à la fonction. La variable X permet de récupérer une valeur en sortie de la routine.

Cette fonction particulièrement limitée est surtout fournie pour permettre une compatibilité avec le Basic Microsoft. De toutes façons, dans la version actuelle du Basic, ces instructions ne marchent pas. Il faut donc leur préférer l'instruction CALL.

CALL : syntaxe : CALL varadr[(liste de paramètres)]

Où varadr est une variable qui contient l'adresse de départ de la sous-routine et la liste de paramètres est une zone optionnelle qui peut être composée de différents paramètres séparés par une virgule.

Cette instruction permet l'appel à un sous-programme externe au Basic et écrit en langage machine. Ce sous-programme doit avoir son point d'entrée (le premier octet exécutable) à l'adresse spécifiée dans la variable varadr.

Fonctionnement

Après l'appel de l'instruction CALL, le registre compteur programme du processeur est positionné à l'adresse indiquée dans la variable varadr.

Chaque paramètre est constitué de huit octets, quel que soit son type. Lors du passage de l'appel, le registre pointeur de pile utilisateur (A7) pointe sur deux paramètres. Le premier est constitué de deux octets et fournit le nombre de paramètres passés à la routine (de 0 à 65535 paramètres). Le second est un pointeur d'adresses sur quatre octets qui fournit l'adresse de la table de $n * 8$ octets qui contient la valeur des n paramètres.

Méthodes de chargement à l'intérieur du Basic

Méthode DATA et POKE

C'est la méthode la plus simple. Ce n'est pas la plus rapide ni la plus économique.

Procédure : il suffit de mettre les valeurs à introduire en mémoire (en hexadécimal ou en décimal) dans les lignes de DATA, ensuite, on les lit et on les envoie en mémoire par des POKE successifs, et ce, au moyen d'une boucle FOR - NEXT.

Hélas, il faut s'assurer que l'adresse choisie soit libre et n'interfère pas avec le Basic ou avec le DOS. En effet, l'instruction CLEAR ne permet pas, comme dans les autres Basic, de protéger la mémoire. Cette méthode est donc à déconseiller et n'est donnée que comme exemple.

Remarque : l'exemple figurant ci-après ne réalise rien de particulier mais il ne plante pas le système (78 et 117 produisent un RTS).

Exemple Ecrire les octets 42,72,63,60,78,117 à l'adresse 500000.

```
10 FOR I=500000 TO 500005
20 READ A
30 POKE I,A
40 NEXT I
50 DATA 42,72,63,60,78,117
```

Le même exemple, mais avec des valeurs hexadécimales :

```
10 FOR I=500000 TO 500005
20 READ AS
30 POKE I,VAL("&H"+AS)
40 NEXT I
50 DATA 2A,48,3F,3C,4E,75
```

BASIC ET LANGAGE MACHINE

Méthode de la chaîne de caractères

Cette méthode, très en vogue dans les premiers Basic Microsoft, s'applique assez aisément au Basic ATARI.

Inconvénients de la méthode

- Routine limitée à 255 caractères ;
- le programme doit être indépendant de l'adresse d'implantation.

Description de la méthode

Cette méthode est proche de la précédente mais au lieu de déposer les octets en mémoire à l'aide de POKE, il suffit de les déposer dans une chaîne de caractères au moyen d'une instruction du type :

```
VAR$ = VARS + CHR$(n)
```

où n représente la valeur de l'octet à mémoriser.

L'exemple précédent devient donc :

```
10 N$=""
20 FOR I=1 TO 6
30 READ A
40 N$=N$+CHR$(A)
50 NEXT I
60 DATA 42,72,63,60,78,117
```

L'appel de la routine est réalisée par la suite d'instructions suivante :

```
70 AD=0
80 AD=PEEK(VARPTR(N$)+2)
90 CALL AD
```

Remarque : la fonction PEEK fournit l'adresse de la routine sur 32 bits (4 octets). L'initialisation de la variable AD à 0 est indispensable afin de ne pas créer une nouvelle variable (en l'occurrence AD) après la lecture du VARPTR. En effet, la variable créée pourrait modifier l'emplacement de la chaîne de caractères.

Méthode de la variable tableau

On peut charger une routine en langage machine dans une variable tableau entière.

Avantages

- Pas de protection nécessaire ;
- le transfert d'arguments est très facile.

Inconvénient

- Le programme doit être indépendant de l'adresse.

Description de la méthode

- Définir la variable tableau comme variable entière ;
- diviser le nombre d'octets de la routine par 2 et prendre le plus grand entier - 1 ;
- dimensionner la variable avec la valeur ainsi trouvée ;
- calculer la valeur de chaque élément du tableau en utilisant la formule suivante :

$$256 * \text{octet}(n) + \text{octet}(n+1)$$

ou, plus facilement, en stockant 2 octets sous la forme d'un entier hexadécimal (4 digits) ;

- établir les égalités d'éléments ;
- définir le point d'entrée au moment du CALL par :

AD = 0

AD = VARPTR(N(0)) : CALL AD

où N est le nom de la variable tableau.

L'exemple précédent devient :

```

10 DEFINT A
20 DIM A(2):REM (6 octets)
30 A(0)=&H12A48
40 A(1)=&H3F3C
50 A(2)=&H4E75
60 Z=0
70 Z=VARPTR(A(0))
80 CALL Z
    
```

Méthode du BLOAD

La commande **BLOAD** permet de charger directement un programme en langage machine en mémoire. Ce procédé nécessite un temps de chargement plus court que la méthode des **DATA**, il permet de charger des sous-routines préalablement sauveées par un **BSAVE** ou créées par l'éditeur de liens avec un en-tête **BLOAD**.

Avantages

- Permet de charger des grandes routines ;
- chargement aisé.

Inconvénient

- Nécessité d'utiliser un Assembleur.

FORMAT INTERNE DES VALEURS ET VARPTR

La fonction **VARPTR** constitue un des plus merveilleux outils du Basic. Elle permet d'atteindre l'adresse de stockage des valeurs assignées aux variables ainsi que différentes informations sur leur contenu.

A l'aide des adresses obtenues par la fonction **VARPTR**, de l'instruction **POKE** et de la fonction **PEEK**, on peut effectuer une foule d'opérations très utiles.

L'utilisation principale de la fonction **VARPTR** est certainement de retrouver des informations concernant les chaînes de caractères.

Lorsqu'on écrit : `10 A$="COUCOU"`, le système d'exploitation de l'interpréteur Basic doit sauvegarder la valeur affectée à **A\$** (en l'occurrence **COUCOU**) quelque part dans la mémoire. Lorsque, quelques lignes plus bas, on écrit : `50 PRINT A$`, le système devra être capable de retrouver **COUCOU**.

Pour effectuer cette opération, le Basic utilise la **TV** (Table des Variables). Chaque fois qu'il rencontre une nouvelle variable, il l'ajoute à la **TV**.

Chaque fois que le Basic rencontre une nouvelle variable, il fouille la **TV** pour voir si cette variable a déjà été affectée. Si ce n'est pas le cas, il l'ajoute à la liste.

Remarque : le temps pris par le système pour retrouver une variable est un facteur influençant très fort la vitesse d'exécution des programmes. Il est possible d'améliorer de façon notable la vitesse d'exécution d'un programme en définissant, au début du programme, une liste des variables les plus souvent utilisées.

Les variables simples sont définies la première fois qu'on leur attribue une valeur, les variables tableau sont définies lors de l'instruction **DIM**.

Utilisation de **VARPTR**

L'instruction `X=VARPTR(A$)` fournira une valeur **X**, adresse où des informations sur **A\$** pourront être trouvées.

La variable sur laquelle on demande des renseignements peut être une variable entière, une variable simple ou double précision, une variable alphanumérique ou un élément d'une variable tableau de n'importe quel type.

Exemple :

`X=VARPTR(A$(2))` est parfaitement licite.

L'utilisation que l'on peut faire de l'adresse contenue dans **X** à l'issue de l'instruction est fonction du type de la variable.

FORMAT INTERNE DES VALEURS ET VARPTR

La valeur contenue dans X est une adresse (4 octets) comprise entre 0 et la taille de la mémoire.

Contenu de AD dans la fonction AD=VARPTR(var)

Le contenu de l'adresse fournie par la fonction VARPTR varie en fonction du type de la variable.

Si AD est l'adresse fournie par la fonction VARPTR, alors, quel que soit le type de la variable :

AD-1 contient le type de la variable
AD et AD+n contiennent la valeur de la variable.

Rappel : si la variable est du type :

- entier : n=1
- simple précision : n=3
- double précision : n=7

AD+n+1 contient le nombre de caractères que comprend le nom de la variable.

AD+n+2 à AD+n+X contiennent le nom de la variable.

Remarque importante : lors de vos investigations, pour que le PEEK porte sur un seul octet, il faut écrire :

```
DEF SEG = 1
```

L'offset est alors de 1. Il ne faut bien entendu pas oublier d'ajouter cet offset à toutes les adresses.

Variable entière

Une variable entière est mémorisée sur 16 bits en binaire signé. Le bit le plus significatif (B15) est le bit de signe.

AD contient les 8 bits les plus significatifs de la variable (B0 à B7).

AD+1 contient les 8 bits les moins significatifs de la variable (B8 à B15).

Exemple :

a) Si la variable entière vaut 12345 décimal (3039 en hexa), alors :

- l'adresse AD contient 48 (30H) et AD+1 contient 57 (39H) ;
- et $57 + 256 * 48 = 12345$.

FORMAT INTERNE DES VALEURS ET VARPTR

```
10 DEF SEG = 1
20 DEFINI B
30 B=12345
40 AD=0
50 AD=VARPTR(B)
60 PRINT PEEK(AD-1);PEEK(AD)
70 PRINT 256*PEEK(AD-1)+PEEK(AD)
```

b) Si la variable vaut -12345 (CFC711) :

- l'adresse AD contient 199 (C711) et AD+1 contient 207 (CF11) ;
- et $199 + 256 * 207 = 65536 = -12345$.

rem: Attention à l'offset!

Variable simple précision

Les quatre octets de AD à AD+3 contiennent la valeur de la variable.

La façon de représenter les variables simple et double précisions est particulièrement complexe.

- 0 se code par 4 octets à 0 ;
- le bit B7 de AD+3 indique le signe 0=positif, 1=négatif ;
- AD+3 - 4111 représente la puissance de 2 pour obtenir le nombre inférieur le plus proche du nombre recherché ; autrement dit, le nombre contenu dans var est m alors AD+3 contient le logarithme en base 2 de m augmenté de 4111 ;
- les autres bits doivent être lus en partant de B6 de AD jusqu'à B0 de AD+2. Ils indiquent la fraction du nombre représenté par AD+3 qui doit être ajoutée pour obtenir le nombre désiré. B6 de AD vaut 1/2, B5 vaut 1/4 ... B0 vaut 1/64, et ainsi de suite avec les bits de AD+1 et AD+2. Le bit B7 de AD est toujours à 1.

Exemples

a) Si AD+3 contient 48H et AD contient E0H, alors le nombre est :

- 2 exposant 48-41 soit 2 exposant 7 = 128
- AD = E0 = 11100000 c'est à dire +1/2 +1/4 de 128, soit : $128 * 64 + 32 = 224$.

b) Représentons le nombre -12345.

- Négatif : B7 de AD+3 = 1 ;
- LOG base 2 de 12345 = 13 ou 0D PRINT LOG(12345)/LOG(2) ;
- 2 exposant 13 = 8192 ;
- AD+3 contient donc 4111 + 0DH + 80H (signe) = CEH ;
- exprimez le nombre en binaire : 11000000111001 ;

FORMAT INTERNE DES VALEURS ET VARPTR

- mettez à 0 le premier digit significatif : 01000000111001 ;
- complétez par des 0 à droite pour obtenir un multiple de 8 bits.
Coupez par tranche de 8 bits :
01000000 11100100.
- vous obtenez les valeurs respectives de AD, AD+1 et AD+2 ;
- mettez le bit 7 de AD à 1: 11000000 11100100
AD = C0, AD+1 = E4 et AD+2 = 0.

Variable double précision

Les variables double précision se codent sur huit octets, le format est identique à celui des variables simple précision, mais la mantisse porte sur quatre octets supplémentaires.

Variable tableau

La partie donnée d'une variable tableau est mémorisée de façon identique à celle d'une variable simple. Les éléments sont simplement disposés les uns après les autres en commençant par l'indice 0.

Variable alphanumérique

Deux cas sont possibles :

- si AD-2 = 0 alors AD-1 contient la longueur de la chaîne et AD à AD+n contient la chaîne proprement dite ;
- si AD-2 = 10H (16), alors AD-1 contient la longueur de la chaîne et AD à AD+3 contient l'adresse réelle de la chaîne.

Dans les deux cas, AD-3 contient 6 (type alphanumérique).

Le Basic a été conçu pour être facilement interfacé avec le GEM (Graphic Environment Manager).

GEM est un logiciel créé par Digital Research pour permettre l'élaboration de programmes graphiques indépendants du matériel utilisé. Ainsi, on retrouve le GEM sur les IBM/PC et compatibles.

Le GEM est divisé en deux parties. L'AES qui prend en charge la gestion des fenêtres, des icônes (dessins) et des menus ; le VDI qui prend en charge l'affichage du texte et les tracés graphiques élémentaires.

Le VDI est très facile et presque complètement interfaçable avec le Basic, bien que certaines routines n'offrent aucun intérêt vis-à-vis des fonctions Basic qui les utilisent déjà.

L'AES est bien plus complexe à interfacer avec le Basic et ses utilisations sont très limitées.

La liste complète des routines de l'AES et du VDI ainsi que tous les paramètres utilisés par ces deux sous-ensembles se trouvent dans le tome 2 des "CLEFS POUR ST".

Interfaçage avec le VDI

Pour interfacer le Basic et le VDI, les concepteurs du Basic ont créé l'instruction **VDISYS** qui permet l'appel du VDI.

Pour pouvoir utiliser **VDISYS**, il faut, au préalable, donner une série de renseignements au système, à travers des adresses internes, qui sont fournies par une série de variables internes spécialisées.

Ces variables sont au nombre de cinq. Elles ont pour nom : **INTIN**, **INTOUT**, **PTSIN**, **PTSOUT** et **CONTRL**.

PTSIN et **INTIN** sont les tables de paramètres d'entrée (fournis au VDI lors de son appel). **PTSOUT** et **INTOUT** sont les tables de paramètres de sortie (fournis par le VDI à l'issue de son appel).

Utilisation

La préparation de l'appel se fait par une série de **POKE** aux adresses fournies par les variables internes.

Remarque : la plupart des valeurs sont codées sur 16 bits, il suffit donc de faire un seul **POKE** si une valeur 0 a été affectée à l'instruction **DEF SEG**.

BASIC ET GEM

- 1 - CONTRL + 0 doit contenir le numéro d'appel de la fonction VDI (POKE CONTRL,n où n est le numéro d'appel de la fonction).
- 2 - CONTRL + 2 doit contenir le nombre de paramètres contenus dans PTSIN (éventuellement 0).
- 3 - CONTRL + 4 doit contenir le nombre de paramètres contenus dans PTSOUT.
- 4 - CONTRL + 6 doit contenir le nombre de paramètres contenus dans INTIN.
- 5 - CONTRL + 8 doit contenir le nombre de paramètres contenus dans INTOUT.
- 6 - CONTRL + 10 doit contenir le numéro secondaire éventuel d'appel de fonction.
- 7 - CONTRL + 12 doit contenir le sémaphore de périphérique (1 à 10).
- 8 - Les différentes tables INTIN, INTOUT, PTSIN et PTSOUT doivent contenir les paramètres nécessaires à l'appel de la fonction.
- 9 - L'appel est simplement réalisé par l'instruction VDISYS sans argument.

Exemple : Modification de la taille des lignes.

Pour modifier la taille des lignes tracées, il suffit d'appeler la fonction VDI numéro 16 (voir tome 2).

Il faut mettre la taille du tracé dans le premier paramètre de PTSIN et 0 dans le second paramètre de PTSIN. INTIN ne contient rien.

Pour positionner une ligne d'épaisseur 10, il suffit d'écrire :

POKE CONTRL,16	Numéro de fonction VDI
POKE CONTRL+2,1	paramètre dans PTSIN
POKE CONTRL+6,0	Rien dans INTIN
POKE PTSIN,10	Épaisseur 10
POKE PTSIN+2,0	0 ensuite
VDISYS	Appel de fonction

Interfaçage avec l'AES

Les concepteurs du Basic ont aussi fourni une instruction d'interfaçage entre le Basic et l'AES. Bien que l'utilisation en soit moins évidente, elle n'en est pas pour autant complètement dénuée d'intérêt.

L'instruction est appelée GEMSYS, elle peut être suivie directement du numéro de la fonction AES à appeler.

Une seule variable de paramétrage est utilisée. Cette variable appelée GB fournit l'adresse d'une table de six adresses de quatre octets chacune qui correspondent aux adresses des tables CONTROL, GLOBAL, INTIN, INTOUT, ADDRIN et ADDRROUT de l'AES.

En principe, la table CONTROL et la table GLOBAL ne doivent pas être modifiées. Les autres tables s'utilisent comme les tables du VDI.

Remarque : les valeurs des tables INTIN et INTOUT sont des entiers 16 bits (2 octets). Les valeurs de ADDRIN et ADDRROUT sont des adresses codées sur 32 bits (4 octets).

GENERALITES

Le langage Logo est livré en standard avec tous les systèmes ATARI ST. Il fait donc un peu partie du logiciel de base. C'est pourquoi nous lui consacrons un chapitre complet.

Le langage Logo est considéré à tort comme un langage d'apprentissage de l'informatique destiné aux jeunes enfants.

Cette réputation est due à l'existence du système de déplacement de la tortue qui permet une visualisation directe et, par là, une matérialisation complète des principes de base. En vérité, cette particularité intéressante n'est que le sommet d'un iceberg informatique. Elle permet d'entrer sans difficulté dans un langage riche et puissant dont les caractéristiques intrinsèques dépassent de loin la plupart des langages Basic.

Les amateurs de Basic, qui refusent d'aborder Logo, risquent de passer à côté de concepts aussi importants que la RECURSIVITE, la définition de PROPRIETE ou les notions de FONCTION, VARIABLES GLOBALES et VARIABLES LOCALES.

Enfin, pour terminer, signalons qu'avec un peu de programmation, le Logo ouvre la voie aux systèmes d'intelligence artificielle, et peut être considéré comme le début d'un petit langage Prolog ou un succédané du langage Lisp.

COMMANDES ET MOTS CLES (PRIMITIVES)

- Opérateur de multiplication.
?2*3
- / Opérateur de division.
?2/3
- + Opérateur d'addition.
?2+3
- Opérateur de soustraction ou de moins unaire.
?2-3
- ^ Opérateur d'élevation à puissance.
?2^3
- < > = Un de ces signes placé entre deux nombres fournira TRUE ou FALSE suivant que la vérité de l'expression se vérifie ou non.
?45 < 67
TRUE
?56 = 80
FALSE
- <> >< a <> b a >< b
Fournit TRUE si a n'est pas égal à b, FALSE si a=b.
? <> 3 3
FALSE
? >< 4 5
TRUE
- >= <= => =<
Plus petit ou égal (<= et =<) et plus grand ou égal (>= et =>).
Fournissent TRUE ou FALSE suivant le cas.
?>= 5 7
FALSE
?=> 60 6
TRUE
?<= 2 5
TRUE
?=< 8 2
FALSE
?>= 7 7
TRUE
- () Les parenthèses servent de liaison d'opérateurs lorsque plusieurs arguments sont utilisés.
(+ 2 3 4) équivaut à 2 + 3 + 4
- || Les crochets servent de délimiteur pour les listes et le coeur d'une instruction REPEAT.

COMMANDES ET MOTS CLES (PRIMITIVES)

- .APV** Propriété caractéristique des variables globales.
?MAKE "A 9
?GPROP "A ".APV
9
- .BUR** Propriété des applications qui, lorsqu'elle vaut TRUE, signifie que l'application est cachée. Une application constitue un groupe de procédures réunies sous un même nom qui est le nom de l'application. Lorsqu'une application est cachée, elle n'apparaît pas lors des commandes de visualisation du contenu de l'espace de travail Logo (POPS...). Voir *BURY* et *UNBURY*.
- .CONTENTS** Primitive système qui fournit une liste composée de tous les mots définis dans l'espace de travail du Logo.
- .DEF** Propriété caractéristique des procédures définies.
- .DEPOSIT** .DEPOSIT n1 n2
Dépose la valeur n2 dans l'emplacement mémoire n1. Peut être comparé au POKE du Basic. n2 doit être compris entre 0 et 255. ATTENTION : Si n1 dépasse 4194303 (4 mégaoctets), le système se "plante" irrémédiablement. Une réinitialisation générale est alors nécessaire.
- .ENL** Propriété système désignant la fin d'une ligne d'une procédure qui est rompue par un retour-chariot et des espaces.
- .EXAMINE** .EXAMINE n
Fournit le contenu de l'emplacement mémoire n. Peut être comparé au PEEK du Basic. ATTENTION : Si n dépasse 4194303, le système se plante irrémédiablement. Une réinitialisation générale est alors nécessaire.
- .FMT** Propriété système désignant le début d'une ligne d'une procédure qui est rompue par un retour-chariot et des espaces.
- .FPT** Propriété système identifiant un format de remplissage défini par l'utilisateur.
- .LPT** Propriété système identifiant un format de ligne défini par l'utilisateur.
- .PAK** Propriété système d'un objet identifiant le nom de l'application à laquelle cet objet appartient.
- .PKG** Propriété système signifiant que l'objet est un nom d'application.
- .PRM** Propriété système identifiant une primitive.

COMMANDES ET MOTS CLES (PRIMITIVES)

- .REM** Propriété système identifiant un commentaire.
- .REPLACE** `.REPLACE n :varlist objet`
Remplace le nième élément de la liste par l'objet spécifié. La liste doit être représentée par un nom de variable.
- `?MAKE "MACHIN [A Z E R T Y]`
`?REPLACE 3 :MACHIN [12 34]`
`?MACHIN`
`[A Z [12 34] R T Y]`
- .REPTAIL** `.REPTAIL n varlist objet`
Remplace à partir du nième élément les éléments de la liste par l'objet spécifié. La liste doit être représentée par un nom de variable.
- `?MAKE "MACHIN [A Z E R T Y]`
`?REPTAIL 4 [4 5 6]`
`?MACHIN`
`[A Z E 4 5 6]`
- .SPC** Propriété système identifiant un espace.
- ABS** `ABS n`
Fournit la valeur absolue d'un nombre.
`?ABS -3`
3
- AND** `AND expression expression ...`
Permet d'effectuer des opérations ET logiques.
Fournit TRUE si les toutes les expressions concernées sont vraies et FALSE sinon.
- ARC** `ARC [X Y rayon position1 position2]`
Permet de dessiner un arc de cercle dont le centre est situé au point de coordonnées (X,Y) et les extrémités sont fournies par position1 et position2. Ces dernières sont exprimées en degrés d'angle sachant que la position 0 correspond au nord de l'image (midi) et que la rotation s'effectue dans le sens horlogique.
Cette commande ne modifie pas la position de la tortue.
- ARCTAN** `ARCTAN n`
Fournit l'arctangente en degrés du nombre spécifié.
- ASCII** `ASCII "mot`
Fournit la valeur ASCII de la première lettre du mot spécifié.
`?ASCII "abbc`
97

COMMANDES ET MOTS CLES (PRIMITIVES)

- BACK** BK n
BK Instruction ayant pour effet de faire reculer la tortue de n unités.
- BOX** BOX [X Y hauteur longueur]
Construit un rectangle de hauteur et longueur spécifiées dont le coin inférieur gauche sera le point de coordonnées (X,Y).
- BURY** BURY "application / [liste d'applications]
Cache l'application (PACKAGE) ou la liste d'applications pour les commandes qui traitent l'espace de travail (EDALL, EDNS, ERALL, GLIST, POPS, SAVE...). Voir aussi .BUR.
- BUTFIRST** BF objet
BF Ampute l'objet spécifié de son premier élément.
BF "azer
zer
BF [mn op qrs]
[op qrs]
BF BF [ab cd ef]
[ef]
- BUTLAST** BL objet
BL Ampute l'objet précisé de son dernier élément.
BL "azerty
azert
BL 1234
123
- BYE** Permet de quitter le Logo.
- CATCH** CATCH "variable [procédure]
Lance la procédure spécifiée jusqu'à la rencontre d'un THROW suivi du nom de la variable. Exemple :
?TO TEST
>CATCH "bof [suite]
>PR [coucou]
>END
TEST defined
?TO SUITE
>IF KEYP [THROW "bof]
>suite
>END
SUITE defined
?TEST
- enfoncer une touche -
coucou
Dans la procédure TEST, l'instruction CATCH passe le contrôle à la procédure SUITE. Le THROW, suivi du nom de la variable de l'envoyeur ("bof) repasse le contrôle à la

COMMANDES ET MOTS CLES (PRIMITIVES)

procédure TEST. Les deux instructions CATCH et THROW peuvent être comparées aux GOSUB et RETURN du Basic.

- CHANGEF** CHANGEF nouveaunom anciennom
Permet de changer le nom d'un fichier.
- CHAR** CHAR n
Fournit le caractère ASCII correspondant au nombre spécifié. Elle permet de disposer de certains caractères graphiques.
- CIRCLE** CIRCLE [X Y rayon]
Permet de dessiner un cercle de rayon spécifié et dont le centre correspond au point de coordonnées (X,Y).
- CLEAN** Permet d'effacer l'écran graphique sans modifier la position courante de la tortue.
- CLEARSCREEN** Efface l'écran graphique et replace la tortue à sa position
CS de départ (centre de l'écran, tête tournée vers le haut).
- CLEARTEXT** Efface l'écran texte et y replace le curseur en haut à gauche.
- CO** Permet de continuer l'exécution d'un programme interrompu par CTRL Z ou PAUSE. Après un CTRL Z, il est possible d'interroger la machine sur l'état de la procédure en cours d'exécution.
- COPYDEF** COPYDEF nouveaunom anciennom
Effectue une copie de la définition d'une procédure sous un autre nom.
- COPYOFF** Inhibe l'écho imprimante.
- COPYON** Active l'écho imprimante.
- COS** COS n
Donne le cosinus de l'angle à exprimer en degrés.
- COUNT** COUNT objet
Fournit le nombre d'éléments composant l'objet spécifiée.
COUNT [ab cd ef]
3
COUNT "coucou"
6
- DEFINE** DEFINE "nom [liste de définition]
Définit une procédure en mode direct.

COMMANDES ET MOTS CLES (PRIMITIVES)

DEFINEDP	DEFINEDP objet Fournit la valeur TRUE si l'objet spécifié identifie une procédure définie. Dans le cas contraire, il fournit la valeur FALSE.
DEGREES	DEGREES n Fournit le nombre de degrés correspondant à n radians.
DIR	DIR <"fichier > Fournit la liste de tous les nom de fichiers Logo présents sur le disque (.LOG) par défaut ou sur le disque spécifié. Cette commande accepte un nom de fichier ambigu. Pour plus de précisions à ce sujet, voyez le chapitre traitant des commandes DOS.
EDALL	EDALL <"application / [liste d'applications]> Charge dans l'éditeur toutes les variables et toutes les procédures existantes dans l'espace de travail.
EDF	EDF "nom Charge à partir du disque le fichier dont le nom est spécifié. Le chargement se fait dans l'espace EDITEUR et celui-ci est activé.
EDIT ED	ED <"procédure / "variable / [liste de procédures] / [liste de variables]> Active l'éditeur et y charge la (les) procédure(s) ou la (les) variable(s). Pour le fonctionnement de l'éditeur, reportez-vous à la table des codes de contrôle.
EDNS	EDNS <"application / [liste d'applications]> Charge toutes les variables de l'espace de travail dans l'éditeur texte.
EDPS	EDPS <"application / [liste d'applications]> Charge l'application (package) ou la liste d'applications dans l'espace d'édition. Si rien n'est spécifié, cette primitive charge toutes les procédures contenues dans l'espace de travail dans le tampon d'édition.
ELLIPSE	ELLIPSE [X Y rayonX rayonY] Dessine une ellipse dont le centre se trouve au point de coordonnées (X,Y), dont le rayon vertical vaut rayonX et le rayon horizontal, rayonY.
EMPTYP	EMPTYP objet Fournit la valeur système correspondant à l'objet donné. TRUE indiquera un objet vide, FALSE un objet non vide. ?EMPTYP "coucou

COMMANDES ET MOTS CLES (PRIMITIVES)

	FALSE ?EMPTYP [] TRUE
END	Constitue la dernière ligne d'une procédure. END signale la fin de la procédure et ramène au signal de sollicitation logo (?).
EQUALP	EQUALP objet1 objet2 Fournit TRUE dans le cas où les deux objets sont deux nombres, deux mots ou deux listes identiques. Dans le cas contraire, cette fonction fournit FALSE. ?EQUALP [ad 45 po] [ad 45 po] TRUE
ERALL	ERALL <"application / [liste d'applications]> Efface toutes les procédures et variables non cachées de l'espace de travail ou de(s) application(s) spécifiée(s).
ERASE ER	ER "procédure / [liste de procédures] Supprime de l'espace de travail Logo la (les) procédure(s) non cachée(s) désignée(s).
ERASEFILE	ERASEFILE "fichier Efface le fichier disque spécifié.
ERN	ERN "variable / [liste de variables] Supprime de l'espace de travail Logo la (les) variable(s) non cachée(s) spécifiée(s).
ERNS	ERNS <"application / [liste d'applications]> Supprime toutes les variables non cachées de l'espace de travail ou de l'(des) application(s) non cachée(s) spécifiée(s).
ERPS	ERPS <"application / [liste d'applications]> Supprime toutes les procédures non cachées de l'espace de travail ou de l'(des) application(s) non cachée(s) spécifiées.
ERRACT	Valeur système qui, lorsqu'elle vaut TRUE, a pour effet de provoquer une PAUSE lorsqu'une erreur se produit.
ERROR	Fournit une liste dont les éléments décrivent l'erreur la plus récente. ?ERROR [29 [Not enough input to CIRCLE] CIRCLE [CIRCLE] [] []]
EXP	EXP n Fournit le nombre n exposant e. ?EXP 1 2.71828

COMMANDES ET MOTS CLES (PRIMITIVES)

FALSE	Valeur système.
FENCE	Commande empêchant la tortue de sortir de l'écran graphique. La ligne : ?FENCE FD 500 produira le message suivant : turtle out of bounds.
FILL	Colorie la zone limitée par des traits de la couleur courante du crayon au moyen du graphisme prévu dans FILLATTR et ce, lorsque GFILL possède la valeur TRUE. Un ?MAKE "GFILL "TRUE la lui fournira.
FILLATTR	Fournit les attributs de style, d'index et de couleur du motif de remplissage courant. Voir SETFILL.
FIRST	FIRST objet Fournit le premier élément de l'objet précisé. ?FIRST "abc a
FOLLOW	FOLLOW "procédure "procédure Réorganise l'espace de travail pour que le premier nom de procédure précisé soit directement suivi par le second. FOLLOW ne modifie pas l'ordre des procédures dans une application.
FORWARD FD	FD n Permet de faire avancer la tortue de n unités.
FPUT	FPUT objet1 objet2 Construit un nouvel objet en utilisant le premier objet comme premier élément du second objet. ?FPUT "ne "nni nenni ?FPUT [ne] [nni] [[ne] nni] ?FPUT "ne [nni] [ne nni] ?FPUT [ne] "nni FPUT doesn't like nni as input On ne peut pas ajouter une liste à un mot.
GETTEXT	Fournit le numéro de la fonte de caractère. Voir SETTEXT. La fonte par défaut est la fonte 0.
GFILL	Valeur système qui, lorsqu'elle vaut TRUE, a pour effet le remplissage des dessins graphiques au moyen des attributs courants de remplissage.

COMMANDES ET MOTS CLES (PRIMITIVES)

- GLIST** GLIST "propriété <"application / [liste d'applications]> Fournit la liste de tous les objets présents dans l'espace de travail Logo ou dans la (les) application(s) spécifiée(s) et possédant la propriété précisée. Ainsi :
?GLIST ".DEF
fournira la liste de toutes les procédures définies dans l'espace de travail Logo.
- GO** GO "mot
Utilisé au sein d'une procédure pour passer le contrôle à la ligne portant le label désigné.
?TO ESSAI
>LABEL "ici
>PR [ceci est un bete programme]
>GO "ici
>END
- GPROP** GPROP "nom "propriété
Fournira le contenu de "nom suivant la propriété désignée (.APV = variable, .DEF = procédure).
?GPROP "A ".DEF
fournira le contenu de la procédure A.
?GPROP "A ".APV
fournira le contenu de la variable A.
- GRAPHICS** Contient la liste des propriétés définies par l'utilisateur pour les motifs de ligne et de remplissage.
- HEADING** Fournit la direction dans laquelle la tête de la tortue est tournée, exprimée en degrés d'angle. La position 0 indique la tête tournée vers le haut de l'écran.
- HIDETURTLE** Rend la tortue invisible.
- HT** Cette commande permet de clarifier les dessins.
- HOME** Place la tortue au point de coordonnées (0,0), tête tournée vers le haut.
- IF** IF proposition [action 1] [action 2]
Si la proposition est vraie, l'action 1 sera effectuée.
Si elle est fausse, l'action 2 sera effectuée. Il n'est pas obligatoire de spécifier une action 2. En effet, dans ce cas et si la proposition est fausse, rien ne sera effectué.
?IF 4=5 [PR [ca alors]] [PR [faut pas rêver]]
faut pas rêver.
- IFFALSE** IFFALSE [action]
Exécute l'action spécifiée dans le cas où l'expression TEST la plus récente est fausse. Pour l'exemple, voyez les explications concernant TEST.

COMMANDES ET MOTS CLES (PRIMITIVES)

IFTRUE	IFTRUE [action] Exécute l'action spécifiée dans le cas où l'expression TEST la plus récente est vraie. Pour l'exemple, voyez les explications concernant TEST.
INT	INT n Fournit la valeur entière du nombre n. C'est-à-dire tout ce qui se trouve devant le point. ?INT -1.2 -1
ITEM	ITEM n objet Fournit le nième élément de l'objet spécifié. ?ITEM 3 [oui non zut flut] zut ?ITEM 3 1/2 5 Les opérations arithmétiques sont effectuées avant la fonction ITEM. 1/2 valant 0.5, son troisième élément est donc 5.
KEYP	Commande qui teste le clavier et fournit une valeur de vérité TRUE si une touche a été enfoncée ou FALSE si aucune touche n'a été enfoncée. ?IF KEYP = "TRUE [action 1] [action 2] effectuera l'action 1 si une touche a été enfoncée et l'action 2 sinon.
LABEL	LABEL "mot Permet de disposer une étiquette à un endroit déterminé d'une procédure. Voir l'instruction GO pour l'exemple.
LAST	LAST objet Fournit le dernier élément de l'objet précisé. ?LAST [av cf ion de] de
LEFT LT	LT n Fait tourner la tortue de n degrés d'angle vers la gauche.
LINEATTR	Fournit les différents attributs correspondant à la ligne courante.
LIST	LIST objet objet <objet...> Place les objets précisés dans une liste. ?LIST [ab c] "d [[ab c] d]
LISTP	LISTP objet Fournit la valeur TRUE si l'objet précisé est une liste, la valeur FALSE dans les autres cas.

COMMANDES ET MOTS CLES (PRIMITIVES)

LOAD	LOAD "fichier Permet de charger un fichier disque dans l'espace de travail Logo. Voir <i>SAVE</i> .
LOADPIC	LOADPIC "fichier Charge le fichier écran graphique spécifié.
LOCAL	LOCAL "variable Permet de rendre le contenu de certaines variables valable uniquement à l'intérieur de la procédure comportant cette instruction. ?TO ESSAI >LOCAL "C >MAKE "C "coucou >PR :C >END ESSAI DEFINED ?MAKE "C "bonjour ?ESSAI coucou ?:C bonjour
LOG	LOG n Fournit le logarithme en base e du nombre n. ?LOG 2 0.693147
LOG10	LOG10 n Fournit le logarithme en base 10 du nombre n. ?LOG10 100 2
LOWERCASE LC	LC "MOT Convertit le mot spécifié en minuscules. ?LC "COUCOU coucou
LPUT	LPUT objet1 objet2 Ajoute objet1 à la suite de objet2 pour former un nouvel objet. ?LPUT 7 [4 5 6] [4 5 6 7]
MAKE	MAKE "variable "contenu Permet d'assigner une valeur à une variable. ?MAKE "D 1/2 ?:D 0.5

COMMANDES ET MOTS CLES (PRIMITIVES)

- MEMBERP** MEMBERP objet1 objet2
Fournit la valeur TRUE si objet1 est un élément d'objet2. Sinon, il fournit la valeur FALSE.
?MEMBERP "LOGO "LOGOPEDE
TRUE
- MOUSE** Fournit une liste contenant la position courante de la souris et ce, sous la forme :
[X Y A B C]
X et Y représentent les coordonnées du point où elle se trouve ; A et B représentent l'état des boutons de la souris, ils valent TRUE lorsqu'on les presse ; C vaut TRUE si le pointeur de la souris se trouve en dehors du champ graphique.
- NAME** NAME objet "variable
Donne à l'objet précisé le nom de variable choisi.
?NAME 2 "D
?:D
2
- NAMEP** NAMEP objet
Fournit la valeur TRUE si l'objet est une variable définie, FALSE sinon.
- NODES** Fournit un nombre exprimant la taille de l'espace de travail Logo non encore utilisé à l'instant t. Un noeud (node) vaut 4 bytes. Il faut d'abord savoir que toute action effectuée consomme de l'espace de travail puisqu'elle y est enregistrée.
Essayez : ?REPEAT 2000 [PR NODES]
Les nombres affichés seront de plus en plus petits car les instructions successives PR NODES consomment de la place. Lorsque cet espace disponible approchera du nombre 100, vous constaterez un temps d'arrêt. Ensuite, le nombre affiché sera de nouveau élevé. En fait, durant cet arrêt, le Logo a réorganisé son espace de travail afin d'y libérer de la place car il considèrerait en avoir trop peu. Voir *RECYCLE*.
- NOFORMAT** Supprime le formatage des procédures de l'espace de travail.
- NOT** Inverse la valeur de vérité.
?NOT (5=9)
TRUE
- NOTRACE** Inhibe la fonction TRACE. Cette dernière affiche les noms des procédures en cours d'exécution.
- NOWATCH** Inhibe la fonction WATCH. Celle-ci affiche les noms d'expressions en cours d'exécution.

COMMANDES ET MOTS CLES (PRIMITIVES)

- NUMBERP** NUMBERP objet
Fournit TRUE si l'objet spécifié est un nombre et FALSE sinon.
?NUMBERP "35
TRUE
?NUMBERP "VINGT
FALSE
- OR** OR expression1 expression2
Fournit FALSE si toutes les expressions proposées sont fausses. Fournit TRUE sinon.
?OR (7=9) (6=6)
TRUE
- OUTPUT**
OP Sort de la procédure et retourne à la procédure appelante en lui passant la valeur désignée.
>IF 3 = 1+2 [OUTPUT "TRUE]
TRUE
- PACKAGE** PACKAGE "application "nom / [liste de noms]
Affecte à l'application désignée le nom ou la liste de noms spécifiée.
?PACKAGE "JEUX [belote whist bridge]
- PALETTE**
PAL PAL n
Fournit une liste de trois nombres compris entre 0 et 1000 et représentant les intensités des couleurs primaires (R=rouge, G=vert, B=bleu) composant la couleur numéro n. Voir *SETPAL* pour modifier les couleurs. n représente le numéro de couleur et sa valeur se situe entre 0 et 15 dans le cas du mode 16 couleurs.
?PAL 1
[1000 1000 1000]
- PATH** Fournit le nom du drive courant par défaut (A: ou :)
?PATH
A:
- PAUSE** Utilisé dans une procédure pour en stopper momentanément l'exécution. Un CO sera nécessaire pour la redémarrer.
- PENDOWN**
PD Abaisse le crayon graphique. La tortue tracera des lignes en se déplaçant. Voir *PU*.
- PENERASE**
PE Remplace la couleur du crayon de la tortue par la même couleur que le fond. Cela permettra d'effacer des traits en repassant dessus.

COMMANDES ET MOTS CLES (PRIMITIVES)

- PENREVERSE** Modifie la couleur du crayon en sa couleur logique inverse.
PX Tout ce qui a été dessiné auparavant changera ainsi de couleur.
- PENUP** Relève le crayon de la tortue ; ainsi, ses déplacements ne dessineront plus de traits. Pour rabaisser le crayon, il faudra utiliser PD.
- PI** Fournit la valeur du nombre PI.
?PI
3,1416
- PIECE** **PIECE** n1 n2 objet
Fournit les parties de l'objet situées entre les emplacements n1 et n2. Cette fonction est à rapprocher de la fonction Basic MID\$.
?PIECE 3 7 [a b c d e f g h i j k l]
c d e f g
- PKGALL** **PKGALL** "application
Introduit toutes les procédures et variables ne faisant pas encore partie d'une application dans l'application spécifiée.
?PKGALL "AUTRES
- PLIST** **PLIST** "mot
Fournit la liste des propriétés attribuées au mot désigné.
?MAKE "A 2
?PLIST "A
[APV 2]
?TO ADD :SUJ :REL :OBJ
>PPROP :SUJ :REL :OBJ
>PPROP :OBJ WORD :REL "_est :SUJ
>END
ADD DEFINED
?ADD "Anne "maman_de "Yves
?PLIST "Anne
[maman_de Yves]
?PLIST "Yves
[maman_de _est Anne]
- PO** **PO** "procédure / "variable / [liste de procédures] /
[liste de variables]
Affiche à l'écran le contenu de la (des) procédure(s) et/ou de la (des) variable(s) désignée(s).

COMMANDES ET MOTS CLES (PRIMITIVES)

- POALL** Affiche tous les contenus des variables et des procédures contenues dans l'espace de travail.
- POCALL** POCALL "procédure
Fournit les noms des procédures appelées dans la procédure désignée.
- POLY** POLY [X1 Y1 X2 Y2 Xn Yn]
Dessine un polygone reliant les différents points de coordonnées (X,Y).
- PONS** PONS <"application / [liste d'applications]>
Affiche les noms et valeurs de toutes les variables contenues dans l'espace de travail ou dans l'(les) application(s) spécifiée(s).
- POPKG** POPKG <"application / [liste d'applications]>
Fournit les noms et contenus de toutes les applications présentes dans l'espace de travail ou dans l'(les) application(s) spécifiée(s).
- POPS** POPS <"application / [liste d'applications]>
Affiche toutes les procédures contenues dans l'espace de travail ou dans l'(les) application(s) désignée(s).
- POREF** POREF "procédure / [liste de procédures]
Fournit les noms des procédures qui appellent la (les) procédure(s) désignée(s).
- POS** Fournit une liste de deux nombres représentant les coordonnées (X,Y) de la position de la tortue.
- POTL** Fournit les noms des procédures TOP LEVEL. Ces dernières ne sont jamais appelées par d'autres procédures présentes dans l'espace de travail.
- POTS** POTS <"application / [liste d'applications]>
Affiche à l'écran la première ligne (celle qui commence par TO) de toutes les procédures présentes dans l'espace de travail ou dans l' (les) application(s) spécifiée(s).
- PPROP** PPROP "nom "propriété
Permet d'attribuer une propriété nouvelle à un nom.
?PPROP "toto ".APV 12
a le même effet que :
?MAKE "toto 12
De même :
PPROP "W ".DEF [[]contenu de procédure]
pourra être utilisé au lieu de :

COMMANDES ET MOTS CLES (PRIMITIVES)

	<pre>?TO W >contenu de procédure >END</pre>
PPS	<pre>PPS <"application / [liste d'applications]> Fournit toutes les paires de propriétés non système de tous les objets contenus dans l'espace de travail ou dans l'(les) application(s) spécifiée(s).</pre>
PRIMITIVEP	<pre>PRIMITIVEP objet Fournit TRUE si l'objet désigné est un nom de primitive, fournit FALSE sinon.</pre>
PRINT PR	<pre>PR "mot / [liste] Affiche à l'écran le mot ou la liste suivi d'un retour-chariot. Comparez avec SHOW et TYPE. Des parenthèses seront nécessaires si on désire faire suivre PRINT de plusieurs objets. ?PR "a PR "z a z ?PR [adc] adc ?(PR "a "b "c) a b c</pre>
PROCLIST	<pre>Fournit une liste dont le contenu est constitué de tous les noms de procédures définies.</pre>
PRODUCT	<pre>PRODUCT n1 n2 n3 ... nn Fournit un nombre égal au produit des nombres désignés. ?PRODUCT 3 2 5 30</pre>
QUOTIENT	<pre>QUOTIENT n1 n2 Fournit le quotient entier de la division de n1 par n2. ?QUOTIENT 24 6 4</pre>
RADIANS	<pre>RADIANS n Convertit en radians le nombre n, en supposant que n exprime un nombre de degrés. ?RADIANS 90 1.570796</pre>
RANDOM	<pre>RANDOM n Fournit un nombre entier aléatoire compris entre 0 et n-1. n doit être compris entre 32767 et -32768.</pre>

COMMANDES ET MOTS CLES (PRIMITIVES)

- READCHAR** Attend un caractère frappé au clavier et l'affiche.
RC Cette fonction est très utile dans les procédures où l'on désire que l'utilisateur fournisse une réponse.
 ?IF RC = "o [PR "oui] [PR "non]
- READLIST** Fournit une liste contenant une ligne suivie d'un
RL retour-chariot frappée à l'écran ou lue depuis un fichier.
 ?READLIST
 A B C
 [A B C]
- READQUOTE** Fournit un mot formé d'une ligne suivie d'un retour-chariot
RQ frappée à l'écran ou lue depuis un fichier.
 ?RQ
 1 2 3
 1 2 3
- RECYCLE** Cette commande ordonne au Logo de réorganiser son espace
 de travail afin d'y libérer de la place. Cette opération porte le
 nom de "garbage collection" (collecte d'ordures). Elle est
 effectuée automatiquement lorsque l'espace de travail devient
 trop petit (100 nodes).
- REDEFP** Variable système contenant FALSE en standard et ayant pour
 rôle d'interdire la redéfinition des primitives en tant que
 procédure. Si sa valeur est TRUE, il sera alors possible de
 redéfinir les primitives. Naturellement, cette variable est à
 utiliser prudemment. En effet, lorsqu'une primitive est
 redéfinie, elle ne fait plus ce pour quoi elle a été prévue.
 Sauvez votre espace de travail et essayez :
 ?TO FD
 FD is a primitive
 ?MAKE "REDEFP "TRUE
 ?TO FD
 >END
 FD defined
 ?FD 100
 Plus rien désormais ne se passera car, ayant été redéfinie,
 l'expression FD n'est plus une primitive.
- REMAINDER** **REMAINDER n1 n2**
 Fournit le reste de la division entière de n1 par n2. A
 rapprocher de la fonction Basic MOD.
- REMPROP** **REMPROP "nom "nature**
 Supprime les propriétés de nature spécifiée attribuées au nom
 désigné.
 ?MAKE "x "coucou
 ?PLIST "x
 [.APV coucou]

COMMANDES ET MOTS CLES (PRIMITIVES)

	?:x coucou ?REMPROP "x ".APV ?PLIST "x [] ?:x x has no value.
REPEAT	REPEAT n [action] Effectue n fois l'action spécifiée. ?REPEAT 4 [FD 50 RT 90] dessinera un carré.
RERANDOM	Réinitialise le générateur de nombres aléatoires, donc les primitives RANDOM et SHUFFLE.
RIGIIT RT	RT n Fait tourner la tortue de n degrés d'angle vers la droite.
ROUND	ROUND n Arrondit le nombre spécifié. Le passage à l'unité directement supérieure se fait à partir de la demi-unité (.5).
RUN	RUN [liste d'instructions] Effectue la liste d'instructions désignées. Revient à écrire repeat l.
SAVE	SAVE "fichier <"application / [liste d'applications]> Permet de sauvegarder tout ou partie de l'espace de travail Logo dans le but de le réutiliser par la suite. Un LOAD permettra de le recharger ultérieurement.
SAVEPIC	SAVEPIC "fichier Sauve l'écran graphique sur disque sous le nom spécifié.
SCREENFACTS SF	Fournit une liste dont les éléments décrivent l'arrangement courant de l'écran. 1- Numéro de la couleur de fond. Modifiable par SETBG. 2- Etat des bords de l'écran. Vaut WINDOW, WRAP ou FENCE. 3- Rapport d'écran. Modifiable par SETSCRUNCH. 4- ZOOM = facteur d'agrandissement des objets visibles sur l'écran graphique, modifiable par SETZOOM. 5- Point central de l'écran graphique, modifiable par SETPAN. ?SETBG 4 ?FENCE SETSCRUNCH 2 SETZOOM 2 ?SETPAN [60 60] ?SCREENFACTS [4 FENCE 2 2 60 60]

COMMANDES ET MOTS CLES (PRIMITIVES)

- SENTENCE SE** SE élément1 élément2 ...
Fournit une liste composée des éléments spécifiés.
Voyez également LIST.
?SE [ab c] "d
[ab c d]
?SE [[a]] [b]
[[a] b]
- SETBG** SETBG n
Positionne la couleur du fond. Un CLEARSCREEN doit suivre cette instruction pour que la nouvelle couleur de fond s'affiche.
- SETFILL** SETFILL [style index couleur]
Permet de fournir au système les attributs de remplissage des figures fermées.
- 1- **Style**: 5 styles sont possibles, numérotés de 0 à 4.
0 : incolore
1 : noir
2 : petits points
3 : ligné
4 : définissable par l'utilisateur. En standard, représente le dessin Atari.
La formule suivante permet de définir le motif désiré :
?PPROP "GRAPHICS ".FPT [liste de 16 val] Les 16 valeurs sont chacune comprises entre 0 et 65535 (16 bits) de manière à définir une matrice de 16 points sur 16 points. Chaque nombre définit une ligne de 16 points. Pour en déterminer un, il suffit d'additionner les poids des bits correspondants aux points à allumer.
- 2- **Index** : vaut 0 ou 1 et n'a d'influence que lorsque le style 2 ou 3 est sélectionné.
Style 2 sélectionné : index peut prendre toutes les valeurs entre 0 et 24. 0 et 1 donnant le même effet, 24 types de petits points peuvent être sélectionnés. Style 3 est sélectionné : index peut prendre toutes les valeurs entre 0 et 12. 0 et 1 donnant le même effet, 12 types de "lignés" peuvent être sélectionnés.
- 3- **Couleur** : nombre compris entre 0 et 15 permettant de sélectionner une des 16 couleurs définies en mode 16 couleurs. Les modes 4 couleurs et noir/blanc n'acceptent respectivement que des valeurs comprises entre 0 et 3, et 0 et 1.
- SETHREADING** SETH n
SETH Oriente la tortue dans la direction indiquée par n, en degrés d'angle. *Voir aussi TF.*
?SETH 90

COMMANDES ET MOTS CLES (PRIMITIVES)

SETLINE

SETLINE [style épaisseur couleur]
Permet de sélectionner le type de lignes que tracera la tortue.

1- **Style** : nombre compris entre 0 et 7:

- 0 et 1 : ligne continue. _____
- 2 : traits -----
- 3 : points
.....
- 4 : trait-point -.-.-.-.-
- 5 : traits -----
- 6 : trait-point-point -.-.-.-.-
- 7 : ligne continue _____

Le style 7 peut être redéfini par l'utilisateur au moyen de la formule :

?GPROP "GRAPHICS ".LPT n

où n est un nombre compris entre 0 et 65535. Ce qui permet de définir un modèle de 16 points. n doit valoir la somme des poids des bits correspondant aux points à allumer.

2- **Épaisseur** : nombre compris entre 0 et 39 permettant de sélectionner 40 épaisseurs différentes. 0 est la plus fine, 39 la plus grosse. Au-delà de l'épaisseur 3, les pointillés deviennent des lignes.

3- **Couleur** : nombre compris entre 0 et 15 et permettant de sélectionner une des 16 couleurs du système. En 4 couleurs et en noir/blanc, ce nombre sera compris entre 0 et 3 ou 0 et 1.

SETPAL

SETPAL n [R G B]

Fournit les intensités respectives des couleurs primaires R(rouge) G(vert) B(bleu) pour la couleur numéro n. En mode 16 couleurs, n est compris entre 0 et 15. Chaque intensité est un nombre compris entre 0 et 1000. En fait, 8 niveaux d'intensités sont perceptibles. Il suffira de multiplier le niveau désiré (de 0 à 7) par 143. En 16 couleurs, $8 \times 8 \times 8 = 512$ couleurs différentes sont donc sélectionnables.

?PAL 1
[0 0 0]
?SETPAL 1 [1000 0 0]
?PAL 1
[1000 0 0]

SETPAN

SETPAN [X Y]

Positionne le point central de la fenêtre graphique. Par défaut, il vaut (0,0).

SETPATH

SETPATH ":A

Permet de sélectionner le disque courant. :A et :B sont possibles.

COMMANDES ET MOTS CLES (PRIMITIVES)

?PATII
:B
SETPATH "A
?PATH
:A

SETPC SETPC n
Détermine la couleur dans laquelle la tortue dessinera.

SETPEN SETPEN [Px n]
Établit le mode de la tortue (PD ou PU) et le numéro n de la couleur qu'elle devra utiliser.
?SETPEN [PU 1]

SETPOS SETPOS [X Y]
Déplace la tortue vers le point de coordonnées (X,Y) tout en conservant sa direction.
?SETPOS [55 40]

SETSCRUNCH SETSCRUNCH n
Donne le nombre n spécifié comme rapport de la taille horizontale de l'écran sur la taille verticale.

SETTEXT SETTEXT n
Permet de sélectionner le type de caractère frappé. n est compris entre 0 et 63.
8 bits permettent cette sélection (notez que B7 et B6 ne sont pas utilisés, pour cette raison, ils valent toujours 0) :

B7	B6	B5	B4	B3	B2	B1	B0	Poids	caractères
0	0	0	0	0	0	0	0	0	normal
0	0	0	0	0	0	0	1	1	gras
0	0	0	0	0	0	1	0	2	pointillé
0	0	0	0	0	1	0	0	4	italique
0	0	0	0	1	0	0	0	8	souligné
0	0	0	1	0	0	0	0	16	inverse
0	0	1	0	0	0	0	0	32	normal

Les combinaisons sont possibles en additionnant les poids des bits à mettre à 1. Par exemple, on désire du gras pointillé : n vaudra 1+2=3.

SETX SETX n
Déplace la tortue horizontalement vers le point de coordonnée horizontale n.
?SETX -30

SETY SETY n

COMMANDES ET MOTS CLES (PRIMITIVES)

Déplace la tortue verticalement vers le point de coordonnée verticale n.

?SETY 40

SETZOOM SETZOOM n
Permet de définir un nouveau facteur d'agrandissement de l'écran graphique sans pour cela altérer le dessin en cours.
?SETZOOM 3

SHOW SHOW objet
Affiche l'objet désigné tel qu'il se présente en le faisant suivre d'un retour-chariot. Voir aussi PR et TYPE.
?SHOW [A] SHOW [B]
[A]
[B]

SHOWTURTLE Rend la tortue visible sur l'écran graphique.
ST

SHUFFLE SHUFFLE [liste]
Fournit une liste contenant les éléments spécifiés mélangés de façon aléatoire.
SHUFFLE [a b c d e]
[c e a b d]

SIN SIN n
Fournit la valeur de l'angle n exprimé en degrés.
?SIN 90
1

SORT SORT [liste]
Fournit une liste contenant les éléments spécifiés triés dans un ordre ascendant (du plus petit au plus grand).
?SORT [G J 3 U 8 F Z 1 9]
[1 3 8 9 F G J U Z]

SQRT SQRT n
Fournit la racine carrée du nombre n.
?SQRT 36
6

STOP Utilisé dans une procédure pour en arrêter l'exécution. Dans le cas où STOP se trouve dans une procédure appelée par une autre, STOP renvoie le contrôle à la procédure appelante. Dans l'autre cas, STOP renvoie le contrôle au TOPLEVEL (?). Un THROW *TOPLEVEL, par contre, stoppera l'exécution de toute procédure et reviendra au signal TOPLEVEL (?) de sollicitation Logo.

SUM SUM n1 n2
Fournit la somme des nombres (n1 + n2 + ...) spécifiés.

COMMANDES ET MOTS CLES (PRIMITIVES)

- ?SUM 4 7 8
19
- TAN** TAN n
Fournit la valeur de la tangente de l'angle n exprimé en degrés.
?TAN 45
1
- TEST** TEST expression
Retient le résultat de l'expression (TRUE ou FALSE) pour les prochains IFTRUE et IFFALSE.
?TO ESSAI
>TEST 1=RANDOM 2
>IF 1=RANDOM 100 [PRINT [raté] STOP]
>IFTRUE [TYPE "oui]
>IFFALSE [TYPE "non]
>END
- TEXT** TEXT "procédure
Fournit le contenu de la procédure spécifiée.
- THING** THING "variable
Fournit la valeur de la variable spécifiée.
?MAKE a "coucou
?THING "a
coucou
- THROW** THROW "variable
Exécute la ligne identifiée par la variable spécifiée dans une expression CATCH prévue. Voir CATCH pour l'exemple. Un THROW "TOPLEVEL permet de quitter la procédure en cours et de remonter au niveau du signal de sollicitation Logo (?) sans repasser par les procédures appelantes.
- TO** TO procédures <variables>
Signale qu'on va définir une procédure.
?TO rectangle :longueur :largeur
>...
- TOPLEVEL** Signal de sollicitation de l'interpréteur Logo (?). Niveau pour lequel aucune procédure n'est activée. Voir aussi THROW.
- TOWARDS** TOWARDS [X Y]
Pointe la tortue en direction du point de coordonnées (X,Y).
- TRACE** Active la fonction trace (affichage du nom de la procédure en cours d'exécution).

COMMANDES ET MOTS CLES (PRIMITIVES)

- TRUE** Valeur système.
- TURTLEFACTS** Fournit une liste composée de six éléments permettant de définir l'état courant de la tortue :
- TF**
- 1 et 2 : coordonnées horizontale et verticale de la position courante de la tortue.
 - 3 : direction de la tortue en degrés d'angle.
 - 4 : état du crayon (PD ou PU)
 - 5 : couleur utilisée par la tortue.
 - 6 : visibilité de la tortue. Vaut THROW si elle est visible et FALSE sinon.
- TURTLETEXT** TT "mot / [liste]
Permet d'afficher des caractères texte dans l'écran graphique.
?REPEAT 4[FD 100 TT "xx RT 90]
- TYPE** TYPE "mot / [liste]
Affiche à l'écran l'objet désigné sans effectuer de retour-chariot. Voir aussi PR et SHOW.
?TYPE "a TYPE "b
ab
- UNBURY** UNBURY "application
Rend l'application spécifiée active dans l'espace de travail. Voir BURY et .BUR.
- UPPERCASE** UPPERCASE "mot
UC Convertit le mot spécifié en majuscules.
?UC "bonjour
BONJOUR
- WATCH** WATCH <"procédure \[liste de procédures]>
Affiche le nom de chaque expression en cours d'exécution. WATCH permet d'utiliser l'éditeur et l'interpréteur.
- WHERE** Fournit l'emplacement de l'objet1 dans l'objet2 de la plus récente expression MEMBERP.
?MEMBERP "s [a f s r t y u i]
TRUE
?WHERE
3
- WINDOW** Spécifie que le mode écran fonctionne comme une fenêtre. La tortue peut dès lors quitter les bords de l'écran sans que rien ne se passe. Voir aussi WRAP et FENCE.
?FENCE FD 300
?WINDOW FD 300
- WORD** WORD "mot1 "mot2 <"mot3...>
Fournit un seul mot formé par la concaténation des mots spécifiés.

COMMANDES ET MOTS CLES (PRIMITIVES)

- ?WORD "cou "cou
coucou
- WORDP** WORDP objet
Fournit TRUE lorsque l'objet spécifié est un mot ou un nombre, FALSE s'il est une liste.
?WORDP 1/2
TRUE
?WORDP [a d s e]
FALSE
- WRAP** Après cette commande, lorsque la tortue dépasse les bords de l'écran, elle revient par le côté opposé.
?WRAP FD 700
Voir WINDOW et FENCE.
- XCOR** Fournit la valeur de la coordonnée horizontale de la position courante de la tortue.
- YCOR** Fournit la valeur de la coordonnée verticale de la position courante de la tortue.

COMMANDES LOGO UTILISANT LES CARACTERES DE CONTROLE

Ces commandes sont utilisées en vue d'effectuer des mouvements de curseur et d'écran.

Pour les utiliser, il faut enfoncer la touche CTRL et la lettre correspondante ensemble.

Les commandes suivies de * ne fonctionnent que pour l'éditeur de texte.

<i>Caractère</i>	<i>Rôle</i>
Ctrl-A	Amène le curseur au début de la ligne.
Ctrl-B	Recule le curseur d'une position vers la gauche.
Ctrl-C *	Quitte l'éditeur de texte et introduit dans l'espace de travail Logo les modifications effectuées au cours de l'édition.
Ctrl-E	Amène le curseur en fin de ligne.
Ctrl-F	Avance le curseur d'une position vers la droite.
Ctrl-G	En dehors de l'éditeur de texte, Ctrl-G a pour effet de stopper l'exécution de la procédure en cours (break). En éditeur de texte, Ctrl-G permet de quitter l'éditeur de texte sans enregistrer les modifications survenues au cours de l'édition.
Ctrl-H	Efface le caractère situé à gauche du curseur.
Ctrl-I	Déplace le curseur à la tabulation suivante (colonne 5, 9, 13, ...) et insère 4 espaces dans la ligne courante.
Ctrl-K	Efface tous les caractères situés à la droite du curseur. Les caractères effacés sont stockés dans un buffer et peuvent être récupérés au moyen de Ctrl-Y.
Ctrl-L *	A l'intérieur de l'éditeur texte, Ctrl-L réarrange l'écran de manière à positionner la ligne indiquée par le curseur au centre de la fenêtre. Si le curseur est situé à moins de 12 lignes du début du buffer, l'éditeur texte fait réapparaître le début du texte en haut de fenêtre.
Ctrl-M	Génère un retour-chariot et introduit les informations dans l'ordinateur.
Ctrl-N	Déplace le curseur à la ligne suivante de l'éditeur de texte.

COMMANDES LOGO UTILISANT LES CARACTERES DE CONTROLE

<i>Caractère</i>	<i>Rôle</i>
Ctrl-O *	Ouvre une nouvelle ligne dans l'éditeur de texte. Cette commande est équivalente à Enter suivi de Ctrl-B.
Ctrl-P	Fait remonter le curseur d'une ligne.
Ctrl-Q	Génère un caractère dièse # qui permet au Logo de traiter les caractères délimiteurs comme des caractères texte. Les caractères délimiteurs sont : [] () " ; ; = < > + / ^ .
Ctrl-R *	Positionne le curseur au début du buffer de l'éditeur de texte.
Ctrl-S *	Permet de délimiter des blocs.
Ctrl-T *	Copie les blocs précédemment délimités par des Ctrl-S.
Ctrl-U *	Remonte d'une page dans l'éditeur de texte, si toutefois cette page existe.
Ctrl-V *	Descend à la page suivante dans l'éditeur de texte, si toutefois cette page existe.
Ctrl-W *	Efface le bloc précédemment délimité par des Ctrl-S.
Ctrl-X *	Positionne le curseur à la fin du buffer de l'éditeur de texte.
Ctrl-Y	Fait réapparaître la plus récente ligne stockée dans le buffer par Enter, Ctrl-K ou bloc.
Ctrl-Z	Intrompt l'exécution de la procédure en cours et affiche un signal de pause permettant des interactions. Un CO permettra de continuer l'exécution, un THROW "TOPLEVEL ramènera le signal ? de sollicitation Logo et un STOP remontera d'un niveau (à la procédure appelante).

- Number too big.
Nombre trop grand.
- No file selected.
Aucun fichier n'est sélectionné.
- (mot) is a primitive.
Le mot entre parenthèses est une primitive. On a essayé de la redéfinir alors que REDEFN vaut FALSE.
- Can't find LABEL (mot).
Ne trouve pas le mot précisé.
- Can't (symbol) from the editor.
Ce mot n'est pas utilisable depuis l'éditeur.
- I'm having trouble with the disk.
Signale un problème disque.
- Disk is full.
Le disque est rempli.
- Can't divide by zero.
Impossible de diviser par zéro.
- File is not open.
Le fichier n'est pas ouvert.
- File already exists.
On a essayé de créer un fichier qui existe déjà.
- File not found.
Ne trouve pas le fichier demandé.
- Can't find CATCH for (mot).
Un THROW ne trouve pas le CATCH qui lui correspond.
- I'm out of space.
L'espace de travail Logo est plein.
- (mot) is not true nor false.
Le mot indiqué ne contient ni TRUE ni FALSE dans ses propriétés.
- Not enough inputs to (procédure).
Lors de l'appel d'une procédure, on a oublié de spécifier les valeurs des variables.
- Too few items in (liste).
Trop peu d'éléments dans la liste indiquée.

MESSAGES D'ERREUR LOGO

- Turtle out of bounds.
La tortue a quitté l'écran.
- I don't know how to (mot).
Je ne comprends pas le mot spécifié.
- (mot) has no value.
Aucune valeur n'est affectée à la variable indiquée.
-) without (.
On a oublié de refermer la parenthèse.
- I don't know what to do with (mot).
Le Logo ne sait ce qu'il doit faire du mot spécifié.
- Primitive not implemented.
La primitive demandée existe dans le système mais n'est pas implantée. On ne peut donc pas s'en servir.
- Disk is write-protected.
Le disque est protégé en écriture. On ne peut écrire dessus.
- (procédure) doesn't like (mot) as input.
La procédure indiquée n'accepte pas le mot spécifié en entrée.
- The word is too long.
Un mot trop long a été encodé.
- I don't have enough buffer space.
Le buffer ne dispose pas de suffisamment de place pour l'opération demandée.
- IF wants []'s around instruction list.
IF nécessite des crochets [] autour de sa liste d'instructions.
- (mot) isn't a parameter.
Le mot proposé n'est pas un paramètre.
- I can't (mot) while loading.
Logo ne peut exécuter le mot spécifié pendant le chargement du disque.
- The file is write-protected.
Le fichier est protégé en écriture.
- I can't find the disk drive.
Logo ne trouve pas le lecteur de disque demandé.
- No PAN with FENCE or WRAP.
On ne peut utiliser PAN avec FENCE ou WRAP.
- Error messages for pictures files.
Messages d'erreur pour fichiers graphiques.

TABLE DES ADRESSES DES MOTS CLES LOGO CLASSES PAR MOTS CLES

Les valeurs des adresses sont fournies en décimal.

<i>Mot clé</i>	<i>Adresse</i>
ABS	5354
AND	2496
ARC	26192
ARCTAN	2812
ASCII	6366
BACK (BK)	16526
BOX	26000
BURY	14216
BUTFIRST (BF)	6400
BUTLAST (BL)	6972
CATCH	9656
CHANGEF	20470
CHAR	6454
CIRCLE	26254
CLEAN	16574
CLEARSCREEN (CS)	16600
CLEARTEXT (CT)	17808
CO	2618
COPYDEF	13496
COPYOFF	19010
COPYON	18928
COS	2860
COUNT	6510
DEFINE	11964
DEFINEDP	13420
DEGREES	5382
DIR	20880
EDALL	13800
EDF	19724
EDIT (ED)	13976
EDNS	13888
EDPS	13932
ELLIPSE	26628
EMPTYP	6946
EQUALP	7058
ERALL	15036
ERASE (ER)	15150
ERASEFILE	19628
ERN	15174
ERNS	15208
ERPS	15228
ERROR	9834

<i>Mot clé</i>	<i>Adresse</i>
EXP	5656
FENCE	34588
FILL	27388
FILLATTR	26464
FIRST	7084
FOLLOW	13694
FORWARD (FD)	16642
FPUT	7144
GETTEXT	26954
GLIST	11726
GO	10004
GPROP	11312
HEADING	18102
HIDETURTLE (HT)	16918
HOME	16944
IF	2652
IFFALSE (IFF)	5302
IFTRUE (IFT)	5328
INT	2920
ITEM	7252
KEYP	17760
LABEL	9984
LAST	7374
LEFT (LT)	16968
LINEATTR	26830
LIST	7472
LISTP	7502
LOAD	19948
LOADPIC	25588
LOCAL	10932
LOG	5686
LOG10	5746
LOWERCASE (LC)	9014
LPUT	7542
MAKE	10878
MEMBERP	7862
MOUSE	29214
NAME	11096
NAMEP	11004
NODES	10446
NOFORMAT	11922
NOT	2452

LANGUAGE LOGO

TABLE DES ADRESSES DES MOTS CLES LOGO CLASSES PAR MOTS CLES

<i>Mot clé</i>	<i>Adresse</i>
NOTRACE	10568
NOWATCH	10746
NUMBERP	8194
OR	2512
OUTPUT (OP)	2398
PACKAGE	15288
PALETTE (PAL)	28740
PATH	20700
PAUSE	2226
PENDOWN (PD)	33212
PENERASE (PE)	33252
PENREVERSE (PX)	33276
PENUP (PU)	33238
PI	5458
PIECE	8752
PKGALL	15360
PLIST	11360
PO	14778
POALL	14856
POCALL	12278
POLY	26518
PONS	14928
POPKG	14374
POPS	14982
POREF	13020
POS	18056
POTL	12654
POTS	14886
PPROP	11392
PPS	11468
PRIMITIVEP	13462
PRINT (PR)	18522
PROCLIST	13444
PRODUCT	3042
QUOTIENT	3354
RADIANS	5420
RANDOM	3214
READCHAR (RC)	18800
READLIST (RL)	18870
READQUOTE (RQ)	18722
RECYCLE	63296
REMAINDER	3414
REMPROP	11678
REPEAT	2336

<i>Mot clé</i>	<i>Adresse</i>
RERANDOM	3504
RIGHT (RT)	17004
ROUND	3520
RUN	2432
SAVE	20118
SAVEPIC	25612
SCREENFACTS (SF)	16802
SENTENCE (SE)	82548
SETBG	17030
SETFILL	26380
SETHEADING (SETH)	17074
SETLINE	26750
SETPAL	28632
SETPAN	16396
SETPATH	20584
SETPC	17100
SETPEN	17136
SETPOS	17352
SETSCRUNCH	17822
SETTEXT	26886
SETX	17400
SETY	17448
SETZOOM	16370
SHOW	18674
SHOWTURTLE (ST)	17496
SHUFFLE	4734
SIN	2890
SORT	4924
SQRT	5596
STOP	2414
SUM	3616
TAN	5506
TEST	5272
TEXT	12218
THING	11070
THROW	9852
TOWARDS	17522
TRACE	10542
TURTLEFACTS (TF)	16680
TURTLETEXT (TT)	28812
TYPE	18538
UNBURY	14264
UPPERCASE	8992
WATCH	10660

TABLE DES ADRESSES DES MOTS CLES LOGO CLASSES PAR MOTS CLES

<i>Mot clé</i>	<i>Adresse</i>
WHERE	8972
WINDOW	34482
WORD	8512
WORDP	8482
WRAP	17782
XCOR	18012
YCOR	18034
+	3616
-	3634
•	3042

<i>Mot clé</i>	<i>Adresse</i>
/	4324
^	5832
<	4394
>	4408
=	7058
.CONTENTS	10492
.DEPOSIT	18204
.EXAMINE	18124
.REPLACE	7706
.REPTAIL	7772

TABLE DES ADRESSES DES MOTS CLES LOGO CLASSES PAR ADRESSES

Adresse	Mot clé
2226	PAUSE
2336	REPEAT
2398	OUTPUT (OP)
2414	STOP
2432	RUN
2452	NOT
2496	AND
2512	OR
2618	CO
2652	IF
2812	ARCTAN
2860	COS
2890	SIN
2920	INT
3042	PRODUCT *
3214	RANDOM
3354	QUOTIENT
3414	REMAINDER
3504	RERANDOM
3520	ROUND
3616	SUM +
3634	-
4324	/
4394	<
4408	>
4734	SHUFFLE
4924	SORT
5272	TEST
5302	IFFALSE (IFF)
5328	IFTRUE (IFT)
5354	ABS
5382	DEGREES
5420	RADIANS
5458	PI
5506	TAN
5596	SQRT
5656	EXP
5686	LOG
5746	LOG10
5832	^
6366	ASCII
6400	BUTFIRST (BF)
6454	CHAR
6510	COUNT

Adresse	Mot clé
6946	EMPTYP
6972	BUTLAST (BL)
7058	EQUALP =
7084	FIRST
7144	FPUT
7252	ITEM
7374	LAST
7472	LIST
7502	LISTP
7542	LPUT
7706	.REPLACE
7772	.REPTAIL
7862	MEMBERP
8194	NUMBERP
8254	SENTENCE (SE)
8482	WORDP
8512	WORD
8752	PIECE
8972	WHERE
8992	UPPERCASE (UC)
9014	LOWERCASE (LC)
9656	CATCH
9834	ERROR
9852	THROW
9984	LABEL
10004	GO
10446	NODES
10492	.CONTENTS
10542	TRACE
10568	NOTRACE
10660	WATCH
10746	NOWATCH
10878	MAKE
10932	LOCAL
11004	NAMEP
11070	THING
11096	NAME
11312	GPROP
11360	PLIST
11392	PPROP
11468	PPS
11678	REMPROP
11726	GLIST
11922	NOFORMAT

**TABLE DES ADRESSES DES MOTS CLES LOGO
CLASSES PAR ADRESSES**

<i>Adresse</i>	<i>Mot clé</i>
11964	DEFINE
12218	TEXT
12278	POCALL
12654	POTL
13020	POREF
13420	DEFINEDP
13444	PROCLIST
13462	PRIMITIVEP
13496	COPYDEF
13694	FOLLOW
13800	EDALL
13888	EDNS
13932	EDPS
13976	EDIT
14216	BURY
14264	UNBURY
14374	POPKG
14778	PO
14856	POALL
14886	POTS
14928	PONS
14982	POPS
15036	ERALL
15150	ERASE
15174	ERN
15208	ERNS
15228	ERPS
15288	PACKAGE
15360	PKGALL
16370	SETZOOM
16396	SETPAN
16526	BACK
16574	CLEAN
16600	CLEARSCREEN (CS)
16642	FORWARD (FD)
16680	TURTLEFACTS (TF)
16802	SCREENFACTS (SF)
16918	HIDETURTLE (HT)
16944	HOME
16968	LEFT
17004	RIGHT (RT)
17030	SETBG
17074	SETHEADING (SETH)
17100	SETPC

<i>Adresse</i>	<i>Mot clé</i>
17136	SETPEN
17352	SETPOS
17400	SETX
17448	SETY
17496	SHOWTURTLE (ST)
17522	TOWARDS
17760	KEYP
17782	WRAP
17808	CLEARTEXT (CT)
17822	SETSCRUNCH
18012	XCOR
18034	YCOR
18056	POS
18102	HEADING
18124	.EXAMINE
18204	.DEPOSIT
18522	PRINT (PR)
18538	TYPE
18674	SHOW
18722	READQUOTE (RQ)
18800	READCHAR (RC)
18870	READLIST (RL)
18928	COPYON
19010	COPYOFF
19628	ERASEFILE
19724	EDF
19948	LOAD
20118	SAVE
20470	CHANGEF
20584	SETPATH
20700	PATH
20880	DIR
25588	LOADPIC
25612	SAVEPIC
26000	BOX
26192	ARC
26254	CIRCLE
26380	SETFILL
26464	FILLATTR
26518	POLY
26628	ELLIPSE
26750	SETLINE
26830	LINEATTR
26886	SETTEXT

**L
A
N
G
A
G
E

L
O
G
O**

**TABLE DES ADRESSES DES MOTS CLES LOGO
CLASSES PAR ADRESSES**

<i>Adresse</i>	<i>Mot clé</i>
26954	GETTEXT
27388	FILL
28632	SETPAL
28740	PALETTE (PAL)
28812	TURTLETEXT (TT)
29214	MOUSE
33212	PENDOWN (PD)

<i>Adresse</i>	<i>Mot clé</i>
33238	PENUP (PU)
33252	PENERASE (PE)
33276	PENREVERSE (PX)
34588	FENCE
34482	WINDOW
63296	RECYCLE

GENERALITES

Le MC 68000 est un processeur 16/32 bits fabriqué par Motorola en technologie VLSI (intégration à très grande échelle). Il est inclus dans un boîtier à 64 broches. Ce nombre élevé est atteint à cause d'un jeu de signaux très complet et du fait de la séparation des bus de données et d'adresses.

Attardons nous sur son brochage pour montrer que la quantité de broches ne constitue pas un handicap mais, qu'au contraire, elle facilite son interfaçage avec d'autres composants. Le bus de données est de 16 bits mais il peut également fonctionner sur 8 bits.

Le bus d'adresses possède 23 bits (A1 à A23). Le bit A0 est remplacé par deux signaux UDS et LDS (Upper et Lower Data Strobe) qui indiquent respectivement l'accès à l'octet de poids fort ou de poids faible et également à un mot complet. L'espace d'adressage est donc de 16 Mo ou de 8 Mmots de 16 bits.

Une autre particularité du 68000 est son fonctionnement asynchrone. Contrairement aux processeurs 8 bits, qui eux travaillent avec l'horloge figurant sur tous leurs circuits périphériques, le 68000 signale quel est son mode de travail (utilisateur ou superviseur) et le type de cycle qu'il exécute (à l'aide des signaux FC0, FC1 et FC2).

L'attribution du bus pour le DMA (Direct Memory Acces) se réalise à l'aide des signaux BR (Bus Request), BG (Bus Grant) et BGACK (Bus Grant ACKnowledge).

La commande du bus se fait grâce aux signaux AS, UDS, LDS et R/W. Le signal DTACK indique que la donnée a été prise en compte.

Comme on peut le remarquer, l'asynchronisme fonctionne à la manière du "Handshaking".

Les commandes système sont au nombre de trois : BERR signale qu'une information erronée circule sur le bus, RESET permet la réinitialisation du processeur et de ses périphériques, HALT sert à le bloquer. Les deux

GENERALITES

dernières présentent une caractéristique bien particulière : ils sont bidirectionnels et donc également commandables par le processeur en personne.

Le 68000 dispose de trois broches d'interruption qui à elles trois fournissent le niveau de l'interruption (IPL0, IPL1 et IPL2). Un autre avantage de ce processeur est la possibilité de s'interfacer avec des composants de la famille 6800 à l'aide des signaux E, VPA et VMA.

<i>FC2</i>	<i>FC1</i>	<i>FC0</i>	<i>type de cycle</i>
0	0	0	réservé
0	0	1	données utilisateur
0	1	0	programme utilisateur
0	1	1	réservé
1	0	0	réservé
1	0	1	données superviseur
1	1	0	programme superviseur
1	1	1	reconnaissance d'interruption

ARCHITECTURE INTERNE DU MC 68000

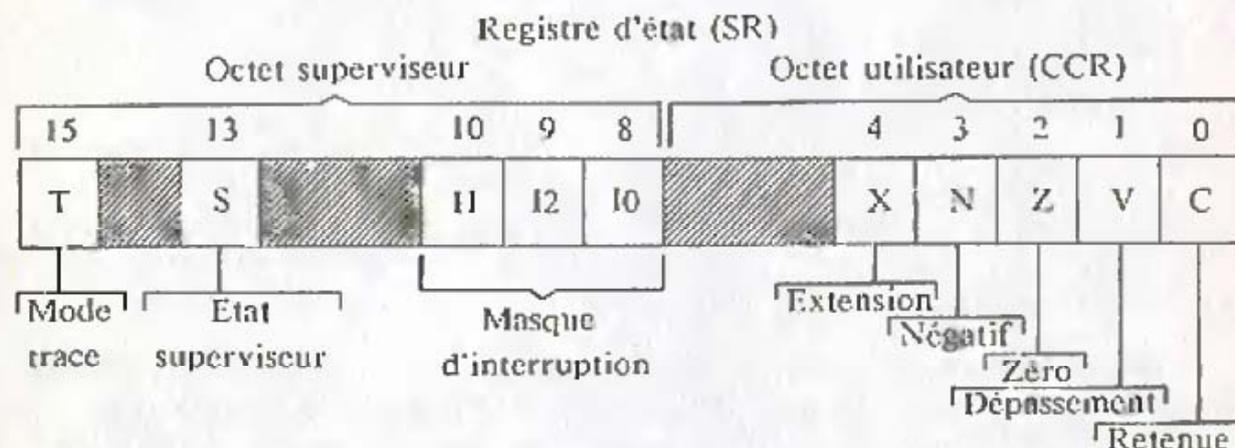
L'utilisateur dispose sur le 68000 de huit registres de données numérotés de D0 à D7. Chacun de ces registres présente une largeur de 32 bits mais le traitement de ces derniers peut se faire sur 1, 8 16 ou 32 bits. Lors du traitement des données au niveau du bit, tous les 32 bits des registres sont accessibles. Par contre, les traitements sur 8 ou 16 bits ne se font que sur les octets ou sur les mots de poids faible.

Il existe en outre sept registres d'adresses désignés par A0 à A6, d'une longueur de 32 bits chacun. Ils sont essentiellement utilisés pour l'adressage et non pour le calcul des données car on ne peut y traiter que des mots (16 bits) ou des mots longs (32 bits).

Le 68000 possède deux pointeurs de pile, l'un destiné au mode superviseur et l'autre, au mode utilisateur. Selon le mode de fonctionnement, le pointeur adéquat est activé. Ils portent tous deux le nom A7 et peuvent aussi être utilisés comme registre d'adresse 7.

Le compteur d'instructions est également un registre à 32 bits. Ainsi, un espace d'adressage de 4 giga-octets serait disponible. Mais, comme le bus d'adresses ne possède que 24 lignes, son domaine est limité à 16 méga octets.

Contrairement aux autres, le registre d'état n'est constitué que de 16 bits dont 10 seulement sont utilisés. En fait, il est scindé en deux : les 8 bits de poids faible représentent l'Octet Utilisateur, et les 8 bits de poids fort l'Octet Superviseur. En mode utilisateur, on ne peut modifier l'octet superviseur. Ainsi, seul le superviseur peut changer le mode de travail de l'unité centrale.



L'octet superviseur, parfois appelé octet système, se compose des bits suivants :

- Bit T : trace

Lorsqu'il est positionné (T=1), ce bit permet le déroulement pas à pas d'un programme pour, par exemple, contrôler son exécution lors de la mise au point.

ARCHITECTURE INTERNE DU MC 68000

■ Bit S : superviseur ou système

Ce bit indique dans quel mode (utilisateur : S=0 ; superviseur : S=1) le processeur évolue. Le mode superviseur est également réservé au système d'exploitation. Dans ce cas, il n'y a aucune restriction sur le jeu d'instructions ni sur les zones d'adresses. Avec le mode utilisateur, les instructions dites privilégiées sont inutilisables.

Grâce à l'installation judicieuse du matériel (hardware), il est possible d'interdire l'accès à certains emplacements de la mémoire ou l'accès à la zone d'entrées/sorties.

On peut ainsi facilement passer du mode superviseur au mode utilisateur, mais l'inverse est interdit.

■ Bits I2, I1 et I0 : masque d'interruption

Ces bits correspondent aux masques d'interruption et sont directement reliés aux broches IPL0 à IPL2. Ils indiquent le niveau d'interruption en cours, de sorte que seules les interruptions de niveau supérieur sont prises en compte.

Niveau	I2	I1	I0	
7	1	1	1	→ niveau le plus élevé.
6	1	1	0	
5	1	0	1	
4	1	0	0	
3	0	1	1	→ niveau le moins élevé.
2	0	1	0	
1	0	0	1	
0	0	0	0	→ aucune priorité.

L'octet **utilisateur** contient des résultats d'opérations arithmétiques et de comparaisons qui pourront être utilisés dans des branches de programmes conditionnées à ces résultats. C'est le registre des codes de condition (CCR). Les bits qui le constituent sont les suivants :

■ Bit N : négative

Ce bit est positionné si le bit le plus significatif du résultat d'une opération est à 1.

■ Bit Z : zéro

Ce bit est mis à 1 lorsque le résultat d'une opération est nul.

ARCHITECTURE INTERNE DU MC 68000

■ Bit V : overflow

Ce bit est utilisé pour indiquer à l'utilisateur que l'intervalle numérique couvert a été dépassé lors d'une opération arithmétique.

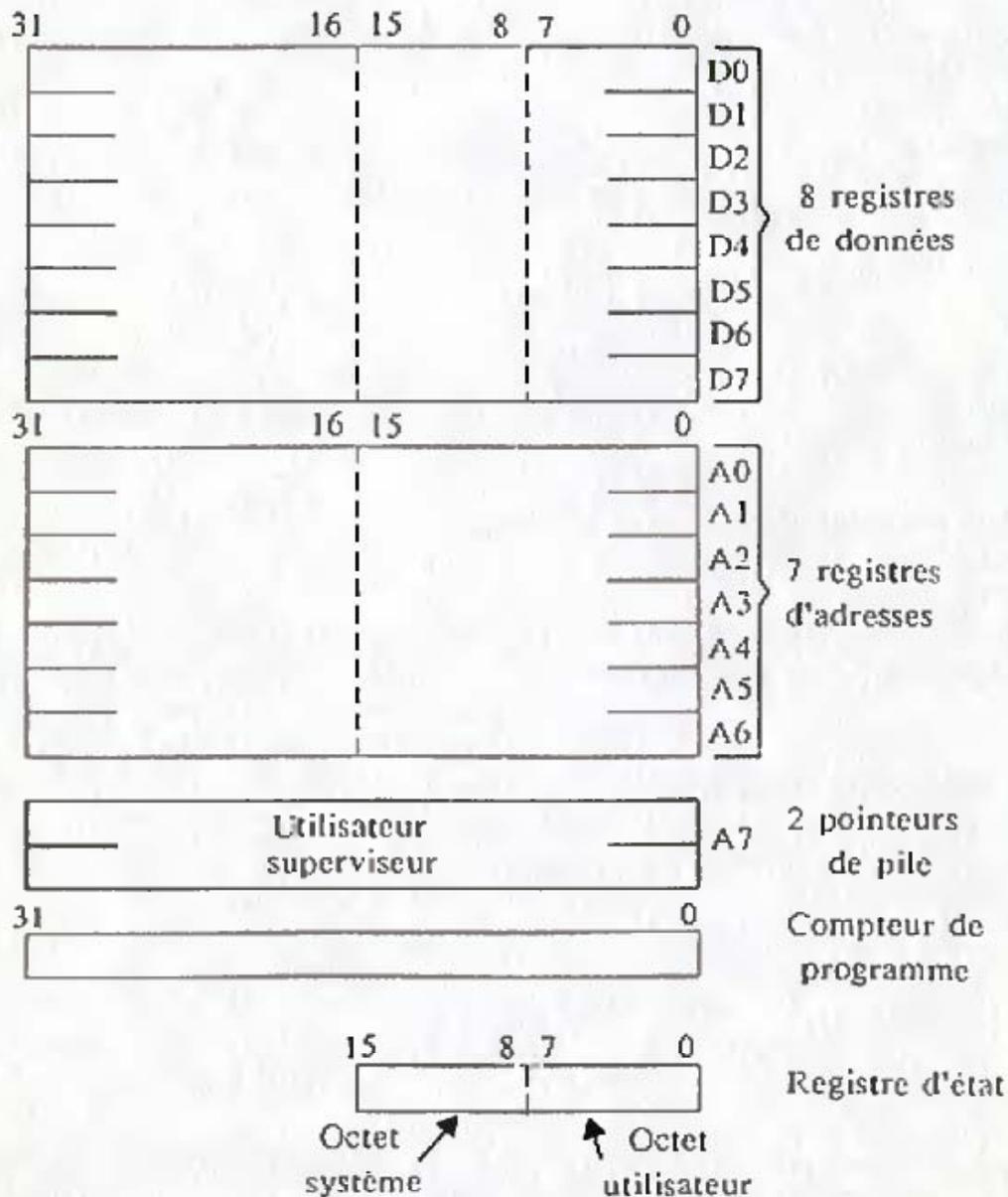
■ Bit C : carry

Ce bit indique qu'une retenue a été générée.

■ Bit X : extend

Ce bit présente le même comportement que le bit C pour les opérations sur les opérandes étendus.

Structure Interne



ARCHITECTURE INTERNE DU MC 68000

Modes d'adressage

Les modes d'adressage du 68000 constituent incontestablement sa principale caractéristique. Son énorme capacité de traitement provient principalement de ses nombreuses possibilités d'adressage. Il en possède 14 types distincts qui sont regroupés sous 6 rubriques :

Adressage absolu

Dans ce type d'adressage, l'adresse de l'opérande est précisée dans l'instruction. Suivant sa largeur, on parle d'adressage absolu court ou long.

- **Adressage absolu court** traite des adresses d'une largeur d'un mot (16 bits). Dans ce cas, le mot est étendu à un mot long (32 bits) en y incluant le signe avant l'exécution de l'opération. L'accès mémoire est limité à 64 K octets.

NOT.W \$3FFF

L'adresse \$3FFF devient \$00003FFF. Si l'adresse était \$FF00, elle serait devenue \$FF000000.

- **Adressage absolu long** utilise des adresses d'une largeur de 24 bits, ce qui permet d'accéder à 16 méga-octets de mémoire.

NOT.L \$04F000

Adressage direct de registres

La seule différence avec l'adressage absolu est que l'opérande source est un registre. Celui-ci peut être un registre de données, un registre d'adresses ou le registre d'état.

- **Un registre de données** est un des huit registres 32 bits Dm. L'opérande peut être un bit, un octet, un mot ou un mot long. ADD.B D0,D1
- **Un registre d'adresses** qui peut être un des sept registres 32 bits An. Dans ce cas, seuls sont possibles les mots et les mots longs. MOVE .L A1,D4
- **Le registre d'état** Les instructions utilisant ce type d'adressage sont des instructions privilégiées :

ANDI to SR
EORI to SR
MOVE to SR
ORI to SR

EORI \$007F.SR

Adressage immédiat

L'opérande est pris tel quel dans les arguments de l'instruction. Dans cet adressage, une adresse \$XXXX est notée #XXXXX.

ARCHITECTURE INTERNE DU MC 68000

- **Adressage immédiat simple** : l'opérande suit directement l'instruction et peut avoir une longueur d'un mot ou d'un mot long.

ADDI.W #\$AFAF,D2

lors du transfert d'un mot, le registre d'adresses entier est concerné car le mot subit une extension sur 32 bits avant le transfert.

MOVEA.W,A2 mettra S00002000 dans A2.

- **Adressage immédiat rapide** : la donnée est incluse dans le code de l'instruction. Dans cette variante, les constantes ne doivent porter que sur 8 bits et ne peuvent donc prendre que les valeurs de 0 à 7, sauf pour l'instruction MOVE vers un registre de données où les constantes peuvent porter sur 8 bits.

ADDQ .L #1,A0 (ADD Quick)

MOVEQ #200,D2

Adressage indirect de registres

En général, dans ce mode, un registre cité dans l'instruction contient l'adresse de l'opérande. Ce type d'adressage comporte quatre variantes :

- **Adressage indirect simple** : l'adresse de l'opérande est contenue dans le registre d'adresses spécifié dans l'instruction.

MOVE .W (A3),D4

- **Adressage indirect post-incrémenté** : l'adressage de l'opérande se fait de la même manière que précédemment, mais, après l'opération, le contenu du registre d'adresses est incrémenté de N :

N=1 si l'opérande est un octet (.B).

N=2 si l'opérande est un mot (.W ou pas d'extension).

N=4 si l'opérande est un mot double (.L).

BSET .B #0,(A0)+

- **Adressage indirect prédécémenté** : dans ce cas, le registre d'adresses est diminué de la même valeur N avant l'adressage de l'opérande.

AND.L D2,-(A6)

Ces deux derniers modes d'adressage permettent à l'utilisateur de réaliser très aisément des structures de pile et de file d'attente. Il utilisera, par exemple, la prédécémentation pour remplir la pile et la postincrémentation pour la vider.

- **Adressage indirect avec déplacement** : l'opérande se trouve à une adresse égale à la somme de celle contenue dans le registre d'adresses et du déplacement codé sur 16 bits (et étendu à 32 bits).

EOR.L D1,\$0200(A3)

- **Adressage indirect indexé avec déplacement** : l'adresse de l'opérande est égale au mode précédent auquel on rajoute cependant le contenu d'un autre registre (l'index). Le décalage ne porte ici que sur 8 bits.

MOVE.W \$12 (A5,D2.L),A3

↑
déplacement

↑
registre d'index

ARCHITECTURE INTERNE DU MC 68000

Adressage relatif au compteur d'instructions (de programme)

Dans ce cas, l'adresse effective est fonction d'un déplacement codé sur 16 bits et du compteur d'instructions. Les programmes écrits avec ce mode d'adressage peuvent tourner à n'importe quelle adresse du système et sont donc "relogeables".

- Adressage relatif au compteur d'instructions (avec déplacement) : les 16 bits du déplacement contenus dans un mot d'extension de l'instruction sont ajoutés au compteur d'instructions. Pour les instructions de branchement, le déplacement peut être codé sur 8 bits.

MOVE.B \$0F00 (PC),D0

- Adressage Indexé relatif au compteur d'instructions : pour déterminer l'adresse de l'opérande, il faut ajouter au compteur de programme un déplacement codé sur 8 bits et le contenu d'un registre d'index.

MOVE .W \$12 (PC, D0 .W), \$3000

Adressage implicite

L'opérande est indiqué de façon implicite dans l'instruction. Les opérations sur les piles contiennent par exemple implicitement le pointeur de pile comme pointeur sur l'adresse de l'opérande.

MOVE SR,D3

Récapitulatif des modes d'adressage

Mode	Adresse effective
Adressage absolu : - absolu court - absolu long	AE = (mot suivant) AE = (2 mots suiv.)
Adressage direct de registres : - registre de données - registre d'adresses	AE = Dn AE = An
Adressage immédiat : - immédiat simple - immédiat rapide	donnée = mot suivant donnée implicite
Adressage indirect de registres : - indirect simple - indirect post-incrémenté - indirect pré-décrémenté - indirect avec déplacement - indirect indexé avec déplacement	AE = (An) AE = (An) ; An ← An+N An ← An-N ; AE = (An) AE = (An) + d16 AE = (An)+(Xn) + d8
Adressage relatif au compteur d'instructions : - relatif avec déplacement - relatif avec index et déplacement	AE = (PC) + d16 AE = (PC) + (Xn) + d8
Adressage implicite	AE = SR, USP, SP, PC

JEU D'INSTRUCTIONS

Le jeu d'instructions du MC 68000 comporte 56 types d'instructions. C'est peu si on le compare à d'autres processeurs, mais, à l'aide de ses 14 modes d'adressage, il lui est en réalité possible de générer plus de 1000 instructions différentes.

De nombreuses instructions du 68000 peuvent travailler avec différents types de données. La plupart opèrent sur des octets, sur des mots ou sur des mots longs.

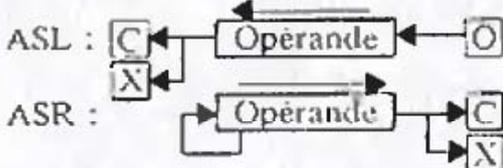
Une instruction du 68000 comporte 1 à 5 mots. La longueur et le type sont déterminés par le premier mot de l'instruction.

Description de chaque instruction

Mnémonique	Opération réalisée	codes cond X N Z V C
ABCD	Addition décimale avec le bit d'extension entre deux registres de données ou entre deux cases mémoire adressées par prédécrémentaion à l'aide des registres d'adresses. (destination) ₁₀ +(source) ₁₀ +X --> (destination) Syntaxe : ABCD Dd,Ds ou ABCD -(Ad),-(As)	* I * I *
ADD	Addition binaire entre un opérande et le contenu d'un registre de données. (destination) + (source) --> (destination) Syntaxe : ADD [AE],Dn ou ADD Dn,[AE]	* * * * *
ADDA	Addition binaire entre un opérande et le contenu d'un registre d'adresses. (destination) + (source) --> (destination) Syntaxe : ADDA [AE],An	- - - - -
ADDI	Addition immédiate entre la donnée immédiate et l'opérande de destination. (destination) + donnée imméd. --> (destinat) Syntaxe : ADDI #[donnée],[AE]	* * * * *

JEU D'INSTRUCTIONS

Mnémonique	Opération réalisée	X N Z V C
ADDQ	Addition rapide entre la donnée immédiate et l'opérande de destination. La donnée ne peut prendre que les valeurs de 0 à 7 (000 111). [donnée imméd.] + (destinat.) --> (destinat) Syntaxe : ADDQ #[donnée],[AE]	* * * * *
ADDX	Addition binaire avec le bit d'extension entre l'opérande destination et l'opérande source. Les opérandes sont contenus dans des registres de données ou dans des positions mémoire. (destination) + (source) + X --> destination Syntaxe : ADDX Dd,Ds ou ADDX -(Ad),-(As)	* * * * *
AND	ET logique entre deux opérandes qui ne peuvent être contenus dans un registre d'adresses. (destination) ^ (source) --> (destination) Syntaxe : AND [AE],Dn ou AND Dn,[AE]	- * * 0 0
ANDI	ET logique immédiat entre la donnée immédiate et l'opérande de destination. [donnée imméd.] ^ [destination] --> destination Syntaxe : ANDI # [donnée],[AE]	- * * 0 0
ANDI to CCR	ET logique entre la donnée immédiate et le registre des codes de conditions. [don. imméd.] ^ [reg codes cond] --> reg c.cond Syntaxe : ANDI # [donnée 8 bits],CCR	* * * * *
ANDI to SR	ET logique entre la donnée immédiate et le registre d'état SR si le processeur est en état superviseur. Sinon, l'instruction est interdite. [donnée immédiate] ^ SR --> SR Syntaxe : ANDI # [donnée 16 bits],SR	* * * * *

Mnémonique	Opération réalisée	X N Z V C																										
ASL, ASR	<p>Décalage arithmétique de l'opérande dans la direction indiquée.</p> <p>Pour le décalage d'un registre, le nombre de décalages peut être :</p> <ul style="list-style-type: none"> - une valeur immédiate (0 à 7) - une valeur contenue dans un registre de données. <p>Pour le décalage du contenu d'une case mémoire, seul un décalage peut être effectué et l'opérande est un mot.</p> <p>(destinat) décalée du [nb de décal] --> dest</p> <p>Syntaxe : ASi Ds,Dd ASi # [donnée],Dd ASi [AE] avec i = L ou i = R</p> 	*****																										
Bcc	<p>Branchement conditionnel à l'adresse (PC) + déplacement.</p> <ul style="list-style-type: none"> - La valeur du PC vaut l'adresse de l'instruction actuelle + 2. - Si le déplacement est de 8 bits, il est contenu dans l'instruction ; s'il est de 16 bits, il suit l'instruction. - Les conditions sont : <table border="1" data-bbox="470 1500 1173 2049"> <tbody> <tr> <td>CC Retenue à zéro</td> <td>C = 0</td> </tr> <tr> <td>CS Retenue à un</td> <td>C = 1</td> </tr> <tr> <td>EQ Egal</td> <td>Z = 1</td> </tr> <tr> <td>NE Différent</td> <td>Z = 0</td> </tr> <tr> <td>GE Supérieur ou égal</td> <td>$N \oplus V = 0$</td> </tr> <tr> <td>GT Supérieur</td> <td>$Z + (N \oplus V) = 0$</td> </tr> <tr> <td>HI Supérieur</td> <td>C + Z = 0</td> </tr> <tr> <td>LE Inférieur ou égal</td> <td>$Z + (N \oplus V) = 1$</td> </tr> <tr> <td>LS Inférieur ou égal</td> <td>C + Z = 1</td> </tr> <tr> <td>LT Inférieur</td> <td>$N \oplus V = 1$</td> </tr> <tr> <td>MI Négatif</td> <td>N = 1</td> </tr> <tr> <td>PL Positif</td> <td>N = 0</td> </tr> <tr> <td>VC Dépassement à zéro</td> <td>V = 0</td> </tr> </tbody> </table>	CC Retenue à zéro	C = 0	CS Retenue à un	C = 1	EQ Egal	Z = 1	NE Différent	Z = 0	GE Supérieur ou égal	$N \oplus V = 0$	GT Supérieur	$Z + (N \oplus V) = 0$	HI Supérieur	C + Z = 0	LE Inférieur ou égal	$Z + (N \oplus V) = 1$	LS Inférieur ou égal	C + Z = 1	LT Inférieur	$N \oplus V = 1$	MI Négatif	N = 1	PL Positif	N = 0	VC Dépassement à zéro	V = 0	-----
CC Retenue à zéro	C = 0																											
CS Retenue à un	C = 1																											
EQ Egal	Z = 1																											
NE Différent	Z = 0																											
GE Supérieur ou égal	$N \oplus V = 0$																											
GT Supérieur	$Z + (N \oplus V) = 0$																											
HI Supérieur	C + Z = 0																											
LE Inférieur ou égal	$Z + (N \oplus V) = 1$																											
LS Inférieur ou égal	C + Z = 1																											
LT Inférieur	$N \oplus V = 1$																											
MI Négatif	N = 1																											
PL Positif	N = 0																											
VC Dépassement à zéro	V = 0																											

JEU D'INSTRUCTIONS

Mnémonique	Opération réalisée	X N Z V C						
	<table border="1" data-bbox="483 304 1190 488"> <tr> <td>VS Dépassement à un</td> <td>V = 1</td> </tr> <tr> <td>T Vrai</td> <td>1</td> </tr> <tr> <td>F Faux</td> <td>0</td> </tr> </table> <p data-bbox="483 528 1098 600">Si (condition vraie) => (PC) + d --> PC Syntaxe : Bcc [étiquette]</p>	VS Dépassement à un	V = 1	T Vrai	1	F Faux	0	
VS Dépassement à un	V = 1							
T Vrai	1							
F Faux	0							
BCHG	<p data-bbox="483 678 1190 786">Un bit de l'opérande de destination est testé et affecte l'indicateur Z, puis ce bit est inversé.</p> <p data-bbox="483 790 1031 898">Bit n°xxx de la destination est testé --> Z est affecté --> le bit est inversé</p> <p data-bbox="483 902 895 976">Syntaxe : BCHG Dn,[AE] ou BCHG #[donnée],[AE]</p>	-- * --						
BCLR	<p data-bbox="483 1055 1190 1162">Un bit de l'opérande de destination est testé et affecte l'indicateur Z, puis ce bit est mis à zéro.</p> <p data-bbox="483 1167 970 1274">Bit n°xxx de la destination testé --> Z est affecté --> le bit est mis à zéro</p> <p data-bbox="483 1279 895 1352">Syntaxe : BCLR Dn,[AE] ou BCLR #[donnée],[AE]</p>	-- * --						
BRA	<p data-bbox="483 1431 1190 1505">Branchement inconditionnel à l'adresse (PC) + déplacement.</p> <ul data-bbox="483 1509 1190 1693" style="list-style-type: none"> - La valeur du PC vaut l'adresse de l'instruction actuelle plus deux. - Si le déplacement est sur 8 bits, il est contenu dans l'instruction; s'il est sur 16 bits, il suit l'instruction. <p data-bbox="483 1697 767 1733">(PC) + d --> PC</p> <p data-bbox="483 1738 932 1774">Syntaxe : BRA [étiquette]</p>	-----						
BSET	<p data-bbox="483 1848 1190 1955">Un bit de l'opérande de destination est testé et affecte l'indicateur Z, puis, ce bit est mis à 1.</p> <p data-bbox="483 1960 963 2067">Bit n°xxx de la destination testé --> Z est affecté --> le bit est mis à 1</p>	-- * --						

Anémorique	Opération réalisée	X N Z V C
	<p>Syntaxe : BSET Dn,[AE] ou BSET #[donnée],[AE]</p>	
BSR	<p>Branchement à un sous-programme. Le compteur de programme est sauvegardé dans la pile et l'exécution du programme se poursuit à l'adresse (PC) + un déplacement. PC --> -(SP) (PC) + d --> PC Syntaxe : BSR [étiquette]</p>	- - - - -
BTST	<p>Un bit de l'opérande de destination est testé et son état est reflété par l'indicateur Z. Bit n°xxx de la destination testé --> affectation de Z. Syntaxe : BTST Dn,[AE] ou BTST #[donnée],[AE]</p>	- - * - -
CHK	<p>Examiner si la valeur spécifiée par l'adresse effective est contenue dans l'intervalle [0,(Dn)] et générer une exception dans le cas contraire. Si Dn < 0 ou Dn > ([EA]) alors exception. Syntaxe : CHK [AE],Dn</p>	- * I I I
CLR	<p>Mise à zéro des bits de l'opérande destination. 0 --> destination Syntaxe : CLR [AE]</p>	- 0 1 0 0
CMP	<p>L'opérande source est soustrait de l'opérande destination en affectant les indicateurs. La destination est inchangée. ([Dn]) - ([AE]) (destination) - (source) Syntaxe : CMP [AE],Dn</p>	- * * * *

Mnémonique	Opération réalisée	X N Z V C
	(destination) / (source) --> destination Syntaxe : DIVS [AE],Dn	
DIVU	Division réalisée en arithmétique non signée. Les caractéristiques sont les mêmes que pour l'instruction précédente. (destination) / (source) --> destination Syntaxe : DIVU [AE],Dn	- * * * 0
EOR	OU exclusif entre la destination et la source. (destination)⊕(source) --> destination Syntaxe : EOR Dn,[AE]	- * * 0 0
EORI	OU exclusif entre la donnée immédiate et l'opérande destination. [donnée imméd.]⊕(destination) --> destinat Syntaxe : EORI #[donnée],[AE]	- * * 0 0
EORI to CCR	OU exclusif entre la donnée 8 bits et les 8 bits de poids faible du registre d'état. [donnée immédiate]⊕CCR --> CCR Syntaxe : EORI #[donnée 8 bits],CCR	* * * * *
EORI to SR	OU exclusif entre la donnée immédiate et le registre d'état si le processeur est en état superviseur. Sinon, il y a génération d'exception (n°8). [donnée immédiate]⊕SR --> SR si état superviseur. Syntaxe : EORI #[donnée 16 bits],SR	* * * * *
EXG	Echange entre deux registres de 32 bits (donnée ou adresse). Rx <--> Ry Syntaxe : EXG Rx,Ry	- - - - -
EXT	L'opérande destination est complété avec son bit 7 (cas d'un mot) ou avec son bit 16 (cas d'un mot long).	- * * 0 0

JEU D'INSTRUCTIONS

Mnémonique	Opération réalisée	X N Z V C
	Syntaxe : EXT Dn	
ILLEGAL	L'instruction dont le code hexadécimal est \$4AFC génère une exception (n°4). (PC) --> -(SSP) ; (SR) --> -(SSP) (vecteur instruction illégale) --> PC Syntaxe : non imposée.	- - - - -
JMP	Le déroulement du programme continue à l'adresse précisée. destination --> PC Syntaxe : JMP [AE]	- - - - -
JSR	Le déroulement du programme continue à l'adresse précisée. L'adresse de l'instruction suivant le JSR est conservée dans la pile. PC --> (SP) ; destination --> PC Syntaxe : JSR [AE]	- - - - -
LEA	Charge un registre d'adresses avec l'adresse effective. destination --> An Syntaxe : LEA [AE],An	- - - - -
LINK	Le contenu du registre d'adresses est placé dans la pile. Ce registre est alors chargé avec SP. Enfin, le déplacement est ajouté à SP. An --> -(SP) SP --> An SP + d --> SP Syntaxe : LINK An, #[déplacement sur 16 bits]	- - - - -
LSL,LSR	Même effet et mêmes restrictions que pour ASL et ASR.	* * * 0 *

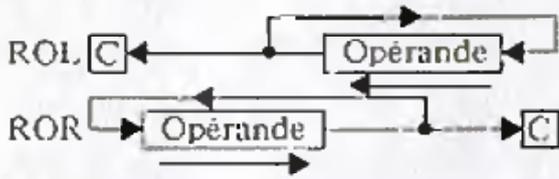
<i>Mnémonique</i>	<i>Opération réalisée</i>	<i>X N Z V C</i>
MOVE	Transfert de l'opérande source vers l'opérande de destination. (source) --> (destination) <i>Syntaxe :</i> MOVE [AE],[AE]	- * * 0 0
MOVE to CCR	Transfert de l'opérande source dans le registre codes de conditions. Les 8 bits de poids fort sont ignorés. (source) --> CCR <i>Syntaxe :</i> MOVE [AE],CCR	* * * * *
MOVEA	Transfert de l'opérande source vers un registre d'adresses. (source) --> destination <i>Syntaxe :</i> MOVEA [AE],An	- - - - -
MOVE USP	Transfert du contenu du pointeur de pile de ou vers un registre d'adresses. si état superviseur alors USP --> An ou An --> USP sinon génération d'une exception. <i>Syntaxe :</i> MOVE USP,A1 ou MOVE A1,USP	- - - - -
MOVE from SR	Transfert du registre d'état vers l'opérande destination. SR --> destination <i>Syntaxe :</i> MOVE SR,[AE]	- - - - -
MOVE to SR	Transfert de l'opérande source vers le registre d'état si état superviseur, sinon, génération d'une exception. Si état superviseur alors (source) --> SR sinon exception. <i>Syntaxe :</i> MOVE [AE],SR	* * * * *
MOVEM	Transfert des registres cités vers ou à partir de l'adresse mémoire indiquée par l'adresse effective. Registres --> destination	- - - - -

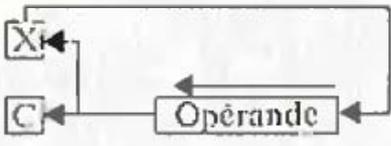
JEU D'INSTRUCTIONS

Mnémonique	Opération réalisée	X N Z V C
	(source) --> registres <i>Syntaxe</i> : MOVEM [liste de registres],[AE] ou MOVEM [AE],[liste de registres]	
MOVEP	Transfert de données entre des registres de données et une position mémoire à partir de l'adresse indiquée et par incrémentation de deux. Le transfert se fait par octet en commençant par celui de poids fort. Si l'adresse est paire, le transfert se fait sur la moitié supérieure du bus de données. Si l'adresse est impaire, le transfert se fait sur la moitié inférieure. (source) --> destination <i>Syntaxe</i> : MOVEP Ds,d(Ad) ou MOVEP d(As),Dd	- - - - -
MOVEQ	Transfert rapide d'une donnée immédiate vers un registre de données. la donnée codée sur 8 bits est contenue dans l'opérande. [donnée immédiate] --> registre <i>Syntaxe</i> : MOVEQ #[donnée],Dn	- * * 0 0
MULS	Multiplication de deux opérandes 16 bits, signés et résultat signé sur 32 bits. [source] X [destination] --> destination <i>Syntaxe</i> : MULS [AE],Dn	- * * 0 0
MULU	Multiplication de deux opérandes 16 bits sans considération de signe. [source] X [destination] --> destination <i>Syntaxe</i> : MULU [AE],Dn	- * * 0 0
NBCD	L'opérande situé à l'emplacement destination et le bit X sont soustraits à zéro en décimal. 0 - [destination] - X --> destination <i>Syntaxe</i> : NBCD [AE]	* - * - *

Mnémonique	Opération réalisée	X N Z V C
NEG	L'opérande situé à l'emplacement destination est soustrait de zéro. 0 - [destination] --> destination Syntaxe : NEG [AE]	*****
NEGX	L'opérande situé à l'emplacement destination et le bit X sont soustraits à zéro. 0 - [destination] - X --> destination Syntaxe : NEGX [AE]	* - * - *
NOP	Aucune opération n'est effectuée. Syntaxe : NOP	- - - - -
NOT	Complément à 1 de l'opérande destination. [destination] --> destination Syntaxe : NOT [AE]	- * * 0 0
OR	Effectue un OU logique entre les deux opérandes qui sont des registres de données. [source] V [destination] --> destination Syntaxe : OR [AE],Dn ou OR Dn,[AE]	- * * 0 0
ORI	Effectue un OU logique entre la donnée immédiate et l'opérande destination. [donnée immédiate] V [destination] --> dest. Syntaxe : ORI #[donnée],[AE]	- * * 0 0
ORI to CCR	Effectue un OU logique entre la donnée immédiate et le registre codes de conditions. [source] V CCR --> CCR Syntaxe : ORI #[donnée],CCR	*****
ORI to SR	Effectue un OU logique entre la donnée immédiate et le registre d'état si le processeur est en état superviseur. Sinon, il y a génération d'une exception (n°8).	*****

JEU D'INSTRUCTIONS

Mnémonique	Opération réalisée	X N Z V C
	[source] V SR --> SR Syntaxe : ORI #[donnée 16 bits],SR	
PEA	L'adresse effective est sauvegardée dans la pile. destination --> -(SP) Syntaxe : PEA [AE]	- - - - -
RESET	Si le processeur est en état superviseur, la broche RESET est activée afin de remettre à zéro les circuits externes. Sinon, il y a génération d'une exception. Syntaxe : RESET	- - - - -
ROL, ROR	<p>Rotation dans la direction précisée des bits de l'opérande. Dans le cas d'une case mémoire, une seule rotation peut être réalisée. Dans le cas d'un registre, le nombre de rotations est signifié soit par une valeur immédiate contenue dans l'instruction, soit par un registre.</p>  <p>[destination] subit une rotation de [n] --> destination Syntaxe : ROi Ds,Dd ou ROi #[donnée immédiate],Dd ou ROi [AE] avec i=L ou R.</p>	- * * 0 *
ROXL, ROXR	Rotation dans la direction précisée des bits de l'opérande par le bit X. Dans le cas d'une case mémoire, une seule rotation est permise. Dans le cas d'un registre, le nombre de rotations est signifié soit par une valeur immédiate contenue dans l'instruction, soit par un registre.	* * * 0 *

Mnémonique	Opération réalisée	X N Z V C
	<p>ROXL </p> <p>ROXR </p> <p>[destination] subit une rotation de [n] --> destination Syntaxe : ROXi Ds,Dd ou ROXi #[donnée],Dd ou ROXi [AE]</p>	
RTE	<p>Si le processeur est en état superviseur, il y a retour d'exception ; c'est-à-dire que le registre d'état et le compteur de programme sont restitués par la pile. Sinon, il y a génération d'une exception. (SP)+ --> SR (SP)+ --> PC Syntaxe : RTE</p>	*****
RTR	<p>Retour et restitution des codes de conditions. (SP)+ --> CCR (SP)+ --> PC Syntaxe : RTR</p>	*****
RTS	<p>Retour de sous-programme. (SP)+ --> PC Syntaxe : RTS</p>	-----
SBCD	<p>L'opérande source et le bit X sont soustraits de l'opérande destination en décimal. [destination]₁₀ - [source]₁₀ --> dest. Syntaxe : SBCD Ds,Dd ou SBCD -(As).-(Ad)</p>	* - * - *

JEU D'INSTRUCTIONS

Mnémonique	Opération réalisée	X N Z V C
Sec	<p>L'octet indiqué par l'adresse effective est mis à \$FF si la condition est remplie, sinon, il est effacé. (Pour les conditions, reportez-vous à l'instruction Bcc).</p> <p>Si cc vraie alors \$FF --> destination Si cc fausse alors \$FF --> destination</p> <p>Syntaxe : Sec [AE]</p>	- - - - -
STOP	<p>S'il est en état superviseur, le processeur est interrompu. La donnée immédiate est transférée dans le registre d'état. L'exécution des instructions n'est réactivée que par une interruption externe</p> <p>[donnée immédiate] --> SR ; puis arrêt</p> <p>Syntaxe : STOP #[donnée 16 bits]</p>	* * * * *
SUB	<p>L'opérande source est soustrait de l'opérande de destination.</p> <p>[destination] - [source] --> destination</p> <p>Syntaxe : SUB [AE],Dn ou SUB Dn,[AE]</p>	* * * * *
SUBA	<p>L'opérande source est soustrait de l'opérande de destination qui est un registre d'adresses.</p> <p>[destination] - [source] --> destination</p> <p>Syntaxe : SUBA [AE],An</p>	- - - - -
SUBI	<p>La donnée immédiate est soustraite de l'opérande destination.</p> <p>[destination] - [donnée imméd] --> destinat</p> <p>Syntaxe : SUBI #[donnée immédiate],[AE]</p>	* * * * *
SUBQ	<p>La donnée immédiate est soustraite de l'opérande destination. La donnée fait partie de l'instruction et peut prendre les valeurs de 1 à 8. Les opérations sur les mots et sur les mots doubles sont aussi permises avec les registres d'adresses, mais alors les indicateurs ne sont pas positionnés.</p>	* * * * *

Mnémonique	Opération réalisée	X N Z V C
	[destination] - [donnée imméd] --> destinat. Syntaxe : SUBQ #[donnée],[AE]	
SUBX	L'opérande source et l'indicateur X sont soustraits de l'opérande destination. [destination] - [source] - X --> destination Syntaxe : SUBX Ds,Dd ou SUBX -(As),-(Ad)	* * * * *
SWAP	Echange des 16 bits de poids fort avec les 16 bits de poids faible. bits [31:16] <--> [15: 0] Syntaxe : SWAP Dn	- * * 0 0
TAS	Test d'un octet adressé par l'adresse effective, puis le bit 7 de l'octet est mis à 1. Syntaxe : TAS [AE]	- * * 0 0
TRAP	Le processeur exécute une procédure d'exception. Le numéro du vecteur est indiqué par 4 bits contenus dans l'instruction. PC --> -(SSP) SR --> -(SSP) (vecteur) --> PC Syntaxe : TRAP #[vecteur]	- - - - -
TRAPV	Si l'indicateur V est positionné, il y a génération de l'exception dont le vecteur vaut 7. Sinon, l'exécution se poursuit. Syntaxe : TRAPV	- - - - -
TST	L'opérande est comparé à zéro; les indicateurs sont positionnés en fonction du résultat. [adresse effective] testée --> positionnement des indicateurs Syntaxe : TST [AE]	- * * 0 0

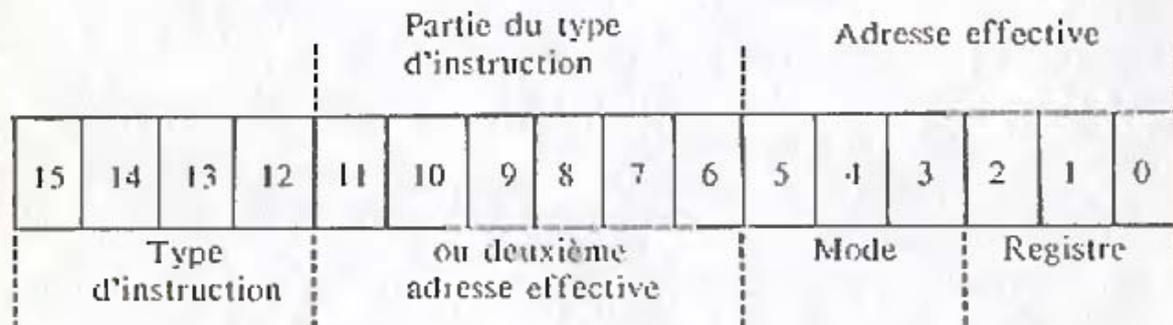
JEU D'INSTRUCTIONS

<i>Mnémonique</i>	<i>Opération réalisée</i>	<i>X N Z V C</i>
UNLK	<p>Le registre d'adresses spécifié est transféré dans le pointeur de pile et le contenu de la pile est ensuite chargé dans le registre d'adresses.</p> <p>An --> SP [SP]+ --> An Syntaxe : UNLK An</p>	- - - - -
<i>Légende :</i>	<p>* signifie positionné I signifie indéfini - signifie non affecté 0 signifie toujours mis à zéro I signifie toujours mis à un.</p>	

TABLE DES CODES OPERATOIRES TRIES PAR CODE

Etant donné la puissance du mode d'adressage et le nombre de registres dont dispose le 68000, la représentation de tous les codes opératoires nécessiterait un volume entier. Par conséquent, nous avons choisi de vous fournir plusieurs exemples pour chaque instruction. A partir de ces exemples, de l'explication des modes d'adressage donnée précédemment et des quelques règles suivantes, le lecteur sera à même de déterminer les codes qui lui seront utiles.

Pour les instructions avec adresse effective, les codes opératoires ont la structure suivante :



<i>Type d'instruction</i>	<i>Bits 15 à 12</i>
Manipulation de bits, MOVEP	0 0 0 0
MOVE sur des octets	0 0 0 1
MOVE sur des mots longs	0 0 1 0
MOVE sur des mots	0 0 1 1
Autres opérations	0 1 0 0
ADDQ, SUBQ, Sec, DBcc	0 1 0 1
Bcc, BSR	0 1 1 0
MOVEQ	0 1 1 1
DIV, OR, SBCD	1 0 0 0
SUB, SUBX	1 0 0 1
Non utilisé	1 0 1 0
CMP, EOR	1 0 1 1
ABCD, AND, EXG, MUL	1 1 0 0
ADD, ADDX	1 1 0 1
Décalages et rotations	1 1 1 0
Non utilisés	1 1 1 1

Les modes d'adressage se codifient de la façon suivante dans la partie adresse effective du code :

TABLE DES CODES OPERATOIRES TRIES PAR CODE

<i>Mode d'adressage</i>	<i>Mode</i>	<i>Registre</i>
Direct de registre de donnée	0 0 0	numéro du registre
Direct registre d'adresses	0 0 1	numéro du registre
Indirect registre d'adresses	0 1 0	numéro du registre
Indirect registre d'adresses avec post-incrémentation	0 1 1	numéro du registre
Indirect registre d'adresses avec pré-incrémentation	1 0 0	numéro du registre
Indirect registre d'adresses avec déplacement	1 0 1	numéro du registre
Indirect registre d'adresses avec index	1 1 0	numéro du registre
Absolu court	1 1 1	0 0 0
Absolu long	1 1 1	0 0 1
Relatif au compteur de programme avec déplacement	1 1 1	0 1 0
Relatif au compteur de programme avec index	1 1 1	0 1 1
Immédiat ou registre d'état	1 1 1	1 0 0

Aucune adresse effective n'est nécessaire avec l'adressage implicite.

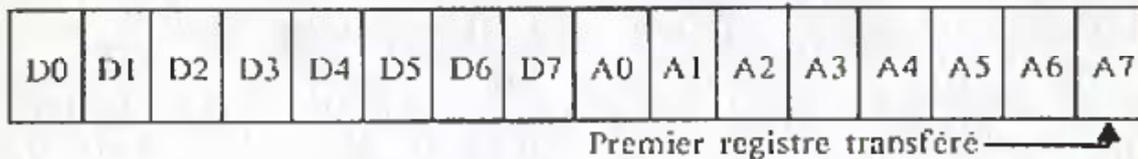
Les conditions des instructions conditionnelles (Bcc, DBcc et Scc) sont codées de la manière suivante :

0 1 0 0	CC	retenue à zéro
0 1 0 1	CS	retenue à un
0 1 1 0	NE	différent
0 1 1 1	EQ	égal
1 1 0 0	GE	supérieur ou égal
1 1 1 0	GT	supérieur à
0 0 1 0	HI	supérieur
1 1 1 1	LE	supérieur ou égal
0 0 1 1	LS	inférieur ou égal
1 1 0 1	LT	inférieur
1 0 1 1	MI	négatif
1 0 1 0	PL	positif
1 0 0 0	VC	dépassement à zéro
1 0 0 1	VS	dépassement à un
0 0 0 0	T	vrai
0 0 0 1	F	faux

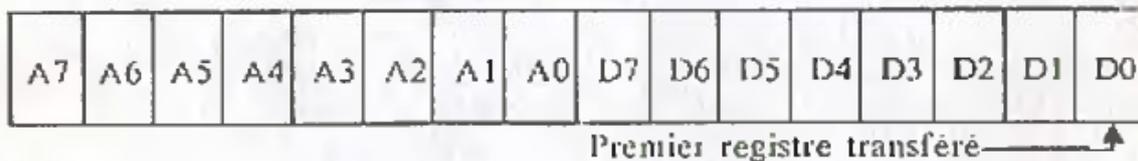
TABLE DES CODES OPERATOIRES TRIES PAR CODE

Pour le transfert multiple de registres (MOVEM), le mot d'extension de l'instruction porte le nom de masque des registres. Lorsqu'un bit y est mis à un, le registre correspondant est utilisé.

Pour la prédécrémentation, ce masque est le suivant :



Pour les autres cas :



Code	Mnémonique
0022 A9	ORL.B #SA9-(A2)
003C 00B5	ORI #SB5,CCR
0062 EF16	ORL.W #EF16-(A2)
007C 15C0	ORI #S15C0,SR
009C FFFF 8604	ORL.L #SFFFF8604,(A4)+
0222 A9	ANDL.B #SA9,-(A2)
023C 00B5	ANDI #SB5,CCR
0262 EF16	ANDL.W #SEF16,-(A2)
027C C015	ANDI #SC015,SR
029C FFFF 8604	ANDL.L #SFFFF8604,(A4)+
041C A9	SUBL.B #SA9,(A4)+
045C 16AC	SUBL.W #S16AC,(A4)+
04A2 E0E0 6840	SUBL.L #SE0E06840,-(A2)
05C4	BSET D2,D4
05CC 0100	MOVEP.L D2,\$100(A4)
061C A9	ADDL.B #SA9,(A4)+
065C EF16	ADDL.W #SEF16,(A4)+
06A2 FFFF 8604	ADDL.L #SFFFF8604,-(A2)
070C 01AA	MOVEP S1AA(A4),D3
0712	BTST D3,(A2)
071A	BTST D3,(A2)+
0752	BCHG D3,(A2)
075C	BCHG D3,(A4)+
078C 0020	MOVEP.D3,\$20(A4)

LANGAGE MACHINE

**TABLE DES CODES OPERATOIRES
TRIES PAR CODE**

<i>Code</i>	<i>Mnémonique</i>
07F8 2FFF	BSET D3,S2FFF
0802 0004	BTST #4,D2
0852 0002	BCHG #2,(A2)
085C 0000	BCHG #0,(A4)+
0892 0002	BCLR #2,(A2)
08A3 0000	BCLR #0,-(A3)
08C3 0002	BSET #2,D3
08DC 0004	BSET #4,(A4)+
094A 0003	MOVEP.L \$3(A2),D4
0982	BCLR D4,D2
098A 0300	MOVEP D4,\$300(A2)
09AA 0300	BCLR D4,S300(A2)
09CA 00FF	MOVEP.L D4,SFF(A2)
0A03 0016	EORL.B #S16,D3
0A3C 00F5	EORI #SF5,CCR
0A44 5555	EORL.W #S5555,D4
0A7C A0A0	EORI #SA0A0,SR
0A9C AAAA CCCC	EORL.L #SAAAACCCC,(A4)+
0C03 0016	CMPL.B #S16,D3
0C43 AAAA	CMPL.W #SAAAA,D3
0CAC 0200 FFFF A316	CMPL.L #SFFFFA316,\$200(A4)
1413	MOVE.B (A3),D2
21F1 0002 2000	MOVE.L 2(A1,D0),\$2000
2844	MOVEA.L D4,A4
36DC	MOVE.V (A4)+,(A3)+
3868 0030	MOVEA.W \$30(A0),A4
4022	NEGX.B -(A2)
4043	NEGX.W D3
40B8 7F00	NEGX.L SF00
40DB	MOVE SR,(A3)+
40F8 1000	MOVE SR,S1000
41B8 2F16	CHK S2F16,D0
4238 2000	CLR.B \$2000
4284	CLR.L D4
43F8 3000	LEA \$3000,A1
4438 1000	NEG.B \$1000
4453	NEG.W (A3)
44A8 0020	NEG.L \$20(A0)
44C4	MOVE D4,CCR
44F8 3000	MOVE \$3000,CCR
4620	NOT.B -(A0)
4644	NOT.W D4
46B9 0003 2200	NOT.L \$032200
46C3	MOVE D3,SR
46E8	MOVE S10(A0),SR

TABLE DES CODES OPERATOIRES
TRIES PAR CODE

<i>Code</i>	<i>Mnémonique</i>
47AA 0300	CHK \$300(A2),D3
481C	NBCD (A4)+
4830 1810	NBCD \$10(A0,D1)
4842	SWAP D2
4846	SWAP D6
4853	PEA (A3)
4878 2000	PEA \$2000
4883	EXT.B D3
4893 8808	MOVEM D0/D4/A4,(A3)
48A0 88F0	MOVEM D0/D4/A0-A3,-(A0)
48C3	EXT.W D3
48C7	EXT.W D7
48E0 88F0	MOVEM.L D0/D4/A0-A3,-(A0)
49EA 0200	LEA \$200(A2),A4
4A03	TST.B D3
4A78 2000	TST.W \$2000
4AC3	TAS D3
4ADA	TST.L (A2)+
4AEE 001A	TAS \$1A(A6)
4AFC	ILLEGAL
4C9D 1FFF	MOVEM (A5)+,A0-A4/D0-D7
4CB8 0103 3000	MOVEM \$3000,A0/D0/D1
4E44	TRAP #4
4E4C	TRAP #12
4E53 1000	LINK A3,#\$1000
4E54 1000	LINK A4,#\$1000
4E68	UNLK A0
4E6B	UNLK A3
4E70	RESET
4E71	NOP
4E72 F0F0	STOP #F0F0
4E73	RTE
4E75	RTS
4E76	TRAPV
4E77	RTR
4EA4	MOVE A4,USP
4EAB	MOVE USP,A3
4EB0 1804	JSR \$4(A0,D1)
4EB8	JSR \$0100
4ED3	JMP (A3)
4EF8 2000	JMP \$2000
5000 2F15	BRA \$2F15
501A	BRA \$1A
5210	ADDQ.B #1,(A0)
5310	SUBQ.B #1,(A0)

**TABLE DES CODES OPERATOIRES
TRIES PAR CODE**

<i>Code</i>	<i>Mnémonique</i>
545A	ADDQ.W #2,(A2)+
555A	SUBQ.W #2,(A2)+
56CB 2F19	DBNE D3,\$2F19
56D4	SNE (A4)
5BCD 0100	DBMI D5,\$100
5BEA	SMT S4F(A2)
5E9A	ADDQ.L #7,(A2)+
5FCB 0030	DBGT D3,\$30
5F9A	SUBQ.L #7,(A2)+
6100 1A09	BSR S1A09
612C	BSR S2C
6700 8948	BEQ \$8948
673E	BEQ \$3E
6F00 2000	BLE \$2000
72A1	MOVEQ #1A,D1
7606	MOVEQ #6,D3
8100	SBCD D0,D0
81D3	DIVS (A3),D0
81EB 0100	DIVS \$100(A3),D0
829B	OR.L (A3)+,D1
8300	SBCD D0,D1
839B	OR.L D1,(A3)+
8504	SBCD D4,D2
87E2	DIVS -(A2),D3
8810	OR.B (A0),D4
8850	OR.W (A0),D4
88F1 0002	DIVU.2 (A1,D0),D4
8908	SBCD -(A0),-(A4)
9810	OR.B D4,(A0)
8950	OR.W D4,(A0)
8B08	SBCD -(A0),-(A5)
810C	SBCD -(A4),-(A6)
8EDA	DIVU (A2)+,D7
9424	SUB.B -(A4),D2
9464	SUB.W -(A4),D2
951C	SUB.B D2,(A4)+
955C	SUB.W D2,(A4)+
96E4	SUBA.W -(A4),A3
9781	SUBX.L D1,D3
9789	SUBX.L -(A1),-(A3)
97E4	SUBA.L -(A4),A3
98A6	SUB.L -(A6),D4
9902	SUBX.B D2,D4
990A	SUBX.B -(A2),-(A4)
9942	SUBX.W D2,D4

TABLE DES CODES OPERATOIRES
TRIES PAR CODE

<i>Code</i>	<i>Mnémonique</i>
994A	SUBX.W -(A2),-(A4)
999E	SUB.L D4,(A6)+
B052	CMP (A2),D0
B46C 0100	CMP \$100(A4),D2
B5F8 1000	CMPA.L \$1000,A2
B738 2000	EOR.B D3,\$2000
B745	EOR.W D3,D5
B8DC	CMPA.W (A2)+,A4
B90B	CMPM.B (A3)+,(A4)+
B94B	CMPM.W (A3)+,(A4)+
B94E	CMPM.L (A6)+,(A4)+
B9AD 0100	EOR.L D4,\$100(A5)
C100	ABCD D0,D0
C29B	AND.L (A3)+,D1
C300	ABCD D0,D1
C39B	AND.L D1,(A3)+
C4C4	MULU D4,D2
C4F8 2000	MULU \$2000,D2
C504	ABCD D4,D2
C544	EXG D2,D4
C54C	EXG A2,A4
C5C4	MULS D4,D2
C78E	EXG D3,A6
C7F8 3000	MULS \$3000,D3
C810	AND.B (A0),D4
C850	AND.W (A0),D4
C908	ABCD -(A0),-(A4)
C910	AND.B D4,(A0)
C950	AND.W D4,(A0)
CB08	ABCD -(A0),-(A5)
CD0C	ABCD -(A4),-(A6)
D424	ADD.B -(A4),D2
D464	ADD.W -(A4),D2
D51C	ADD.B D2,(A4)+
D55C	ADD.W D2,(A4)+
D6E4	ADDA.W -(A4),A3
D781	ADDX.L D1,D3
D789	ADDX.L -(A1),-(A3)
D7F4	ADDA.L -(A4),A3
D8A6	ADD.L -(A6),D4
D902	ADDX.B D2,D4
D90A	ADDX.B -(A2),-(A4)
D942	ADDX.W D2,D4
D94A	ADDX.W -(A2),-(A4)
D99E	ADD.L D4,(A6)+

**TABLE DES CODES OPERATOIRES
TRIES PAR CODE**

<i>Code</i>	<i>Mnémonique</i>
E0E4	ASR -(A4)
F0E8 0300	ASR \$300(A0)
E1D6	ASL (A6)
E1F8 2000	ASL \$2000
E2E4	LSR -(A4)
E2E8 0300	LSR \$300(A0)
E3D6	LSL (A6)
E3F8 2000	LSL \$2000
E424	ASR.B D2,D4
E42C	LSR.B D2,D4
E43C	ROR.B D2,D4
E47C	ROR.W D2,D4
E4D6	ROXR (A6)
E524	ASL.B D2,D4
E52C	LSL.B D2,D4
E53C	ROL.B D2,D4
E564	ASL.W D2,D4
E56C	LSL.W D2,D4
E574	ROXL.W D2,D4
E57C	ROL.W D2,D4
E5C4	ROXL.B #3,D4
E5F8 7F00	ROXL \$7F00
E604	ASR.B #3,D4
E60C	LSR.B #3,D4
E61C	ROR.B #3,D4
E644	ASR.W #3,D4
E64C	LSR.W #3,D4
E6A6	ASR.L D3,D6
E6AE	LSR.L D3,D6
E6B6	ROXR.L D3,D6
E6BE	ROR.L D3,D6
E6D6	ROR (A6)
E6F8 2000	ROR \$2000
E704	ASL.B #3,D4
E70C	LSL.B #3,D4
E71C	ROL.B #3,D4
E744	ASL.W #3,D4
E74C	LSL.W #3,D4
E7A6	ASL.L D3,D6
E7AE	LSL.L D3,D6
E7BE	ROL.L D3,D6
E7D6	ROL (A6)
E7F8 2000	ROL \$2000
E866	ASR.W D4,D6
E86E	LSR.W D4,D6

TABLE DES CODES OPERATOIRES TRIES PAR CODE

<i>Code</i>	<i>Mnémonique</i>
EA56	ROXR.W #5,D6
EA5E	ROR.W #5,D6
EA86	ASR.W #5,D6
EA8E	LSR.L #5,D6
EA9E	ROR.L #5,D6
EB5E	ROL.W #5,D6
EB86	ASL.L #5,D6
EB8E	LSL.L #5,D6
EB9E	ROL.L #5,D6

TABLE DES CODES OPERATOIRES TRIES PAR MNEMONIQUE

Les remarques et les considérations faites lors de l'élaboration de la table précédente valent également pour la table suivante.

<i>Mnémonique</i>	<i>Code</i>
ABCD D0,D0	C100
ABCD D0,D1	C300
ABCD D4,D2	C504
ABCD -(A0),-(A4)	C908
ABCD -(A0),-(A5)	CB08
ABCD -(A4),-(A6)	CD0C
ADD.B D2,(A4)+	D51C
ADD.W D2,(A4)+	D55C
ADD.L D4,(A6)+	D99E
ADD.B -(A4),D2	D424
ADD.W -(A4),D2	D464
ADD.L -(A6),D4	D8A6
ADDA.W -(A4),A3	D6E4
ADDA.L -(A4),A3	D7E4
ADDI.B #SA9,(A4)+	061C A9
ADDI.W #SEF16,(A4)+	065C EF16
ADDI.L #FFFF8604,-(A2)	06A2 FFFF 8604
ADDQ.B #1,(A0)	5210
ADDQ.W #2,(A2)+	545A
ADDQ.L #7,(A2)+	5E9A
ADDX.B D2,D4	D902
ADDX.W D2,D4	D942
ADDX.L D1,D3	D781
ADDX.B -(A2),-(A4)	D90A
ADDX.W -(A2),-(A4)	D94A
ADDX.L -(A1),-(A3)	D789
AND.B D4,(A0)	C910
AND.W D4,(A0)	C950
AND.L D1,(A3)+	C39B
AND.B (A0),D4	C810
AND.W (A0),D4	C850
AND.L (A3)+,D1	C29B
ANDI.B #SA9,-(A2)	0222 A9
ANDI.W #SEF16,-(A2)	0262 EF16
ANDI.L #FFFF8604,(A4)+	029C FFFF 8604
ANDI #B5,CCR	023C 00B5
ANDI #C015,SR	027C C015
ASL.B D2,D4	E524
ASL.W D2,D4	E564
ASL.L D3,D6	E7A6
ASL.B #3,D4	E704

TABLE DES CODES OPERATOIRES
TRIES PAR MNEMONIQUE

<i>Mnémonique</i>	<i>Code</i>
ASL.W #3,D4	E744
ASL.L #5,D6	EB86
ASL (A6)	E1D6
ASL \$2000	E1F8 2000
ASR.B D2,D4	E424
ASR.W D4,D6	E866
ASR.L D3,D6	E6A6
ASR.B #3.D4	E604
ASR.W #3,D4	E644
ASR.L #5,D6	EA86
ASR -(A4)	E0E4
ASR #300(A0)	E0E8 0300
BEQ \$3E	673E
BEQ \$8948	6700 8948
BLE \$2000	6F00 2000
BCIG D3,(A2)	0752
BCHG D3,(A4)+	075C
BCHG #0,(A4)+	085C 0000
BCIG #2,(A2)	0852 0002
BCLR D4,\$300(A2)	09AA 0300
BCLR D4,D2	0982
BCLR #0,-(A3)	08A3 0000
BCLR #2,(A2)	0892 0002
BRA \$1A	501A
BRA \$2F15	5000 2F15
BSET D2,D4	05C4
BSET D3,\$2FFF	07F8 2FFF
BSET #4,(A4)+	08DC 0004
BSET #2,D3	08C3 0002
BSR \$2C	612C
BSR \$1A09	6100 1A09
BTST D3,(A2)	0712
BTST D3,(A2)+	071A
BTST #4,D2	0802 0004
CHK \$300(A2),D3	47AA 0300
CHK \$2F16,D0	41B8 2F16
CLR.L D4	4284
CLR.B \$2000	4238 2000
CMP (A2),D0	B052
CMP \$100(A4),D2	B46C 0100
CMPA.W (A2)+,A4	B8DC
CMPA.L \$1000,A2	B5F8 1000
CMPI.B # \$16,D3	0C03 0016
CMPI.W # \$AAAA,D3	0C43 AAAA
CMPI.L # \$FFFA316,\$200(A4)	0CAC 0200 FFFF A316

LANGUAGE MACHINE

TABLE DES CODES OPERATOIRES TRIES PAR MNEMONIQUE

<i>Mnémonique</i>	<i>Code</i>
CMPM.B (A3)+,(A4)+	B90B
CMPM.W (A3)+,(A4)+	B94B
CMPM.L (A6)+,(A4)+	B94E
DBNE D3,\$2F19	56CB 2F19
DBGT D3,\$30	5ECB 0030
DBMI D5,\$100	5BCD 0100
DIVS (A3),D0	81D3
DIVS \$100(A3),D0	81EB 0100
DIVS -(A2),D3	87E2
DIVU 2(A1,D0),D4	88F1 0002
DIVU (A2)+,D7	8EDA
EOR.B D3,\$2000	B738 2000
EOR.W D3,D5	B745
EOR.L D4,\$100(A5)	B9AD 0100
EORLB #S16,D3	0A03 16
EORLW #S5555,D4	0A44 5555
EORLL #SAAAACCCC,(A4)+	0A9C AAAA CCCC
EORI #S\$F5,CCR	0A3C 00F5
EORI #S A0A0,SR	0A7C A0A0
EXG D2,D4	C544
EXG A2,A4	C54C
EXG D3,A6	C78E
EXT.B D3	4883
EXT.W D3	48C3
EXT.W D7	48C7
'ILLEGAL'	4AFC
JMP (A3)	4ED3
JMP \$2000	4EF8 2000
JSR \$4(A0,D1)	4EB0 1804
JSR \$0100	4EB8
LEA \$3000.A1	43F8 3000
LEA \$200(A2),A4	49EA 0200
LINK A4,#S1000	4E54 1000
LINK A3,#SFOF0	4E53 F0F0
LSL.B D2,D4	E52C
LSL.W D2,D4	E56C
LSL.L D3,D6	E7AE
LSL.B #3,D4	E70C
LSL.W #3,D4	E74C
LSL.L #5,D6	EB8E
LSL (A6)	E3D6
LSL \$2000	E3F8 2000
LSR.B D2,D4	E42C
LSR.W D4,D6	E86E
LSR.L D3,D6	E6AE

TABLE DES CODES OPERATOIRES TRIES PAR MNEMONIQUE

<i>Mnémonique</i>	<i>Code</i>
LSR.B #3,D4	E60C
LSR.W #3,D4	E64C
LSR.L #5,D6	EA8E
LSR -(A4)	E2E4
LSR S300(A0)	E2E8 0300
MOVE.B (A3),D2	1413
MOVE.W (A4)+,(A3)+	36DC
MOVE.L 2(A1,D0),S2000	21F1 0002 2000
MOVE D4,CCR	44C4
MOVE \$3000,CCR	44F8 3000
MOVEA.W S30(A0),A4	3868 0030
MOVEA.L D4,A4	2844
MOVE USP,A3	4EAB
MOVE A4,USP	4EA4
MOVE SR,(A3)+	40DB
MOVE SR,\$1000	40F8 1000
MOVE \$10(A0),SR	46E8
MOVE D3,SR	46C3
MOVEM D0/D4/A0-A3,-(A0)	48A0 88F0
MOVEM.L D0/D4/A0-A3,-(A0)	48E0 88F0
MOVEM D0/D4/A4,(A3)	4893 8808
MOVEM (A5)+,A0-A4/D0-D7	4C9D 1FFF
MOVEM S3000,A0/D0/D1	4CB8 0103 3000
MOVEP D3,\$20(A4)	078C 0020
MOVEP D4,\$300(A2)	098A 0300
MOVEP.L D2,\$100(A4)	05CC 0100
MOVEP.L D4,\$FF(A2)	09CA 00FF
MOVEP \$1AA(A4),D3	070C 01AA
MOVEP.L S3(A2),D4	094A 0003
MOVEQ #6,D3	7606
MOVEQ #1A,D1	721A
MULS D4,D2	C5C4
MULS S3000,D3	C7F8 3000
MULU D4,D2	C4C4
MULU \$2000,D2	C4F8 2000
NBCD (A4)+	481C
NBCD \$10(A0,D1)	4830 1810
NEG.B \$1000	4438 1000
NEG.W (A3)	4453
NEG.L \$20(A0)	44A8 0020
NEGX.B -(A2)	4022
NEGX.W D3	4043
NEGX.L S7F00	40B8 7F00
NOP	4E71
NOT.B -(A0)	4620
NOT.W D4	4644

TABLE DES CODES OPERATOIRES TRIES PAR MNEMONIQUE

<i>Mnémonique</i>	<i>Code</i>
NOT.L \$032200	46B9 0003 2200
OR.B D1,(A0)	8910
OR.W D4,(A0)	8950
OR.L D1,(A3)+	839B
OR.B (A0),D4	8810
OR.W (A0),D4	8850
OR.L (A3)+,D1	829B
OR.LB #5A9,-(A2)	0022 A9
OR.LW #5EF16,-(A2)	0062 EF16
OR.LL #5FFFF8604,(A4)+	009C FFFF 8604
ORI #5B5,CCR	003C 00B5
ORI #515C0.SR	007C 15C0
PEA (A3)	4853
PEA \$2000	4878 2000
RESET	4E70
ROL.B D2,D4	E53C
ROL.W D2,D4	E57C
ROL.L D3,D6	E7BE
ROL.B #3,D4	E71C
ROL.W #5,D6	EB5E
ROL.L #5,D6	EB9E
ROL (A6)	E71D6
ROL \$2000	E7F8 2000
ROR.B D2,D4	E43C
ROR.W D2,D4	E47C
ROR.L D3,D6	E6BE
ROR.B #3,D4	E61C
ROR.W #5,D6	E45E
ROR.L #5,D6	E49E
ROR (A6)	E6D6
ROR \$2000	E6F8 2000
ROXL.W D2,D4	E574
ROXL.B #3,D4	E5C4
ROXL \$7F00	E5F8 7F00
ROXR.L D3,D6	E6B6
ROXR.W #5,D6	E456
ROXR (A6)	E4D6
RTE	4E73
RTR	4E77
RTS	4E75
SBCD D0,D0	8100
SBCD D0,D1	8300
SBCD D4,D2	8504
SBCD -(A0),-(A4)	8908
SBCD -(A0),-(A5)	8B08

TABLE DES CODES OPERATOIRES
TRIES PAR MNEMONIQUE

<i>Mnémonique</i>	<i>Code</i>
SBCD -(A4),-(A6)	8D0C
SNE (A4)	56D4
SMI S4F(A2)	5BEA
STOP #F0F0	4E72 F0F0
SUB.B D2,(A4)+	951C
SUB.W D2,(A4)+	955C
SUB.L D4,(A6)+	999E
SUB.B -(A4),D2	9424
SUB.W -(A4),D2	9464
SUB.L -(A6),D4	98A6
SUBA.W -(A4),A3	96E4
SUBA.L -(A4),A3	97E4
SUBI.B #S49,(A4)+	041C A9
SUBI.W #S16AC,(A4)+	045C 16AC
SUBI.L #SE0E06840,-(A2)	04A2 E0E0 6840
SUBQ.B #1,(A0)	5310
SUBQ.W #2,(A2)+	555A
SUBQ.L #7,(A2)+	5F9A
SUBX.B D2,D4	9902
SUBX.W D2,D4	9942
SUBX.L D1,D3	9781
SUBX.B -(A2),-(A4)	990A
SUBX.W -(A2),-(A4)	994A
SUBX.L -(A1),-(A3)	9789
SWAP D2	4842
SWAP D6	4846
TAS D3	4AC3
TAS S1A (A6)	4AEE 001A
TRAP #4	4E44
TRAP #12	4E4C
TRAPV	4E76
TST.B D3	4A03
TST.W S2000	4A78 2000
TST.L (A2)+	4ADA
UNLK A3	4E6B
UNLK A0	4E68

LANGUAGE MACHINE

GENERALITES

Le GEMDOS est le système d'exploitation standard de l'ATARI ST. Il est aussi appelé TOS (TRAMIEL OPERATING SYSTEM). Jack Tramiel étant le nom du PDG d'ATARI.

A l'origine, le premier système d'exploitation du ST était le CP/M 68000 de Digital Research. Cependant, la relative lenteur du CP/M 68000 et la difficulté d'interfaçage total avec le GEM (Graphic Environment Manager) ont forcé Digital Research à produire un nouveau système d'exploitation.

Le GEMDOS est donc né. Il reprend bien des caractéristiques de son grand-père le CP/M 68000. Cependant, on lui a ajouté une série de fonctions, comme la gestion des sous-répertoires, qui le rapproche encore plus du MS-DOS, système d'exploitation standard des IBM/PC et compatibles.

Le GEMDOS est donc un système d'exploitation mono-utilisateur et mono tâche spécialement construit pour être utilisé avec le processeur Motorola MC68000 et être interfacé avec le GEM.

Il nécessite un minimum de 128K de mémoire centrale, 160K de mémoire morte (ROM), un écran dont la VIDEORAM se trouve en mémoire centrale (MEMORY MAP) d'une résolution minimum de 320 x 200 points et une souris.

Cette configuration constitue cependant un strict minimum. En effet, pour pouvoir utiliser au mieux les possibilités du GEMDOS, il faut une mémoire centrale de 256 K minimum, 192 K de mémoire morte, un écran d'une résolution de 640 x 400 points, une souris, deux disques souples, une imprimante et un port de communication. On retrouve bien sûr tout cela et bien d'autres choses sur le ST.

GENERALITES

Le GEMDOS est utilisable de deux façons :

- avec la couche interface graphique GEM qui permet d'effectuer les différentes fonctions par l'intermédiaire de la manipulation d'un bureau et d'icônes ; cette méthode sera décrite dans le tome 2 des "Clefs pour Atari ST" ;
- de façon classique en utilisant des commandes semblables à celles de tous les systèmes d'exploitation ; ces commandes seront décrites en détails dans la suite de ce chapitre.

Ce chapitre décrit aussi l'architecture interne de GEMDOS ainsi que toutes les tables et tous les appels de fonctions internes du système quel que soit leur mode de fonctionnement (GEM ou standard).

SPECIFICATIONS DES FICHIERS DU GEMDOS

Nom de fichier

Un nom de fichier du GEMDOS se compose de quatre champs :

- un nom de disque sur une lettre (A à P) suivie d'un ':' ;
- un chemin d'accès à travers les répertoires dans lequel chaque nom de répertoire est séparé par un caractère '\'
- un nom de fichier composé de 1 à 8 caractères alphanumériques ;
- un type ou extension de fichier optionnel de 1 à 3 caractères ; le type est séparé du nom par un caractère '.'.

Schéma : d:\chemin\nom.ext

Exemple :

A:\BASIC\STOCK\SAISIE.PRG

Définit le fichier de nom SAISIE avec une extension PRG qui se trouve sur le disque A dans le sous-répertoire STOCK du sous-répertoire Basic du répertoire principal.

Les caractères suivants sont interdits dans un nom de fichier ou dans un nom de sous-répertoire :

[] () < > = * & , ! ? / \ . ; + -

Carte de sélection

Les caractères ? et * permettent d'introduire une ambiguïté dans le nom de fichier :

? représente n'importe quel caractère (un seul).

* représente n'importe quelle suite de caractères.

L'utilisation de ? et * forme une carte de sélection (WILD CART). Certaines commandes GEMDOS acceptent les noms de fichier ambigus.

Exemples :

- XYZ.PRG représente le fichier unique XYZ.PRG ;
- X?Z.PRG représente tous les fichiers d'extension PRG dont le nom de fichier est composé de trois lettres et dont la première est X et la troisième Z (par exemple : XAZ.PRG ou XYZ.PRG ou encore X3Z.PRG) ;
- *.PRG représente tous les fichiers d'extension .PRG ;
- I*.PRG représente tous les fichiers dont l'extension est PRG et dont le nom commence par I. (par exemple : I.PRG, LUBIE.PRG ou encore LUNE.PRG) ;
- DUMP.* représente tous les fichiers dont le nom est DUMP, quelle que soit leur extension ;
- *.* représente tous les fichiers.

PRINCIPAUX TYPES DE FICHIERS

En général, l'extension du nom du fichier constituée de trois caractères identifie le type ou la classe du fichier considéré. Les lignes suivantes décrivent les principaux types de fichiers que l'on peut rencontrer.

- ACC** Fichiers accessibles par le bureau.
- ASM** Fichiers contenant des programmes écrits en assembleur prêts à être assemblés par le macro-assembleur.
- BAS** Fichiers contenant des programmes pour l'interpréteur Basic.
- BAT** Fichiers contenant une séquence d'instructions destinées à un traitement BATCH.
- C** Fichiers contenant des programmes source écrits en langage C.
- DEF** Fichiers contenant des définitions de dessins.
- DOC** Fichiers contenant la documentation d'un produit.
- FIL** Fichiers contenant une image écran.
- HLP** Fichiers contenant une aide à l'utilisateur.
- LIB** Fichiers contenant une bibliothèque ou une librairie de programmes.
- LOG** Fichiers source écrits en Logo.
- OBJ** Fichiers OBJET qui sont le résultat de l'assemblage par le macro-assembleur.
- OVR** Fichiers d'OVERLAY. Les OVERLAY sont des extensions d'un programme principal.
- PAS** Fichiers contenant des programmes Pascal.
- PRG** Fichiers contenant un programme exécutable. Ces fichiers ne contiennent que le code exécutable et les données utiles.
- RSC** Fichiers Resource Construction Set.
- TMP** Fichiers créés temporairement pour gérer les fichiers trop grands.
- TOS** Fichiers exécutables hors de GEM.

COMMANDES INTERNES

Les commandes internes font partie intégrante de l'interpréteur de commandes **COMMAND.TOS**. Elles sont chargées en mémoire en même temps que le fichier **COMMAND.TOS** et sont exécutées automatiquement.

Syntaxe des commandes

Les règles suivantes sont utilisées pour la syntaxe des commandes :

- les mots en majuscules doivent être introduits tels quels ; ils constituent les mots clés ;
- les éléments entre les signes "<>" sont obligatoirement associés aux mots clés ;
- les paramètres entre crochets "[]" sont facultatifs ;
- un slash "/" séparant des éléments indique que l'on peut choisir entre ces éléments.

BREAK

Rôle : vérification de la fonction "CTRL-C".

Syntaxe : BREAK ON / OFF

Commentaires : BREAK ON vérifie la demande de CTRL-C chaque fois qu'une application fait appel au DOS. BREAK OFF ne teste la fonction que pendant des opérations effectuées au clavier, à l'imprimante, à l'écran ou sur d'autres périphériques.

CAT

Rôle : identique à TYPE (voir TYPE).

CD

Rôle : changement de répertoire ou affichage du chemin d'accès au répertoire courant.

Syntaxe : CD [unité:] [chemin d'accès]

Commentaires : si l'unité n'est pas spécifiée, c'est l'unité par défaut qui est utilisée. Le répertoire courant est celui où le GEMDOS cherche les fichiers dont le répertoire n'est pas précisé.

La barre oblique (\) indique le répertoire principal.

Les deux-points (:) indiquent le répertoire de niveau juste supérieur.

COMMANDES INTERNES

La commande sans paramètre fournit le chemin d'accès au répertoire en cours.

CHMOD

Rôle : produire un changement de mode.

Syntaxe : CHMOD [unité:] [chemin d'accès] <nomfichier> n

Commentaires : n représente un nombre compris entre 0 et 7.

CLS

Rôle : efface l'écran.

Syntaxe : CLS

Commentaires : la commande CLS consiste à envoyer une séquence de caractères de contrôle qui efface l'affichage de l'écran.

COPY

Rôle : copie un fichier vers un autre fichier.

Syntaxe : COPY [unité:] [chemin d'accès] <nomfichier1>
[unité:] [chemin d'accès] <nomfichier2>

Commentaires : l'instruction COPY copie le fichier de nom nf1 vers le fichier de nom nf2.

ATTENTION une erreur dans l'interpréteur de commandes implique que la copie d'un fichier sous le même nom détruit ce fichier au lieu de produire le message d'erreur prévu à cet effet.

DEL

Rôle : effacer le fichier spécifié.

Syntaxe : DEL [unité:] [chemin d'accès] <nom du fichier>

Commentaires : les fichiers système et les fichiers utilisables en lecture seulement (READ ONLY) ne peuvent être effacés.

DIR

Rôle : lister les fichiers ou les sous-répertoires d'un répertoire.

Syntaxe : DIR [unité:] [chemin d'accès] [carte de sélection] [/F]
[/D] [/T]

Commentaires : la commande sans paramètre affiche les entrées du répertoire qui correspondent à la carte de sélection ou toutes les entrées si celle-ci n'existe pas. L'option /F limite l'édition aux fichiers seuls, l'option /D limite l'édition aux sous-répertoires seuls, l'option /T limite l'édition au nom seul sans les indications de date et de taille.

ECHO

Rôle : affichage d'une chaîne sur écran.

Syntaxe : ECHO "message"

Commentaires : le message entre guillemets est affiché sur écran. Les guillemets ne sont pas obligatoires.

ENV

Rôle : affiche l'environnement.

Syntaxe : ENV

Commentaires : cette commande affiche le chemin de recherche (PATH) s'il existe et l'environnement par défaut du disque d'initialisation.

ERA

Rôle : Identique à DEL.

EXIT

Rôle : sortir de l'interpréteur de commandes et retourner au niveau supérieur.

Syntaxe : EXIT

Commentaires : cette commande permet de revenir au bureau.

GET

Rôle : devrait servir à lire certaines informations sur disque. Devrait être utilisé conjointement à PUT. Ne fonctionne pas correctement.

COMMANDES INTERNES

HELP

Rôle : fournir une liste des principales commandes.

Syntaxe : HELP

INIT

Rôle : initialise la F.A.T.

Syntaxe : INIT <unité>

Commentaires : cette commande efface irrémédiablement tout ce qui se trouve sur le disque.

LS

Rôle : identique à DIR.

MD

Rôle : créer un nouveau répertoire.

Syntaxe : MD [unité:] <chemin d'accès>

Commentaires : l'utilisation de cette commande permet la création d'une structure en arbre. Comme la capacité des disques ne cesse de croître, cette hiérarchie est très intéressante.

MOVE

Rôle : déplace une entrée dans le répertoire.

Syntaxe: MOVE [unité:] [chemin d'accès] <nomfichier1> [unité:]
[chemin d'accès] <nomfichier2>

Commentaires : cette commande permet de déplacer une entrée d'un sous-répertoire à un autre.

NOWRAP

Rôle : empêche la taille de l'édition d'un texte de dépasser une ligne. Les caractères dépassant la fin de la ligne s'inscriront dans la dernière colonne.

Syntaxe : NOWRAP

COMMANDES INTERNES

Commentaires : cette commande produit la séquence ESC w.

PATH

Rôle : établir un chemin d'accès de recherche.

Syntaxe : PATH [(unité:) <chemin d'accès> [(unité:) chemin]...]

Commentaires : PATH demande au DOS de chercher les commandes externes qu'il ne trouve pas dans le répertoire courant dans les répertoires précisés.

PATH introduit sans paramètre affiche le répertoire courant.

PATH suivi d'un point-virgule n'autorise la recherche que dans le répertoire courant.

PAUSE

Rôle : interrompre le programme en cours pendant un instant.

Syntaxe : PAUSE "message"

Commentaires : le programme est interrompu par cette commande et un message peut être affiché. L'exécution du fichier reprendra en appuyant sur une touche quelconque, à l'exception de CTRL-C.

PUT

Rôle : fonction qui présente l'inconvénient de planter le système. A éviter.

PUTBOOT

Rôle : devrait servir à générer un BOOT sur le disque. Hélas, elle plante le système.

RD

Rôle : supprimer un répertoire.

Syntaxe : RD [unité:] [chemin d'accès] <répertoire>

Commentaires : le répertoire à effacer doit être vide. Le dernier répertoire cité est celui à supprimer. Le répertoire principal (TRONC) et le répertoire courant ne peuvent pas être supprimés.

COMMANDES INTERNES

REM

Rôle : identique à ECHO.

REN

Rôle : modifier le nom d'un fichier.

Syntaxe : REN [unité:] [chemin d'accès] <nomfichier1> <nomfichier2>

Commentaires : cette commande remplace le nom du premier fichier par le nom du second. Celui-ci reste dans le même répertoire.

RM

Rôle : identique à DEL.

SHOW

Rôle : afficher l'état d'occupation d'un disque.

Syntaxe : SHOW [unité:]

Commentaires : cette commande fournit le nombre de CLUSTERS sur le disque, le nombre de CLUSTERS libres, le nombre de secteurs par CLUSTER et le nombre d'octets par secteur.

VERSION

Rôle : affiche le numéro de version.

Syntaxe : VERSION

Commentaires : le numéro se compose d'un chiffre indiquant la version suivi d'un point et de deux chiffres qui correspondent au numéro de la révision.

WRAP

Rôle : entraîne la création d'une nouvelle ligne sur laquelle se produit l'édition d'un texte lorsque celle-ci dépasse la longueur d'une ligne.

Syntaxe : WRAP

Commentaires : cette commande envoie la séquence ESC v.

PROGRAMMES BATCH

Les programmes BATCH sont des suites de commandes exécutées les unes après les autres par l'interpréteur de commandes.

Ces programmes sont avantageux car ils permettent d'exécuter une série de commandes au moyen d'une seule instruction.

Exemple

Le fichier DEMO.BAT contient :

```
ECHO introduisez la disquette B
PAUSE
DIR B:
PAUSE
SHOW B:
```

Il suffit de taper **DEMO** pour exécuter toutes ces commandes à la suite les unes des autres.

Les fichiers BATCH doivent contenir l'extension .BAT. Ils peuvent comporter des paramètres fictifs qui seront remplacés par les valeurs fournies lors de l'exécution du fichier.

Exemple

```
DEMO.BAT contient COPY %1.ASM %2.DBL
DEMO TOTO TATA copiera le fichier TOTO.ASM vers le fichier
TATA.DBL.
```

Les paramètres marqués par % seront remplacés successivement par les paramètres fournis à la suite du nom de la commande. Dix paramètres peuvent être utilisés.

Le paramètre %0 représente l'identificateur de commande (DEMO dans notre exemple).

S'il existe, le fichier AUTOEXEC.BAT est exécuté automatiquement au lancement de l'interpréteur de commande COMMAND.TOS.

MESSAGES D'ERREUR DE L'INTERPRETEUR DE COMMANDES

- (A)BORT, (R)ETRY OR (I)GNORE
Le système détecte une erreur lors de l'écriture ou de la lecture d'un périphérique. Il attend une des réponses suivantes :
 - A pour abandonner la commande ;
 - R pour tenter une nouvelle fois la commande ;
 - I pour ignorer l'erreur et continuer le programme.
- WILD CARDS NOT ALLOWED IN PATH NAME
Il est interdit d'utiliser une carte de sélection dans un chemin d'accès.
- FILE NOT FOUND
Le fichier spécifié n'existe pas.
- DESTINATION IS NOT A VALID WILD CARD EXPRESSION
La spécification de la carte de sélection de destination n'est pas correcte.
- COMMAND IS INCOMPLETELY SPECIFIED
La commande est incomplète.
- WILD CARD NOT ALLOWED IN DESTINATION
Une carte de sélection n'est pas admise dans la spécification de destination.
- UNABLE CHANGE MODE ON SUBDIRECTORY OR VOLUMES
La commande CHMOD n'est pas utilisable avec le répertoire principal ou un sous-répertoire.
- INVALID MODE SPECIFICATION
Spécification de la commande CHMOD invalide.
- COMPLETION CODE FOR PREVIOUS COMMAND
Lors de l'utilisation de la commande ERR, ce message précède le dernier code d'erreur produit.
- DIRECTORY NOT FOUND
Répertoire non trouvé.
- DISK FULL
Disque plein.
- COPY FAILED
La copie a échoué.
- COMMAND NOT FOUND
La commande n'est pas trouvée.

ARCHITECTURE INTERNE

Le GEMDOS est composé de trois modules principaux :

- le **CCP** (CONSOLE COMMAND PROCESSOR) interpréteur de commande ;
- le **BDOS** (BASIC DISK OPERATING SYSTEM) système de gestion résident traitant de la gestion générale des disques en dehors de l'interfaçage direct avec le matériel ;
- le **BIOS** (BASIC INPUT OUTPUT SYSTEM) système de gestion des entrées-sorties et des échanges entre le matériel et le logiciel. Il est toujours propre au type de machine utilisé.

En outre, le GEMDOS utilise un espace de travail appelé **TPA** (TRANSIENT PROGRAM AREA) qui est constitué de tout l'espace mémoire non occupé par les trois modules définis ci-dessus.

C'est dans cet espace que GEMDOS charge les programmes exécutables.

Le fichier **COMMAND.TOS** qui contient l'interpréteur de commandes est chargé lui aussi dans cet espace. C'est le seul fichier qui sera analysé dans ce tome. Les utilitaires disponibles en standard seront décrits dans le tome 2.

INITIALISATION DU SYSTEME

Au RESET, le 68000 lit la valeur initiale de sa pile superviseur à l'adresse 0 (4 octets) et la valeur initiale de son compteur programme à l'adresse 4 (4 octets). La valeur initiale du pointeur de pile est indéterminée car, à cet instant, le système ignore toujours la taille de la mémoire centrale.

Le processeur est placé en niveau de priorité d'interruption 7 et une instruction RESET est exécutée.

Un test est effectué pour détecter la présence d'une cartouche ROM de diagnostic (voir chapitre I).

Le mode de fonctionnement de la vidéo est initialisé (couleurs, résolution, synchronisation).

La taille de la mémoire est déterminée par un test destructif.

Le contrôleur MMU est initialisé.

Les variables du BIOS sont initialisées.

Les interruptions sont autorisées et le GEMDOS est initialisé.

Une tentative de lecture d'un secteur BOOT du disque souple ou, si la tentative échoue, du disque dur.

Le système exécute alors le fichier COMMAND.PRG s'il existe et si `_cmdload` est différent de zéro. Sinon, le BIOS construit un environnement par défaut et exécute le GEM.

Si le système n'est pas équipé du système en ROM, alors il existe une ROM BOOT qui contient les fonctions indispensables du BIOS pour gérer le disque souple.

L'initialisation du système est semblable à celle d'un système complet. Cependant, l'entièreté du GEMDOS est lue sur le disque.

Secteur BOOT

Le secteur BOOT contient :

- un numéro de série du volume ;
- un bloc de paramètres du BIOS ;
- un code BOOT optionnel et des paramètres optionnels.

Un secteur BOOT exécutable doit posséder un CHECKSUM (code de contrôle) égal à 1234 pour l'identification de la présence du BOOT. Pendant l'initialisation, le secteur BOOT est chargé dans un tampon mémoire temporaire. Si le CHECKSUM est correct, le système saute (JSR) au premier

INITIALISATION DU SYSTEME

octet du tampon. Comme le tampon peut se trouver n'importe où dans la mémoire, le BOOT doit être indépendant de l'adresse d'exécution.

Le chargeur

Le chargeur est constitué d'une suite d'octets optionnels qui se trouvent sur le secteur BOOT juste à la suite du BPB. Le chargeur a la possibilité de charger une suite de secteurs contigus ou un fichier "image".

Pour permettre à certains outils logiciels de manipuler les secteurs chargés par le chargeur, les six octets de la zone OEM du BPB doivent contenir 4C6F61646572 (Loader).

Un fichier "image" ne contient pas d'en-tête ni d'information de relocation mais simplement une image exacte du programme à exécuter. Le chargeur est capable de charger n'importe quel fichier du disque qu'il soit contigu ou non et quelle que soit sa position dans le répertoire.

Structure du chargeur

<i>Adr</i>	<i>Long</i>	<i>Nom</i>	<i>Fonction</i>
1E	2	EXECFLG	Ce mot est copié à l'adresse _cmdload. Si ce mot est différent de zéro, le fichier COMMAND.PRG est lancé à l'initialisation.
20	2	LDMODE	Si LDMODE vaut 0, un fichier est recherché et chargé. Si LDMODE est différent de zéro, SECTCNT secteurs sont chargés du disque en commençant au secteur SSECT.
22	2	SSECT	Numéro du secteur logique où doit commencer le chargement des secteurs (si LDMODE#0).
24	2	SECTCNT	Nombre de secteurs à charger (si LDMODE#0).
26	4	LDADDR	Adresse mémoire de chargement des secteurs ou du fichier.
2A	4	FATBUF	Adresse d'emplacement de la FAT et du répertoire.
2E	11	FNAME	Nom du fichier à charger (si LDMODE=0).
39	1	-----	Réservé.
3A			Début du code exécutable du BOOT.

INITIALISATION DU SYSTEME

BPB

Le BPB (BIOS PARAMETER BLOC) est une table de configuration qui se trouve au début du secteur BOOT.

Lors de l'appel d'un 'GET BPB', le BIOS lit le secteur BOOT et examine le BPB prototype. Un BPB est construit en partant du prototype. Si le prototype est bizarre, le BIOS retourne un NULL (zéro) comme indicateur d'erreur.

Le BPB est en principe calculé et écrit lors du formatage du disque.

Le BPB prototype est compatible logiquement (matériellement c'est autre chose) avec un BPB MSDOS version 2.x

Comme c'est le cas pour les processeurs INTEL et ZILOG (8080, 8088, 8086), le BPB est écrit avec les octets les moins significatifs des adresses 16 bits inversés.

Structure du BPB

<i>Adr</i>	<i>Lon</i>	<i>Nom</i>	<i>Contenu</i>
00	2	BRA.S	Adresse d'exécution du BOOT.
02	6	Filler	Réservé pour une configuration OEM. (OEM = Original Equipment Manufactured) Nom du constructeur.
08	3	SERIAL	24 bits contenant un numéro de série qui permet de détecter si le disque a été changé.
0B	2	BPS	Nombre d'octets par secteur (512 en standard).
0D	1	SPC	Nombre de secteurs par cluster (unité minimale d'allocation). Ce nombre vaut 2 en standard pour un disque souple.
0E	2	RES	Nombre de secteurs réservés au début du disque y compris le secteur boot (1 en standard). Permet de calculer la position de la première FAT.
10	1	NFAT	Nombre de FAT (FILE ALLOCATION TABLE) du disque. Dans le but d'augmenter la sécurité d'accès au fichier, le disque peut comporter une redondance au niveau de la table d'allocation des fichiers (FAT).

INITIALISATION DU SYSTEME

<i>Adr</i>	<i>Lon</i>	<i>Nom</i>	<i>Contenu</i>
11	2	NDIR	Nombre d'entrées du répertoire (DIRECTORY). Il existe une entrée par fichier, sous-répertoire ou volume du disque.
13	2	NSECTS	Nombre total de secteurs sur le disque.
15	1	MEDIA	Descripteur de média (ignoré du BIOS ST) : Bit0 = 1 = disque simple face Bit0 = 0 = disque double face Bit1 = 1 = 9 secteurs par piste Bit1 = 0 = 8 secteurs par piste.
16	2	SPF	Nombre de secteurs dans chaque FAT. Chaque CLUSTER occupant 1,5 octet de la FAT, ce nombre détermine le nombre de fichiers et leur taille.
18	2	SPT	Nombre de secteurs par piste (9 pour le disque souple).
1A	2	NSIDES	Nombre de faces du disque (1 ou 2 sur le disque souple).
1C	2	NHID	Nombre de secteurs cachés (ignoré du ST). Ce mot est intéressant lorsque l'on veut partitionner le disque dur pour former plusieurs zones manipulables comme des disques distincts.
De 1E à 1FD			on trouve le chargeur et le code exécutable du BOOT.
1FE	2	CHECKSUM	Ces deux octets permettent d'équilibrer le CHECKSUM pour obtenir 1234 (voir <i>BOOT</i>).

CCP ET PROGRAMMES EXTERNES

Après l'initialisation (démarrage à froid), le GEMDOS affiche son symbole de sollicitation (d) où d représente l'unité disque active. Le GEMDOS attend alors l'entrée d'une commande. L'utilisateur peut alors entrer le nom d'un programme exécutable, le nom d'un programme BATCH ou d'une commande interne du fichier COMMAND.TOS.

Les commandes externes (programmes exécutables) sont des programmes, en langage machine, chargés par GEMDOS en mémoire en vue d'être exécutés.

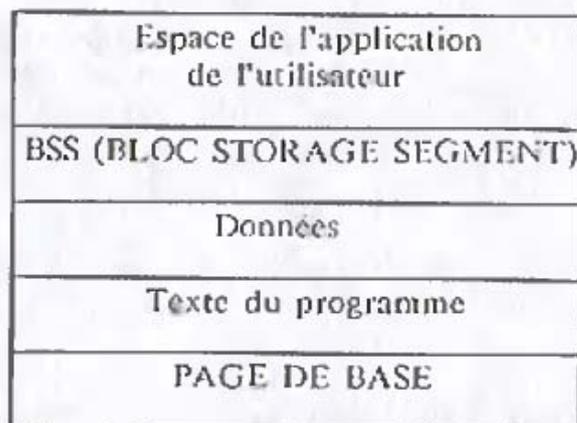
Lorsque l'utilisateur entre une commande, le CCP analyse la commande. Si le nom du programme est spécifié sans extension, le système considère en premier lieu que l'extension est .BAT (fichier BATCH). Il cherche alors une entrée correspondante dans le répertoire. Si aucun fichier n'est trouvé, il associe au nom du fichier l'extension .PRG et recommence sa recherche.

Le CCP ou même un programme utilisateur peuvent charger un autre programme en mémoire en invoquant l'appel 4B du BDOS (voir *table des fonctions du BDOS*). Quand le programme est chargé, la TPA contient le segment de programme (texte, données et BSS), une pile utilisateur (256 octets minimum) et une PAGE DE BASE. Il existe une PAGE DE BASE par programme chargé en mémoire. La PAGE DE BASE est constituée de 256 octets qui définissent l'environnement d'exécution du programme.

Structure de la TPA

Adresse haute

<--- PILE DEPART



La pile est située au sommet de la TPA. Le sommet de la mémoire peut être réduit par l'utilisation de la fonction BDOS 4A (M_SHRINK). Cette opération permet de disposer d'un espace mémoire pour installer un autre programme sans enlever le premier.

Format de la page de base

<i>Adresse</i>	<i>Long</i>	<i>Num</i>	<i>Fonction</i>
00	4	P_LOWTPA	Adresse de base de la TPA
04	4	P_HITPA	Adresse haute de la TPA
08	4	P_TBASE	Adresse du texte (code du programme)
0C	4	P_TLEN	Longueur du texte en octets
10	4	P_DBASE	Adresse de base des données
14	4	P_DLEN	Longueur des données
18	4	P_BBASE	Adresse de base du BSS
1C	4	P_BLEN	Longueur du BSS
2C	4	P_ENV	Pointeur vers la chaîne d'environnement
80	80	P_CMDLIN	Image de la ligne de commande

Toutes les autres adresses sont réservées.

Sortie d'un programme

Il y a deux façons de sortir d'un programme pour retourner au CCP :

- interactivement par la frappe sur le CTRL-C.
Possible pendant l'exécution d'un appel BDOS 01, 02, 08, 09, 0A ou 10.
Cette sortie termine le programme en cours et exécute un démarrage à chaud qui rend la main au CCP.
- par programme (appel de la fonction BDOS 0 ou 31 ou 4C).

Remarque : un fichier exécutable peut se définir suivant trois modes différents :

- le mode **GEM** (en principe extension .PRG) qui est prévu pour les fichiers pouvant s'exécuter sous GEM ;
- le mode **TOS** (en principe extension .TOS) qui est prévu pour s'exécuter sous le GEMDOS sans activation de la souris et de l'écran graphique ;
- le mode **TOS** avec **paramètres** (extension .TTP) qui est prévu pour des fichiers utilisables sous le GEMDOS et nécessitant lors de leur initialisation un ou plusieurs paramètres.

Certains programmes peuvent fonctionner sous différents modes.

FORMAT D'UN FICHER EXECUTABLE

Un fichier exécutable sous GEMDOS est le résultat d'une conversion par l'utilitaire **RELMOD** (voir tome 2).

Un fichier exécutable se compose d'un en-tête suivi d'un segment texte (code exécutable) et d'un segment de données. Optionnellement, il peut contenir une table de symbole et des informations de relogement (RELOCATION).

Format de l'en-tête

<i>Adresse</i>	<i>Long</i>	<i>Fonction</i>
00	2	Contient 601AH
02	4	Nombre d'octets dans le segment texte
06	4	Nombre d'octets dans le segment données
0A	4	Nombre d'octets dans le segment BSS
0E	4	Nombre d'octets dans la table des symboles
12	4	Doivent être 00
16	4	Doivent être 00
1A	2	Réservé

Table des symboles

La table des symboles définit les symboles auxquels le fichier peut faire référence. Chaque entrée de la table est associée à un index qui indique le numéro d'entrée. Les entrées sont numérotées séquentiellement en commençant par zéro. Chaque entrée est composée de 14 octets définis comme suit :

Octets 0 à 7 : nom du symbole (8 octets maximum)

Octets 8 et 9: type :

0100 = relatif à BSS

0200 = relatif au texte

0400 = relatif aux données

0800 = référence externe

1000 = définition d'un registre

2000 = symbole global

4000 = symbole synonyme

8000 = symbole défini

Octets A à D: valeur du symbole. Ça peut être une adresse, un

numéro de registre, une valeur ou une expression.

FORMAT D'UN FICHER EXECUTABLE

Informations de relogement

Ces informations indiquent l'OFFSET (déplacement) entre les adresses longues (4 octets) qui doivent être relogées.

L'OFFSET du premier mot long à reloger se trouve juste après la table des symboles. A la suite de cet OFFSET (4 octets), on trouve une série d'octets qui définissent les OFFSETS successifs des mots longs.

Ces octets peuvent prendre les valeurs suivantes :

- 00 Fin de la relocation
- 01 Ajouter 254 à l'OFFSET
- 02 Réservé
- 03 Réservé
- 04 à 254 OFFSET du mot auquel il faut ajouter 254 fois le nombre de 1 qui le précède.
- 255 Réservé.

HANDLER PERIPHERIQUES

Le descripteur de HANDLER peut pointer indifféremment sur un descripteur de fichier (FCB) ou sur un descripteur de périphérique.

Dans le cas d'un périphérique, le descripteur de HANDLER contient un nombre compris entre 0 et 4 qui définit le périphérique.

- 0 : périphérique d'entrée standard. Les entrées peuvent être déroutées lorsque l'on est sur la ligne de commande.
- 1 : périphérique de sortie standard. Les sorties peuvent être déroutées lorsque l'on est sur la ligne de commande.
- 2 : périphérique d'erreur standard. Les entrées/sorties peuvent être déroutées.
- 3 : périphérique auxiliaire. Les entrées/sorties peuvent être déroutées.
- 4 : périphérique de sortie LIST. Les sorties peuvent être déroutées.

STRUCTURE DU DISQUE

Le disque GEMDOS est structuré de la façon suivante :

- 1- Une zone réservée pour le chargement du BOOT.
- 2- Une copie de la F.A.T.
- 3- Une deuxième copie de la F.A.T. (optionnelle).
- 4- Le directory ou répertoire principal.
- 5- La zone des données ou des sous-répertoires.

Allocation de l'espace disque

Sur un disque, la plus grande partie de l'espace libre est destinée à l'écriture des données en fichier.

L'allocation d'un espace disque se fait à la demande. Lors de l'ouverture d'un fichier, le GEMDOS ne réserve pas d'espace sur le disque.

C'est au moment de l'écriture des données que l'espace disque sera alloué au fichier et ce, par unité d'allocation.

L'unité d'allocation est appelée CLUSTER. Un cluster est toujours constitué d'un ou de plusieurs secteurs. La taille du secteur et de l'unité d'allocation est fonction du disque utilisé et de sa capacité.

Ces paramètres font partie du secteur BOOT.

Lorsque le cluster du fichier est rempli, le GEMDOS en alloue un autre si cela est possible. Le choix du cluster s'effectue suivant deux critères :

- 1- Le cluster est libre et non défectueux.
- 2- Le cluster le plus proche du début du fichier.

Le système peut trouver rapidement le cluster le plus adéquat en consultant la F.A.T. (table d'allocation de fichier) qui référence chaque cluster de la zone donnée du disque.

TABLE D'ALLOCATION DE FICHIER F.A.T.

La F.A.T. est une table qui indique l'état de chaque unité d'allocation ou cluster.

L'état d'un cluster peut être :

- 1- Disponible.
- 2- Utilisé par un fichier.
- 3- Indisponible (secteur défectueux).

Le système d'exploitation vérifie la disponibilité de la F.A.T. avant chaque écriture sur disque. La F.A.T. est enregistrée sur le disque lors de la commande FORMAT et est située juste après le BOOT.

Si une F.A.T. est constituée de plus d'un secteur, ceux-ci sont consécutifs.

La commande FORMAT exécute deux copies de la F.A.T. en vue d'assurer une plus grande fiabilité au système.

La première F.A.T. est chargée par le GEMDOS dans un tampon mémoire, chaque fois qu'un fichier doit être ouvert ou qu'une allocation d'espace disque est nécessaire.

Une haute priorité est donnée au tampon qui contient la F.A.T., de sorte qu'elle réside le plus longtemps possible en mémoire. Ceci a pour effet une diminution considérable des temps d'accès aux fichiers.

Entrées initiales de la F.A.T.

Chaque cluster du disque possède une entrée dans la F.A.T. Une entrée est constituée de 12 bits.

Les deux premières entrées de la F.A.T. correspondent au répertoire et contiennent des valeurs indiquant la taille et le format du disque utilisé.

Remarque : les deuxième et troisième octets de la F.A.T. valent FFFFH.

Entrées fichier de la F.A.T.

Chaque entrée de la F.A.T. est constituée d'une valeur hexadécimale sur trois digits.

Ces valeurs peuvent être :

- 000H : le cluster est disponible.
- 002H-FF6H : le cluster est utilisé, la valeur donne le numéro de l'entrée suivante (chainage des clusters d'un fichier).
- FF7H : le cluster est défectueux et ne sera pas utilisé.
- FF8H-FFFH : dernier cluster (fin de chaîne).
- 001H : INVALIDE.

STRUCTURE DES REPERTOIRES

Le GEMDOS supporte une structure de répertoire arborescent de type UNIX.

Les répertoires en arbre permettent de découper l'ensemble des fichiers d'un disque en différents groupes. A chaque groupe correspond un sous-répertoire dont le nom figure dans le répertoire principal encore appelé tronc ou racine.

Le DOS attribue une unité et un répertoire par défaut au système. Il permet cependant de modifier le répertoire courant (CHDIR) ou de spécifier un chemin d'accès entre les répertoires (PATH).

Organisation au niveau du disque

Le répertoire principal est situé à un endroit bien déterminé. En outre, il possède une taille fixe.

Il est constitué par une table dont chaque entrée correspond soit au nom de disque, soit à un fichier ou à un sous-répertoire du premier niveau.

Les sous-répertoires sont des fichiers classiques dont l'organisation interne suit la même structure que le répertoire principal.

Le nombre de niveaux de l'arbre, autrement dit le nombre de sous-répertoires emboîtés les uns dans les autres, n'est limité que par la longueur de chemin nécessaire pour atteindre le dernier niveau. Le GEMDOS limite cette longueur à 64 caractères.

Exemple :

TEXTES\UTILITAIRES\AVOCAT\....\....

Limité à 64 caractères

Chaque entrée d'un répertoire ou d'un sous-répertoire est constituée de 32 octets.

STRUCTURE DES REPERTOIRES

Structure d'une entrée

<i>Nom du champ</i>	<i>Taille en octets</i>	<i>Offset hexa</i>	<i>Offset déci</i>
Nom	8	00H - 07H	0 - 7
Extension	3	08H - 0AH	8 - 10
Attributs	1	0BH	11
Réservés	10	0CH - 15H	12 - 21
Heure M.A.J.	2	16H - 17H	22 - 23
Date M.A.J.	2	18H - 19H	24 - 25
Premier CLUSTER	2	1AH - ABH	26 - 27
Taille fichier	4	1CH - 1FH	28 - 31

Description des champs

Nom Composé de 1 à 8 caractères. Il est cadré à gauche et complété par des blancs (20H).

En examinant le premier octet du nom, le GEMDOS peut déterminer l'état de l'entrée.

<i>Octet</i>	<i>Signification</i>
00H	L'entrée n'a jamais été utilisée et elle marque la fin du répertoire. Cette méthode permet de diminuer le temps de recherche dans le répertoire.
E5H	L'entrée précédemment occupée est actuellement libre. Le fichier a été effacé (DEL).
2EH	L'entrée correspond au répertoire lui-même. Si le second octet contient également 2EH, l'entrée correspond au sous-répertoire du répertoire précédent.
***	Toute autre valeur référence un fichier standard.

STRUCTURE DES REPERTOIRES

- *Extension* : 3 octets cadrés à gauche et complétés par des blancs.
- *Attribut* : ce champ est constitué d'un seul octet indiquant le type de fichier référencié par le répertoire. (*Voir ATTRIBUT DE FICHIER*).
- *Réservés* : ces dix octets sont réservés par le GEMDOS.
- *Heure M.A.J.* : ce champ de deux octets indique soit l'heure de création du fichier, soit l'heure de la dernière fermeture du fichier.

Format :

```

----- Octet 17H -----      ----- Octet 16H -----
B15 B14 B13 B12 B11 B10 B9 B8 B7 B6 B5 B4 B3 B2 B1 B0
H  H  H  H  H  M  M  M  M  M  M  S  S  S  S  S

```

HHHHH correspond à l'heure (0 - 23) en binaire.
 MMMMM correspond aux minutes (0 - 59) en binaire.
 SSSSS correspond à la moitié des secondes (0 - 29).

- *Date M.A.J.* : ce champ de deux octets indique soit la date de création du fichier, soit la date de dernière fermeture du fichier.

Format :

```

----- Octet 19H -----      ----- Octet 18H -----
B15 B14 B13 B12 B11 B10 B9 B8 B7 B6 B5 B4 B3 B2 B1 B0
A  A  A  A  A  A  A  M  M  M  M  J  J  J  J  J

```

AAAAAAA correspond à l'année : ce nombre doit être ajouté à 1980 pour former la date réelle. Cependant, le système ne permet pas l'entrée directe de date inférieure à 1980 ou supérieure à 1999.
 MMMM correspond au mois (1 à 12).
 JJJJ correspond au jour (1 à 31).

- *Premier CLUSTER* : ce champ de deux octets détermine le numéro du premier CLUSTER du fichier.

Remarque : le premier CLUSTER de la zone donnée du disque est 002.

- *Taille du fichier* : ce champ de quatre octets indique la taille du fichier exprimée en nombre d'octets. L'octet le plus significatif est le quatrième.

ATTRIBUTS DES FICHIERS

Le système d'exploitation GEMDOS associe à chaque entrée de répertoire un certain nombre de bits qui permettent de caractériser le fichier concerné. Regroupés en un octet, ces bits portent le nom d'attributs de fichier. Ils indiquent au système en présence de quel type de fichier il se trouve.

La condition de chaque attribut est réalisée lorsque le bit correspondant est mis à 1. Ils sont au nombre de six :

- bit 0 (01H) : attribut "READ ONLY"
Seule la lecture de ce fichier est autorisée.
Toute tentative d'effacement ou d'écriture fournira une erreur.
- bit 1 (02H) : attribut "HIDDEN"
Les fichiers marqués de cet attribut ne sont pas représentés lors d'une demande d'affichage du répertoire ; ils sont cachés.
- bit 2 (04H) : attribut "SYSTEM"
Il indique que ce fichier fait partie du système d'exploitation. Il est exclu de l'affichage normal du répertoire.
- bit 3 (08H) : attribut "VOLUME LABEL"
Cette valeur signale que les onze caractères enregistrés dans l'espace réservé au nom et à l'extension du fichier sont en fait le nom du disque. Cette entrée ne contient aucune autre information utilisable et n'est significative que dans le répertoire principal (\).
- bit 4 (10H) : attribut "DIRECTORY"
Ce bit indique que cette entrée est en fait un sous-répertoire.
- bit 5 (20H) : attribut "ARCHIVE"
Cet attribut est mis à "1" lorsque le fichier a été modifié et refermé. Il est remis à zéro par la commande BACKUP qui l'utilise pour ne sauvegarder que les fichiers ayant été modifiés lorsque le paramètre /M est précisé.
- bits 6 et 7 : toujours à zéro.

Les quatre attributs "READ ONLY", "HIDDEN", "SYSTEM" et "ARCHIVE" peuvent être utilisés ensemble pour le même fichier sans problème.

Les fichiers marqués de l'attribut "HIDDEN" ne sont pas affichés dans le répertoire lors d'une commande DIR.

FILE CONTROL BLOC (FCB)

Le GEMDOS fournit au programmeur un ensemble de sous-programmes permettant de manipuler les fichiers sur disque. En plus du fait qu'elles accèdent aux données sur disque, ces fonctions utilisent deux emplacements mémoire. Ces derniers sont connus sous le nom de blocs de contrôle de fichiers (FCB) ou HANDLER et espace de transfert au disque (DTA).

Le DTA est un espace mémoire par lequel passeront toutes les données en provenance ou en direction du disque. Lorsqu'un programme prend le contrôle du DOS, le DTA se trouve à l'OFFSET 80H de la page Basic.

Le FCB présente la structure suivante :

		Nom du champ	Taille octets	Offset hexa	Offset décim.
F C B E T E N D U	Extension	Indicateur	1	F9	-7
		Réservé	5	FA	-6
		Attributs	1	FF	1
	F	Disque	1	00	00
	C	Nom du fichier	0	01-08	01-08
	B	Suffixe	3	09-0B	09-11
		Bloc courant	2	0C-0D	12-13
	S	Taille enregistrement	2	0E-0F	14-15
	T	Taille du fichier 4		10-13	16-19
	A	Date de la dernière écriture	2	14-15	20-21
N	Heure de la dernière écriture	2	16-17	22-23	
D	Réservé	8	18-1F	24-31	
A	Enregistrement courant	1	20	32	
R	Numéro de l'enregistrement	4	21-24	33-36	
D					

FCB standard

Le FCB standard est initialisé pour contenir des informations concernant un fichier dont la gestion des attributs n'est pas nécessaire.

Disque (offset 00H)

Cet espace décrit le disque par un chiffre (1=A, 2=B, etc.). La valeur 0 peut être utilisée pour désigner le disque courant ; la fonction Open File (OFH) se chargera de l'ajuster à la valeur correcte.

FILE CONTROL BLOC (FCB)

Nom du fichier (Offset 01H-08H)

Il contient le nom du fichier en huit caractères, justifié à gauche.

Suffixe (Offset 09H-0BH)

Le champ contient l'extension du fichier en trois caractères. Si elle existe, elle est justifiée à gauche.

Bloc courant (Offset 0CH-0DH)

Ce nombre donne l'emplacement du bloc qui contient l'enregistrement courant (un bloc est un groupe de 128 enregistrements).

La valeur du bloc courant est relative au début du fichier et est initialisée à zéro par la fonction 3DH (Open File). Le bloc courant et l'enregistrement courant (Offset 20H) forment ensemble le pointeur vers l'enregistrement actuel pour les lectures et écritures séquentielles.

Taille des enregistrements (Offset 0EH-0FH)

La taille des enregistrements est exprimée en octets. Elle détermine le nombre d'octets à transférer quand un enregistrement est lu ou écrit. Cette valeur est mise à 80H (128 décimal), lors de l'ouverture d'un fichier, mais peut être modifiée par l'utilisateur pour d'autres enregistrements.

Taille du fichier (Offset 10H-13H)

La taille du fichier est enregistrée en octets. Le premier mot (2 octets) correspond à la partie basse de la taille.

Date de la dernière écriture (Offset 14H-15H)

La date de création du fichier ou de sa dernière écriture est indiquée par deux octets. La valeur des jours, mois et années est mémorisée de la façon suivante :

-----	octet 15H	-----	-----	octet 14H	-----										
B15	B14	B15	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
a	a	a	a	a	a	a	m	m	m	m	j	j	j	j	j

où aaaaaaa constitue la valeur binaire de l'année. Elle varie de 0 à 119 et est ajoutée à 1980.

mmmm constitue la valeur binaire du mois. Elle varie de 1 à 12.

jjjjj constitue la valeur du jour et varie de 1 à 31.

Heure de la dernière lecture (Offset 16H-17H)

L'heure de création du fichier ou de sa "dernière" écriture est enregistrée dans cet espace de la manière suivante :

FILE CONTROL BLOC (FCB)

----- Octet 17H ----- Octet 161H -----
 B15 B14 B13 B12 B11 B10 B9 B8 B7 B6 B5 B4 B3 B2 B1 B0
 h h h h h m m m m m m s s s s s

où hhhhh représente l'heure en binaire et varie de 0 à 23.
 mmmmm représente les minutes en binaire et varie de 0 à 59.
 ssss représente les secondes multipliées par deux, en binaire.

Réservé (Offset 18H-1FH)

Cet espace est réservé pour le GEMDOS.

Enregistrement courant (Offset 20H)

Ce champ donne l'emplacement d'un des 128 enregistrements du bloc courant. Il n'est pas initialisé par la fonction ouverture de fichier (3DH). L'utilisateur doit le positionner avant de réaliser une écriture ou une lecture séquentielle sur le disque.

Numéro de l'enregistrement (Offset 21H-24H)

Ce numéro donne la position de l'enregistrement actuellement sélectionné, à partir du début du fichier considéré comme position 0. Il n'est pas initialisé lors de l'ouverture du fichier et doit donc être positionné avant toute opération de lecture aléatoire. Si la longueur d'un enregistrement est inférieure à 64 octets, les deux mots de cet espace sont utilisés ; sinon, seuls les trois premiers octets sont utilisés.

Remarque : si le FCB est positionné à l'offset 5H du PSP, le dernier octet empiètera sur le premier du DTA qui se trouve en 80H. Il faut donc sauver la première ligne du fichier dans un buffer, s'il est examiné après avoir été ouvert.

Contenu de la DTA

<i>adresse</i>	<i>contenu</i>
0 - 20	Réservé au GEMDOS
22 - 23	Heure
24 - 25	Date
26 - 29	Taille du fichier
30 - 43	Nom du fichier et extension

TABLE DES FONCTIONS DU BDOS

Généralités

Les fonctions internes du BDOS de GEMDOS sont semblables à celles du CP/M et du MS-DOS. Les concepteurs ont fait un effort tout particulier pour conserver les mêmes numéros d'appel que ceux du MS-DOS. Cependant, certaines fonctions des premières versions du MS-DOS incompatibles avec le CP/M n'ont pas été implantées.

L'appel des fonctions est cependant fondamentalement différent. Les processeurs de la série 8080 utilisés dans le CP/M et le MS-DOS sont des processeurs dits "à registres", autrement dit les paramètres sont passés par l'intermédiaire des registres du processeur. Bien que le nombre de ses registres soit important, le 68000 est plutôt utilisé comme processeur "à pile". Les paramètres sont donc passés par l'intermédiaire de la pile utilisateur.

L'appel des fonctions est réalisé par l'instruction TRAP #1.

Le programmeur doit prendre en charge la gestion de la pile à l'issue de l'appel.

Après chaque appel, les registres A0 et D0 sont modifiés. D0 contient soit un code d'erreur soit une valeur fournie par la fonction. A0 contient en général l'adresse de la pile. Le programmeur doit donc veiller à sauvegarder ces deux registres si ils contiennent des informations importantes avant l'appel de la fonction.

Exemple d'appel

Pour illustrer la méthode employée, voyons comment utiliser l'appel de la fonction 5 (C_PRNOUT) pour imprimer un caractère (un Z par exemple) sur le périphérique imprimante.

MOVE.W	#'Z',-(SP)	Introduction du caractère Z dans la pile
MOVE.W	#5,-(SP)	Introduction du numéro de fonction (5) dans la pile.
TRAP	#1	Appel de la fonction.
ADDQ.L	#4,SP	Ajoute 4 à la pile en sortie pour la corriger 4 octets = 2 mots de 16 bits (les 2 MOVE.#).
TST.W	D0	Teste D0 pour voir s'il y a une erreur.
BEQ	...	Suite du programme.

Numéro d'erreur

L'appel des fonctions du BDOS fournit un certain nombre de codes d'erreur éventuels qui sont retournés dans le registre D0. Ces codes se présentent sous la forme d'un nombre négatif exprimé sur 32 bits en binaire complément à deux. Ainsi, l'erreur -2 se coderait FFFFFFFE.

TABLE DES FONCTIONS DU BDOS

<i>Erreur GEM-DOS dans D0</i>	<i>Erreur MS-DOS équivalente</i>	<i>Signification</i>
-32	1	Appel de fonction invalide.
-33	2	Fichier non trouvé.
-34	3	Chemin d'accès non trouvé.
-35	4	Plus d'HANDLER disponible. (trop de fichiers ouverts).
-36	5	Accès refusé (raisons diverses).
-37	6	Descripteur de HANDLER invalide.
-39	8	Mémoire insuffisante.
-40	9	Adresse de bloc mémoire invalide.
-46	15	Spécification disque invalide.
-49	18	Plus de fichier (voir F_SNEXT).
-64		Erreur de valeur.
-65		Erreur interne.
-66		Format du programme à charger invalide.
-67		Modification de la taille d'un bloc impossible.

Explications de la table

NUMERO

Le numéro représente le code d'appel de la fonction. C'est donc lui qui doit être poussé dans la pile sous la forme d'un mot de 16 bits. Il doit constituer le sommet de la pile et doit donc être poussé le dernier, juste avant l'appel de l'instruction TRAP #1.

NOM

Le nom représente le nom officiel de la fonction utilisé dans les librairies C (en C il doit être en caractères minuscules).

ENTREES

Entrées représente les paramètres qui doivent être poussés dans la pile. Les paramètres doivent être vus depuis le sommet de la pile. Autrement dit, ils doivent être mis en pile dans l'ordre inverse de leur description.

Exemple :

La fonction 25 (S_SETVEC) demande en entrée deux premiers octets qui fournissent le numéro de vecteur suivi de 4 octets qui fournissent l'adresse de positionnement du vecteur.

TABLE DES FONCTIONS DU BDOS

La pile sera contruite de la façon suivante :

MOVE.L adresse,-(SP) Adresse de positionnement (4 octets).
MOVE.W numvec,-(SP) Numéro du vecteur.
MOVE.W #\$25,-(SP) Numéro de la fonction (S=hexa).

SORTIE

Sortie représente l'état des registres et des tampons à l'issue de l'appel de la fonction.

Remarque : D0.L représente le registre D0 sur 32 bits (long mot).

D0.W représente le registre D0 sur 16 bits (Word=mot). Ce mot est constitué des deux octets les moins significatifs de D0.L.

Numéro : 0

Nom : P_TERM0

Fonction : Termine le processus appelé en cours et retourne au processus appelant avec un code de retour de 0.

Entrée : Rien.

Sortie : Rien.

Numéro : 1

Nom : C_CONIN

Fonction : Lecture d'un caractère en provenance du périphérique standard d'entrée avec echo (copie) sur le périphérique standard de sortie.

Entrée : Rien.

Sortie : D0.L contient le caractère saisi.

<00>	<code ou 00>	<00>	<caractere>
B31-B24	B23-B16	B15-B8	B7-B0

Si le périphérique d'entrée est compatible GEM VDI alors l'appel de la fonction 1 place le code correspondant à la touche enfoncée dans l'octet bas du mot haut du registre D0 (bits 16 à 23). Si le périphérique n'est pas GEM compatible, l'octet est mis à zéro.

Si le caractère entré est 03 (CTRL C), le système exécute un démarrage à froid.

TABLE DES FONCTIONS DU BDOS

Numéro : 2

Nom : C_CONOUT

Fonction : Ecriture d'un caractère sur le périphérique standard de sortie. Le caractère à écrire doit être sur l'octet le moins significatif d'un mot de 16 bits, l'octet le plus significatif doit valoir 0. Cette fonction transforme les tabulations en une suite d'espaces et teste le CTRL-S (arrêt du défilement), CTRL-Q (reprise du défilement) et CTRL-P (copie imprimante). Il traite aussi le CTRL-C qui suspend l'opération en cours et produit un démarrage à chaud du système.

Entrée : Un mot de 16 bits contenant le caractère à écrire.

Sortie : Rien.

Numéro : 3

Nom : C_AUXIN

Fonction : Lecture d'un caractère en provenance du périphérique auxiliaire.

Entrée : Rien.

Sortie : L'octet le moins significatif de D0.L contient le caractère lu.

Numéro : 4

Nom : C_AUXOUT

Fonction : Ecriture d'un caractère sur le périphérique auxiliaire.

Entrée : Un mot de 16 bits dont l'octet le plus significatif vaut 0 et l'octet le moins significatif est le caractère à écrire.

Sortie : Rien.

Numéro : 5

Nom : C_PRNOUT

Fonction : Ecriture d'un caractère sur le périphérique imprimante.

Entrée : Un mot de 16 bits dont l'octet le plus significatif vaut 0 et l'octet le moins significatif est le caractère à écrire.

Sortie : Rien.

TABLE DES FONCTIONS DU BDOS

Numéro : 6

Nom : C_RAWIO

Fonction : Ecriture ou lecture sur la CONSOLE.

Entrée : 1 mot de 16 bits définissant le sens du mouvement (écriture ou lecture). Si le mot vaut 00FF, l'opération de lecture est déclenchée. Si le mot est différent de 00FF, alors l'octet le moins significatif du mot est écrit sur la console.

Sortie : Si le mot d'entrée vaut 00FF (lecture), alors l'octet bas de D0.L contient le caractère lu ou 0 s'il n'y a pas de caractère à lire.

Numéro : 7

Nom : C_RAWCIN

Fonction : Lecture d'un caractère en provenance du périphérique d'entrée standard sans produire d'écho sur le périphérique de sortie standard. Cette fonction ne teste pas les caractères de contrôle.

Entrée : Rien.

Sortie : L'octet le moins significatif de D0.L contient le caractère lu.

Numéro : 8

Nom : C_NECIN

Fonction : Lecture d'un caractère en provenance du périphérique d'entrée standard sans produire d'écho sur le périphérique de sortie standard. Cette fonction interprète les codes de contrôle.

Entrée : Rien.

Sortie : L'octet le moins significatif de D0.L contient le caractère lu.

Numéro : 9

Nom : C_CONWS

Fonction : Ecriture d'une série de caractères terminée par un octet à 00 sur le périphérique de sortie standard. Cette fonction traite le CTRL-C (03) qui suspend l'opération en cours et exécute un démarrage à chaud.

Entrée : 4 octets qui contiennent l'adresse de la chaîne de caractères à écrire.

Sortie : Rien.

TABLE DES FONCTIONS DU BDOS

Numéro : 0A

Nom : C_CONRS

Fonction : Lecture d'une chaîne de caractères en provenance du périphérique standard d'entrée et écriture dans un tampon d'entrée. La fonction s'arrête à la réception d'un retour-chariot (0DH). Si la taille du tampon est atteinte, le système ne prend plus les caractères en compte et produit un BELI. (07II). La fonction CONRS gère les caractères d'édition (BACKSPACE, CTRL-E, CTRL-Q, CTRL-R, CTRL-S, CTRL-U et CTRL-X). Elle suspend l'opération et génère un démarrage à chaud à la réception d'un CTRL-C.

Entrée : 4 octets qui contiennent l'adresse du tampon de réception dont le format est spécifié ci-dessous :

- Octet 0 : Nombre de caractères que peut contenir le tampon. Caractère de fin (0DH) non inclus.
- Octet 1 : Nombre de caractères reçus.
- Octet 2 : Premier caractère reçu.
- Octet .. :
- Octet n : Caractère 0DH (retour-chariot)

Sortie : Le tampon contient la chaîne lue sous le format spécifié ci-dessus.

Numéro : 0B

Nom : C_CONIS

Fonction : Test du statut (état) du périphérique d'entrée standard pour déterminer s'il y a un caractère à lire.

Entrée : Rien.

Sortie : D0.L (32 bits) contient 0 s'il n'y a pas de caractère à traiter.

Numéros : 0C & 0D

Non utilisée

Numéro : 0E

Nom : D_SETDRV

Fonction : Sélection du disque courant (A-P). Ce disque se retrouve dans l'octet 4 du bloc de paramètres (DPB).

TABLE DES FONCTIONS DU |BDOS

Entrée : Un mot de 16 bits qui spécifie le disque courant
B0 = disque A
B1 = disque B
.....
B15= disque P

Sortie : D0.L contient la table des disques suivant le même codage que l'entrée.

Numéro : 0F Non utilisée

Numero : 10 Nom : C_CONOS

Fonction : Test du statut du périphérique de sortie standard pour déterminer s'il est prêt à recevoir un caractère. Cette fonction gère le CTRL-C qui suspend l'opération en cours et produit un démarrage à chaud du système.

Entrée : Rien.

Sortie : D0.L contient 0 si le périphérique de sortie n'est pas disponible.

Numéro : 11 Nom : C_PRNOS

Fonction : Test de l'état du périphérique d'impression pour déterminer s'il est prêt à recevoir un caractère.

Entrée : Rien.

Sortie : D0.L contient 0 si le périphérique n'est pas prêt.

Numéro : 12 Nom : C_AUXIS

Fonction : Test de l'état du périphérique auxiliaire pour déterminer s'il y a un caractère à traiter.

Entrée : Rien.

Sortie : D0.L contient 0 s'il n'y a pas de caractère à traiter.

TABLE DES FONCTIONS DU BDOS

Numéro : 13

Nom : C_AUXOS

Fonction : Test de l'état du périphérique auxiliaire de sortie pour déterminer s'il est prêt à recevoir un caractère.

Entrée : Rien.

Sortie : D0.L contient 0 si le périphérique est inaccessible.

Numéros : 14 à 18

Non utilisée

Numéro : 19

Nom : D_GETDRV

Fonction : Lecture de l'unité disque courante.

Entrée : Rien.

Sortie : D0.L contient le numéro du disque courant (0=A, 1=B, ... 15=P).

Numéro : 1A

Nom : F_SETDTA

Fonction : Positionne l'adresse d'une table DTA (DISK TRANSFER ADDRESS). S'il n'y a pas de DTA positionnée, GEMDOS utilise la DTA par défaut située à l'adresse d'OFFSET 80H par rapport à la page de base.

Entrée : 4 octets qui contiennent l'adresse de positionnement de la DTA.

Sortie : Rien.

Numéros : 1B à 1F

Non utilisée

Numéro : 20

Nom : SUPERVISOR

Fonction : Passage du mode utilisateur en mode superviseur et vice-versa. Cette fonction permet d'appeler en mode utilisateur des variables système ou des périphériques uniquement disponibles en mode superviseur.

TABLE DES FONCTIONS DU BDOS

Entrée : Un mot de 16 bits. Si ce mot vaut 1, La fonction effectue un test du mode courant. Toute autre valeur produit un changement de mode.

Sortie : En sortie, si le mot d'entrée vaut 1, D0.L contient 0 si l'on se trouve en mode utilisateur et -1 (FFFFFFFF) si l'on se trouve en mode superviseur. Dans les autres cas, D0.L contient l'ancienne adresse de la pile superviseur.

Numéros : 21 à 24

Non utilisée

Numéro : 25

Nom : S_SETVEC

Fonction : Positionne un vecteur d'exception. Cette fonction associe un numéro de vecteur à une adresse spécifiée.

Entrée : Les deux premiers octets contiennent un numéro de vecteur sur 16 bits. Les 4 octets suivants contiennent la nouvelle adresse du vecteur.

Sortie : Rien.

Numéros : 26 à 29

Non utilisée

Numéro : 2A

Nom : T_GETDATE

Fonction : Lecture de la date interne.

Entrée : Rien.

Sortie : D0.W (16 bits) contient la valeur de la date lue :
B0 à B4 : Jour (1 à 31)
B5 à B8 : Mois (1 à 12)
B9 à B15: Année (0 à 119) à ajouter à 1980.

Numéro : 2B

Nom : T_SETDATE

Fonction : Positionnement de la date interne.

TABLE DES FONCTIONS DU BDOS

Entrée : 2 octets qui contiennent la date définie comme ci-dessus (fonction 2A).

Sortie : D0.L contient -1 (FFFFFFFF) si la date est erronée.

Numéro : 2C

Nom : T_GETTIME

Fonction : Lecture de l'heure interne.

Entrée : Rien.

Sortie : D0.W (16 bits) contient l'heure interne dans le format suivant :
B0 à B4 : Valeur des secondes par pas de 2 secondes
B5 à B10 : Valeur des minutes (0 à 59)
B11 à B15 : Valeur des heures (0 à 23).

Numéro : 2D

Nom : T_SETTIME

Fonction : Mise à jour de l'heure interne.

Entrée : 2 octets qui contiennent l'heure définie comme ci-dessus (fonction 2C).

Sortie : D0.L contient -1 (FFFFFFFF) si l'heure est erronée.

Numéro : 2F

Nom : F_GETDTA

Fonction : Retourne l'adresse de la DTA courante.

Entrée : Rien.

Sortie : D0.L contient l'adresse de la DTA courante.

Numéro : 30

Nom : S_VERSION

Fonction : Lecture du numéro de version.

Entrée : Rien.

Sortie : D0.W contient l'indice de version. L'octet le moins significatif contient le numéro de version majeure et l'octet le plus significatif contient le numéro de version mineure (petites modifications).

TABLE DES FONCTIONS DU BDOS

Numéro : 31

Nom : P_TERMRES

Fonction : Termine le processus en cours et ne relâche pas la mémoire affectée à ce processus. Le nombre d'octets à verrouiller doit être passé comme paramètre à la fonction. Cette fonction ne doit pas être utilisée pour des raisons de compatibilité avec les prochaines versions du GEMDOS.

Entrée : Les 4 premiers octets contiennent le nombre d'octets à verrouiller. Les deux suivants contiennent un code de sortie éventuel qui sera communiqué au processus appelant par l'intermédiaire de D0.

Numéros : 32 à 34

Non utilisée

Numéro : 35

Nom : S_GETVEC

Fonction : Lecture d'un vecteur d'exception.

Entrée : 1 mot de 16 bits qui contient le numéro du vecteur.

Sortie : D0.L contient l'adresse du vecteur spécifié.

Numéro : 36

Nom : D_FREE

Fonction : Lecture de l'espace disponible sur un disque.

Entrée : Les 4 premiers octets pointent sur un tampon qui recevra les informations concernant l'espace disponible. Les 2 suivants déterminent le numéro de l'unité disque (0= unité courante, 1= disque A, 2= disque B,....).

Sortie : D0.L contient un code erreur éventuel (-46) si l'unité disque spécifiée est invalide.

L'information sur l'espace disponible est fournie sous la forme de 16 octets définis comme suit :

- Octets 0 à 3 : Nombre d'octets disponibles sur le disque.
 - Octets 4 à 7 : Nombre de CLUSTERS disponibles.
 - Octets 8 à 11 : Nombre d'octets par secteur.
 - Octets 12 à 15 : Nombre de secteurs par CLUSTER.
-

TABLE DES FONCTIONS DU BDOS

Numéros : 37 et 38

Non utilisés

Numéro : 39

Nom : D_CREATE

Fonction : Crée un sous-répertoire.

Entrée : 4 octets qui pointent sur un tampon qui contient une chaîne de caractères terminée par 00 déterminant le chemin d'accès et le nom du sous-répertoire.

Sortie : Si le sous-répertoire est créé, D0.L contient 0 ; sinon, il contient un code d'erreur.

-34 (FFFFFFDE) Chemin d'accès non trouvé.

-36 (FFFFFDC) Accès refusé.

Numéro : 3A

Nom : D_DELETE

Fonction : Efface un sous-répertoire.

Entrée : 4 octets qui pointent sur un tampon contenant une chaîne de caractères terminée par 00 qui détermine le chemin d'accès et le nom du sous-répertoire à effacer.

Sortie : D0.L contient 0 si l'opération réussit ; sinon, D0.L contient un code d'erreur.

-34 (FFFFFFDE) Chemin d'accès non trouvé.

-36 (FFFFFFDC) Accès refusé (répertoire non vide).

-65 Erreur interne.

Numéro : 3B

Nom : D_SETPATH

Fonction : Positionnement sur un sous-répertoire.

Entrée : 4 octets qui pointent sur un tampon qui contient une chaîne de caractères terminée par 00 déterminant le chemin d'accès et le nom du sous-répertoire sur lequel on veut se positionner.

Sortie : D0.L contient 0 si l'opération réussit ; sinon, D0.L contient un compte rendu d'erreur.

-34 (FFFFFFDE) Chemin d'accès non trouvé

TABLE DES FONCTIONS DU BDOS

Numéro : 3C

Nom : F_CREATE

Fonction : Création d'un fichier. Si le fichier que l'on crée existe déjà, la fonction retourne un code d'erreur.

Entrée : Les 4 premiers octets contiennent l'adresse d'une chaîne de caractères qui spécifie le chemin d'accès (PATH) et le nom du fichier à créer. Cette chaîne doit se terminer par un octet à 00H. Ces 4 octets sont suivis d'un mot de 16 bits qui détermine les attributs du fichier avec les conventions suivantes :

00 = Fichier en lecture seulement (READ ONLY).

01 = Fichier caché (HIDDEN).

04 = Fichier système caché.

08 = L'entrée est un descripteur de volume et non un descripteur de fichier (création d'un sous-répertoire).

Sortie : D0.L contient n code d'erreur ou 0 si l'opération réussit.

-34 : (FFFFFFDE) Chemin d'accès non trouvé

-35 : (FFFFFFDD) Il n'y a plus de HANDLER (gérant) disponible pour installer le fichier

-36 : (FFFFFFDC) Accès refusé, répertoire plein ou fichier déjà existant.

Numéro : 3D

Nom : F_OPEN

Fonction : Ouvre un fichier. Cette opération est indispensable avant tout accès en lecture ou en écriture.

Entrée : Les 4 premiers octets contiennent l'adresse d'une chaîne de caractères qui contient le chemin d'accès et le nom du fichier. Cette chaîne doit se terminer par un octet à 00H. Ces 4 octets sont suivis par un mot de 16 bits qui contient le mode d'ouverture défini comme suit :

0 = Accès en lecture seulement.

1 = Accès en écriture seulement.

2 = Accès en lecture et en écriture.

Sortie : Si l'opération échoue, D0.L contient un compte rendu d'erreur. Si l'opération réussit, les deux octets les plus significatifs de D0.L contiennent 0 et les deux octets les moins significatifs de D0.L contiennent une valeur qui identifie le HANDLER et qui sera utilisée pour identifier le fichier lors de chaque opération (écriture, lecture, fermeture).

TABLE DES FONCTIONS DU BIOS

- 33 : (FFFFFFDF) fichier absent.
 - 35 : (FFFFFFDD) Plus de HANDLER disponible.
 - 36 : (FFFFFFDC) Accès refusé.
-

Numéro : 3E

Nom : F_CLOSE

Fonction : Ferme le fichier dont le HANDLER est spécifié en entrée. Lors de la fermeture, le répertoire et la table d'allocation sont remis à jour.

Entrée : 2 octets qui contiennent la valeur du HANDLER fournie à l'ouverture.

Sortie : D0.L contient le code -37 (FFFFFFDB) si le HANDLER est invalide.

Numéro : 3F

Nom : F_READ

Fonction : Lit un nombre de caractères à la position courante du fichier précisé.

Entrée : 2 octets qui contiennent le HANDLER fourni à l'ouverture du fichier suivis de 4 octets qui contiennent le nombre de caractères à lire et de 4 octets qui contiennent l'adresse du tampon où il faut écrire les octets lus.

Sortie : D0.L contient -37 (FFFFFFDB) si le HANDLER est incorrect.

Numéro : 40

Nom : F_WRITE

Fonction : Ecrit un nombre d'octets à la position courante dans le fichier précisé.

Entrée : 2 octets qui contiennent l'identificateur du HANDLER fourni à l'ouverture du fichier suivi de 4 octets qui contiennent le nombre de caractères à écrire et de 4 octets qui contiennent l'adresse du tampon qui contient les caractères à écrire.

Sortie : D0.L contient un code d'erreur éventuel :
-36 : (FFFFFFDC) si l'accès au fichier est refusé ou si le disque est plein.
-37 : (FFFFFFDB) si l'identificateur de HANDLER est invalide.

TABLE DES FONCTIONS DU BDOS

Numéro : 41

Nom : F_DELETE

Fonction : Efface le fichier spécifié.

Entrée : 4 octets qui pointent sur une chaîne de caractères terminée par 00 et qui contient l'identificateur du fichier (chemin d'accès et nom).

Sortie : D0.L contient un compte rendu d'erreur éventuel.
-36 (FFFFFFDC) Accès refusé.
-33 (FFFFFFDF) Fichier non trouvé.

Numéro : 42

Nom : F_SEEK

Fonction : Positionne le pointeur courant à l'intérieur d'un fichier.

Entrée : Les 4 premiers octets contiennent une valeur signée qui indique le déplacement à réaliser en avançant dans le fichier (+) ou en reculant (-). Les 2 octets suivants contiennent l'identificateur de HANDLER et les 2 derniers spécifient le mode de déplacement de la façon suivante :

- 0 = déplacement de N octets en partant du début du fichier.
- 1 = Déplacement de N octets en partant de la position courante.
- 2 = Déplacement de N octets en partant de la fin du fichier.

Sortie : D0.L contient un code d'erreur éventuel.
-37 (FFFFFFDB) Identificateur de HANDLER invalide.
-32 (FFFFFFE0) Fonction invalide.

Numéro : 43

Nom : F_ATTRIB

Fonction : Lit ou positionne les attributs d'un fichier.

Entrée : Les 4 premiers octets pointent sur une chaîne de caractères terminée par 00 qui identifie le fichier (chemin d'accès et nom); les deux octets suivants (mot de 16 bits) identifient le type d'opération à effectuer (0 = lecture, 1 = écriture). Les deux derniers octets indiquent les attributs à écrire dans le cas du mode écriture avec les conventions suivantes :

- 01 = Lecture seule.
- 02 = Caché.

TABLE DES FONCTIONS DU BDOS

- 04 = Système caché.
- 08 = Identificateur de volume.
- 10 = Sous-répertoire.
- 20 = Archive.

En sortie : D0.L contient un compte rendu d'erreur éventuel. Dans le cas de la lecture, les deux octets les moins significatifs de D0.L contiennent les attributs lus.

- 33 : (FFFFFFD1) Fichier non trouvé.
- 34 : (FFFFFFDE) Chemin d'accès non trouvé.

Numéro : 44

Nom : F_IOCTL

Fonction : Maintenance de divers paramètres d'entrée/sortie du système. C'est la fonction la plus complexe et la plus longue à décrire du système.

Entrée : Octets 1 et 2 : Code de sous-fonction

- 1 Substitution d'un vecteur d'interruption.
- 0 Lecture des informations du périphérique.
- 1 Réservée.
- 2 Lecture du canal de contrôle du périphérique.
- 3 Ecriture sur le canal de contrôle du périph.
- 4 Lecture du canal de contrôle du disque.
- 5 Ecriture sur le canal de contrôle du disque.
- 6 Lecture de l'état d'entrée.
- 7 Lecture de l'état de sortie.
- 8 lecture du type de média.

Octets 3 et 4 : Code du HANDLER ou du disque en fonction de la sous fonction choisie.

Octets 5 et 6 : Compteur ou bit-map en fonction de la sous fonction choisie.

Octets 7 à 10 : Adresse d'un tampon (4 octets) ou, si la sous fonction vaut -1, adresse du vecteur.

Explication des sous-fonctions

1 - **Entrée** : Octets 3 et 4 contiennent un descripteur de HANDLER.
Octets 5 et 6 contiennent 0.
Octets 7 à 10 contiennent l'adresse du vecteur de gestion du HANDLER.

Sortie : D0.W contient le numéro du vecteur d'interruption courant du périphérique.

TABLE DES FONCTIONS DU BDOS

0 - *Entrée* : Octets 3 et 4 contiennent un descripteur de HANDLER de fichier ou de périphérique.

Sortie : D0.L contient le message d'erreur éventuel -33 ou -37. Si l'opération réussit, D0.L. contient la bit-map du périphérique ou du fichier.

Les bits 0 à 15 dépendent du type de HANDLER.

Le BIT 7 de D0.L. indique le type de HANDLER.

0 = HANDLER FICHIER ; 1 = HANDLER PERIPHERIQUE

B0-B5 Numéro disque : B0 = 1 = CONSOLE INPUT

: B1 = 1 = CONSOLE OUTPUT

: B2 = 1 = NUL PERIPH

: B3 = 1 = CLK

B6 Réservé : B4-B6 Réservé

B8-B13 Réservé : B8-B13 Réservé

B14 = 1 = traitement : B14 = 1 = traitement des

des chaînes de contrôle : chaînes de contrôle

possible : possible

B15 réservé : B15 réservé

2 - *Entrée* : Octets 3 et 4 contiennent le descripteur de HANDLER d'un périphérique de type caractère.

Octets 5 et 6 contiennent le nombre de caractères à lire.

Octets 7 à 10 contiennent l'adresse du tampon qui recevra les caractères lus.

Sortie : D0.L contient le nombre de caractères lus ou un code d'erreur.

3 - Identique à 2 mais le nombre d'octets contenu dans les octets 5 et 6 sont à écrire et non à lire. De même D0.L. contient le nombre d'octets écrits dans le tampon.

4 - Identique à 2 mais les octets 3 et 4 contiennent le numéro du disque à la place du HANDLER (0=disque courant, 1=disque A...).

5 - Identique à 3 mais les octets 3 et 4 contiennent le numéro du disque à la place du HANDLER.

6 - *Entrée* : Les octets 3 et 4 contiennent le descripteur de HANDLER dont vous voulez obtenir le statut d'entrée.

Sortie : Si le HANDLER concerne un disque, D0.L. contient 0 si la fin du fichier est atteinte (EOF).

Si le HANDLER concerne un périphérique, D0.L. contient 0 si le périphérique n'est pas prêt.

7 - Identique à 6 mais pour l'état de sortie.

TABLE DES FONCTIONS DU BDOS

8 - *Entrée* : Octets 3 et 4 contiennent le numéro du disque.

Sortie : D0.L contient 0 si le disque est amovible et une autre valeur si le disque est fixe.

Numéro : 45

Nom : F_DUP

Fonction : Duplique le HANDLER. Cette fonction fournit un HANDLER supplémentaire pour gérer le fichier. Si on exécute un F_SEEK, le HANDLER d'origine et sa copie sont tous deux modifiés.

Entrée : 2 octets qui contiennent l'identificateur de HANDLER à dupliquer.

Sortie : D0.L contient l'identificateur du nouveau HANDLER ou un code d'erreur éventuel.
-35 : (FFFFFFDD) Plus de HANDLER disponible.
-37 : (FFFFFFDB) identificateur du HANDLER invalide.

Numéro : 46

Nom : F_FORCE

Fonction : Force un descripteur de HANDLER pour l'obliger à pointer sur le même fichier ou périphérique qu'un autre descripteur de HANDLER.

Entrée : Les deux premiers octets contiennent le descripteur de HANDLER à forcer, les deux suivants contiennent l'autre descripteur de HANDLER.

Sortie : D0.L contient un code d'erreur éventuel (-37) si un des descripteurs de HANDLER est invalide.

Numéro : 47

Nom : D_GETPATH

Fonction : Lecture du chemin d'accès courant.

Entrée : Les 4 premiers octets contiennent l'adresse d'un tampon de 64 octets qui contiendra le chemin d'accès lu. Les deux octets suivants contiennent le numéro du disque (0=disque courant, 1 = disque A, ...).

Sortie : D0.L contient le code d'erreur éventuel (-46) si le disque est invalide.

TABLE DES FONCTIONS DU BDOS

Numéro : 48

Nom : M_ALLOC

Fonction : Allocation d'un bloc de mémoire centrale.

Entrée : 4 octets qui contiennent le nombre d'octets à allouer. Si ce nombre vaut -1 (FFFFFFF) alors le système alloue le maximum de mémoire disponible.

Sortie : D0.L contient l'adresse de départ du bloc de mémoire alloué ou 0 si l'opération échoue.

Numéro : 49

Nom : M_FREE

Fonction : Libère une tranche de mémoire précédemment allouée.

Entrée : 4 octets qui contiennent l'adresse de départ du bloc à libérer.

Sortie : D0.L contient -40 si l'adresse du bloc est invalide.

Numéro : 4A

Nom : M_SHRINK

Fonction : Rétrécit un espace mémoire alloué précédemment.

Entrée : Les deux premiers octets doivent valoir 00. Les 4 octets suivants doivent contenir l'adresse du bloc de mémoire à rétrécir. Les 4 derniers octets doivent contenir la nouvelle longueur du bloc.

Sortie : D0.L peut contenir un code d'erreur éventuel -40 (adresse de bloc invalide) ou -67 (rétrécissement impossible).

Numéro : 4B

Nom : P_EXEC

Fonction : Chargement et exécution d'un processus.

Entrée : Octets 1 et 2 : paramètre de chargement
1 = charge et exécute ,
3 = charge et n'exécute pas
Octets 3 à 6 : Adresse d'une chaîne de caractères terminée par 00 qui contient le descripteur (unité, chemin, nom) du fichier à charger.

TABLE DES FONCTIONS DU BDOS

Octets 7 à 10 : Adresse d'un tampon qui contient les informations de redirection. Cette adresse est placée à l'octet d'OFFSET 80H par rapport à la page de base du chargement.

Octets 11 à 14 : Adresse d'une suite de chaînes qui contient la configuration de l'environnement du processus. Ces chaînes sont séparées par un octet à 00 et terminées par deux octets à 00.

Sortie : D0.L contient un code d'erreur éventuel -33 (fichier non trouvé), -39 (mémoire insuffisante) ou -66 (format du programme à charger invalide).

Numéro : 4C

Nom : P_TERM

Fonction : Termine le processus courant et retourne au processus appelant en fermant tous les fichiers ouverts par le processus courant.

Entrée : 2 octets qui contiennent un code qui sera transmis au processus appelant.

Sortie : D0.L peut contenir un code d'erreur si l'opération échoue.

Numéro : 4D

Non utilisée

Numéro : 4E

Nom : F_SFIRST

Fonction : Recherche de la première entrée du répertoire qui correspond au nom spécifié avec les attributs spécifiés.

Entrée : Les 4 premiers octets contiennent l'adresse d'un tampon qui contient une chaîne de caractères terminée par 00 et qui contient la description du fichier (unité, chemin d'accès, nom). Cette description peut contenir une carte de sélection. Les 2 octets suivants contiennent un descripteur d'attribut de recherche défini comme suit :

- 00 = Fichier standard.
- 01 = Fichier à lecture seule.
- 02 = Fichier caché.
- 04 = Fichier système.
- 08 = Identificateur de volume.
- 10 = Sous-répertoire.
- 20 = Archive.

TABLE DES FONCTIONS DU BDOS

Si l'attribut vaut 0, la recherche est limitée aux fichiers standards. Si la fonction trouve une entrée qui correspond aux spécifications, elle formate une DTA et la fonction 2F (F_GETDTA) peut retourner l'adresse de cette DTA.

Sortie : D0.L contient un code d'erreur éventuel -33 (fichier non trouvé) ou -49 (pas d'autre fichier).

Numéro : 4F

Nom : F_SNEXT

Fonction : Recherche après l'occurrence suivante d'un descripteur de fichier.

Entrée : Voir fonction 4E (F_SFIRST).

Sortie : D0.L peut contenir un code d'erreur -49 (pas d'autre fichier).

Numéros : 50 à 55

Non utilisée

Numéro : 56

Nom : F_RENAME

Fonction : Change le nom d'un fichier (RENAME).

Entrée : Les 2 premiers octets doivent être 00.
Les 4 octets suivants pointent sur une chaîne de caractères terminée par 00 et qui contient le descripteur (unité, chemin, nom ...) du fichier original.
Les 4 octets suivants pointent sur une chaîne de caractères terminée par 00 qui contient le nouveau descripteur du fichier.

Sortie : D0.L contient un code d'erreur éventuel -34 (chemin non trouvé) ou -36 (accès refusé si le nom du fichier cible existe déjà par exemple).

Numéro : 57

Nom : F_DATIME

Fonction : Lecture ou écriture de la date et de l'heure d'estampillage d'un fichier (STAMP).

TABLE DES FONCTIONS DU BDOS

Entrée : Les 4 premiers octets pointent sur un tampon qui contient la date dans le premier mot et l'heure dans le second. Les deux octets suivants contiennent le descripteur du HANDLER du fichier. Les deux derniers contiennent un sémaphore d'opération (0= positionnement et 1=lecture).

Dans le cas de la lecture, la date et l'heure lues sont mises dans le tampon.

Sortie : En lecture le tampon contient l'heure et la date lues.

FONCTIONS DU BIOS

Les fonctions du BIOS se divisent en deux classes : les fonctions du BIOS standard et les fonctions du BIOS étendu.

Les fonctions du BIOS sont appelables depuis le mode utilisateur du 68000. Ses appels sont réentrants à trois niveaux. Il peut donc y avoir trois appels récursifs du BIOS avant que le système se "plante".

Les fonctions du BIOS standard sont exécutées par l'appel de l'instruction TRAP #13, celles du BIOS étendu par l'appel de l'instruction TRAP #14.

La méthode de passage des paramètres est semblable à celle expliquée dans la description des fonctions du BDOS.

Les résultats ou les codes d'erreurs sont transmis par l'intermédiaire du registre D0.

Les appels de fonctions du BIOS détruisent les registres D0, D1, D2, A0, A1 et A2.

CODES D'ERREURS INTERNES DES FONCTIONS DU BIOS

<i>Code</i>	<i>Fonction</i>
0	Pas d'erreur.
- 1	Erreurs diverses.
- 2	Périphérique non prêt, non attaché ou occupé trop longtemps.
- 3	Commande inconnue du périphérique.
- 4	Erreur de contrôle de CHECKSUM (CRC).
- 5	Le périphérique ne peut pas traiter la commande qui pourrait être correcte dans un autre contexte.
- 6	Le disque ne peut atteindre la piste désignée.
- 7	Essai de lecture d'un média d'un format étranger ou abîmé.
- 8	Secteur non trouvé.
- 9	Il n'y a pas de papier dans l'imprimante.
-10	Erreur pendant une opération d'écriture.
-11	Erreur pendant une opération de lecture.
-12	Code réservé pour usage ultérieur.
-13	Le disque est protégé en écriture.
-14	Le disque a été changé depuis la dernière écriture.
-15	Le périphérique désigné est inconnu du BIOS
-16	Le secteur traité est d'un format incorrect.
-17	Disque absent. Le système demande l'insertion du disque.

TABLE DES FONCTIONS DU BIOS STANDARD

Numéro : 0

Nom : GETMPB

Fonction : Construction d'un MPB (Bloc Pointeur Mémoire).

Entrée : 4 octets qui contiennent l'adresse d'un tampon qui contiendra le MPB en sortie.

Sortie : L'adresse fournie en entrée contient un MPB composé de 12 octets définis comme suit :

Octets 0 à 3 : MP_MFL. pointeur vers une zone MD (Descripteur Mémoire) de mémoire libre définie ci-dessous.

Octets 4 à 7 : MP_MAL Pointeur vers une zone MD de mémoire occupée (Vaut 0).

Octets 8 à B : MP_ROVER Pointeur vers une zone MD identique à la première.

Structure de la zone MD

La zone MD est une zone de 16 octets contenant 4 adresses.

Octets 0 à 3 : M_LINK adresse du MD suivant (non autorisé pour l'instant. Vaut 0).

Octets 4 à 7 : M_START adresse de la mémoire libre.

Octets 8 à B : M_LENGTH longueur de la mémoire libre.

Octets C à F : M_OWN descripteur (vaut 0).

Numéro : 1

Nom : BCONSTAT

Fonction : Retourne l'état d'un périphérique d'entrée.

Entrée : 1 mot de 16 bits qui contient le numéro du périphérique.

0 = PRT : (Imprimante parallèle).

1 = AUX : (Port série RS232).

2 = CON : (Console écran).

3 = MID : (Interface musicale MIDI).

4 = IKBD : (Port de commande du clavier).

L'état d'entrée peut être utilisé pour tester les périphériques 1 à 3.

Sortie : D0.I. contient 0 si il n'y a pas de caractère disponible en entrée ou -1 (FFFFFFFF) s'il y a au moins un caractère disponible.

Numéro : 2

Nom : BCONIN

Fonction : Lecture avec attente d'un caractère en provenance d'un périphérique.

Entrée : Un mot de 16 bits qui contient le numéro du périphérique. Cette fonction est utilisable avec les périphériques 0 à 4.

Sortie : D0.L contient le caractère entré sur son octet de poids faible. Si le périphérique est la CONSOLE (2), la fonction retourne le code clavier IBM compatible dans l'octet bas du mot le plus significatif et le code ASCII dans l'octet le moins significatif.

Numéro : 3

Nom : BCONOUT

Fonction : Ecriture d'un caractère sur un périphérique.

Entrée : Un mot de 16 bits qui contient le numéro du périphérique suivi d'un mot de 16 bits dont l'octet bas contient le code du caractère à écrire.

Sortie : Rien.

Numéro : 4

Num : RWABS

Fonction : Lecture ou écriture de secteurs sur l'unité disque.

Entrée : Octets 0 et 1 : Mots de 16 bits contenant le code de l'opération :

0 = Lecture de secteurs

1 = Ecriture de secteurs

2 = Lecture de secteurs sans tenir compte du changement éventuel de disquette

3 = Ecriture de secteurs sans tenir compte du changement éventuel de disquette

Octets 2 à 5 : Adresse du tampon qui contient les informations à écrire ou qui contiendra les informations à lire.

Octets 6 et 7 : Mot de 16 bits qui contient le nombre de secteurs à lire ou à écrire.

Octets 8 et 9 : Mot de 16 bits qui contient le numéro du premier secteur à écrire ou à lire.

Octets A et B : Mot de 16 bits qui contient le numéro de l'unité disque : 0 = disque A, 2 = disque B...

TABLE DES FONCTIONS DUBIOS STANDARD

Sortie : D0.L. contient 0 si l'opération s'est déroulée correctement. Une valeur négative indique une erreur (voir table).

Numéro : 5

Nom : SETEXC

Fonction : Permet de modifier la valeur de l'adresse de traitement d'un vecteur d'exception.

Entrée : Un mot de 16 bits qui contient le numéro du vecteur suivi d'une adresse sur quatre octets qui contient l'adresse du vecteur. Si on veut simplement connaître l'adresse d'un vecteur sans le modifier, il faut mettre -1 dans l'adresse.

Sortie : D0.L. contient l'ancienne adresse du vecteur ou l'adresse courante si l'adresse fournie vaut -1.

Numéro : 6

Nom : TICKCAL

Fonction : Fournit le nombre de millisecondes passées depuis l'initialisation du système.

Entrée : Rien.

Sortie : D0.L. contient le nombre de millisecondes écoulées.

Numéro : 7

Nom : GETBPB

Fonction : Fournit l'adresse du BPB (Bloc de Paramètres du BIOS).

Entrée : Un mot de 16 bits qui contient le numéro du disque.

Sortie : D0.L. contient l'adresse du BPB ou 0 si le BPB ne peut être déterminé.

Numéro : 8

Nom : BCOSTAT

Fonction : Fournit l'état d'un périphérique en sortie.

Entrée : Un mot de 16 bits qui contient le numéro du périphérique (voir fonction 1).

Sortie : D0.L. contient 0 si le périphérique n'est pas prêt ou -1 si le périphérique est prêt.

Numéro : 9

Nom : MEDIACH

Fonction : Détection du changement de média.

Entrée : Un mot de 16 bits qui contient le numéro du disque.

Sortie : D0.L contient 0 si le média n'a pas été changé, 1 si le média a peut-être été changé, 2 si le média a été changé.

Numéro : A

Nom : DRVMAP

Fonction : Fournit une table des disques utilisables.

Entrée : Rien.

Sortie : D0.L contient la table des disques existants.
 B0 représente le disque A, B1 le disque B...
 Si le bit considéré vaut 0, le disque n'existe pas.
 Si le bit considéré vaut 1, le disque existe.

Numéro : B

Nom : KBSHIFT

Fonction : Positionne le clavier dans un mode particulier.

Entrée : Un mot de 16 bits. Si le mot est positif, la fonction positionne le clavier dans le mode indiqué par les bits 0 à 7 du mot. Si le mot est négatif, la fonction retourne l'état du clavier sans le modifier.

Sortie : L'octet le moins significatif de D0.L contient l'état du clavier.

Assignment des bits

- Bit 0 = Touche SHIFT droite.
- Bit 1 = Touche SHIFT gauche.
- Bit 2 = Touche CONTROL.
- Bit 3 = Touche ALT.
- Bit 4 = Touche CAPS LOCK.
- Bit 5 = Bouton droit de la souris.
- Bit 6 = Bouton gauche de la souris.
- Bit 7 = réservé.

TABLES DES FONCTIONS DU BIOS ETENDU

Numéro : 0

Nom : INITMOUS

Fonction : Initialise le gestionnaire de la souris. Voir chapitre 1 pour toutes les explications nécessaires à la bonne compréhension de cette fonction

Entrée : Octets 0 et 1 : Mot de 16 bits qui contient le type d'action.
0 = Désactivation de la souris.
1 = Activation de la souris en mode relatif.
2 = Activation de la souris en mode absolu.
3 = Inutilisé.
4 = Activation de la souris en mode clavier.

Octets 2 à 5 : Pointeur vers un bloc de paramètres défini comme suit :
1 octet indiquant si la souris est en 0 en haut (1) ou en bas (0).
1 octet pour définir le mode de fonctionnement des boutons de la souris.
1 octet définissant le facteur xparam.
1 octet définissant le facteur yparam.
D'autres octets peuvent compléter ce bloc en fonction du mode choisi.

Octets 6 à 9 : Contiennent l'adresse d'une routine lancée par la fonction rapport de souris (MOUSE REPORT) du processeur clavier.

Numéro : 1

Nom : SSBRK

Fonction : Réserve un certain nombre d'octets en haut de mémoire. Cette fonction est prévue pour être appelée avant l'initialisation du DOS.

Entrée : Un mot de 16 bits qui contient le nombre d'octets à réserver.

Sortie : D0.L contient l'adresse à partir de laquelle les octets sont réservés.

Numéro : 2

Nom : PHYSBASE

Fonction : Fournit l'adresse physique de la mémoire écran.

TABLE DES FONCTIONS DU BIOS ETENDU

Entrée : Rien.

Sortie : D0.L contient l'adresse physique de la mémoire écran.

Numéro : 3

Nom : LOGBASE

Fonction : Fournit l'adresse logique de l'écran.

Entrée : Rien.

Sortie : D0.L contient l'adresse logique de la mémoire écran.

Numéro : 4

Nom : GETREZ

Fonction : Fournit le mode résolution écran.

Entrée : Rien.

Sortie : D0.L contient 0, 1 ou 2.

Numéro : 5

Nom : SETSCREEN

Fonction : Positionne les paramètres écran. Les paramètres négatifs ne modifient pas l'état actuel des choses.

Entrée : Les quatre premiers octets contiennent l'adresse logique de la mémoire écran, les quatre suivants son adresse physique et les deux derniers contiennent le mode résolution (0, 1 ou 2).

Sortie : Rien.

Numéro : 6

Nom : SETPALETTE

Fonction : Positionne le contenu du registre de palette (pour les 16 couleurs).

Entrée : 4 octets qui pointent sur une zone de 16 mots de 16 bits qui définissent les 16 couleurs. Pour le format d'un mot de 16 bits définissant une couleur, reportez-vous au chapitre 1.

Sortie : Rien.

TABLE DES FONCTIONS DU BIOS ETENDU

Numéro : 7

Nom : SETCOLOR

Fonction : Positionne une des couleurs.

Entrée : Le premier mot de 16 bits définit le numéro de la couleur (0 à 15) et le second définit la couleur.

Sortie : D0.W (16 bits) contient l'ancienne couleur.

Numéro : 8

Nom : FLOPRD

Fonction : Lecture d'un ou plusieurs secteurs de l'unité disque souple.

Entrée : Octets 0 à 3 : Adresse du tampon qui recevra le résultat de la lecture. Le tampon doit être suffisamment grand et commencer sur une adresse paire.

Octets 4 à 7 : Inutilisés.

Octets 8 et 9 : Numéro d'unité (0 ou 1).

Octets A et B : Numéro du premier secteur à lire.

Octets C et D : Numéro de la piste où commence la lecture.

Octets E et F : Numéro de la face du disque (0 ou 1).

Octet 10 et 11 : Nombre de secteurs à lire.

Sortie : D0.L contient un compte rendu d'erreur ou 0 si l'opération est réussie.

Numéro : 9

Nom : FLOPWR

Fonction : Ecriture d'un ou de plusieurs secteurs sur le disque souple.

Entrée : Les paramètres sont identiques à ceux de la fonction 8 mais pour une écriture bien sûr.

Sortie : D0.L contient un compte rendu d'erreur ou 0 si l'opération est réussie.

Remarque : Si on écrit sur le secteur BOOT (secteur 1, piste 0, face 0), le système détectera un changement de média si une fonction RWABS ou MEDIACH est invoquée.

TABLE DES FONCTIONS DU BIOS ETENDU

Numéro : A

Nom : FLOPFMT

Fonction : Formatage d'un disque souple.

Entrée :

- Octets 0 à 3 : Pointeur vers un tampon d'au moins 8K.
- Octets 4 à 7 : Inutilisés.
- Octets 8 et 9 : Numéro du disque souple (0 ou 1).
- Octets A et B : Nombre de secteurs par piste.
- Octets C et D : Numéro de la piste (0 à 79).
- Octets E et F : Numéro de face (0 ou 1).
- Octets 10 et 11 : Facteur d'entrelacement (1 en principe).
- Octets 12 et 15 : Nombre magique 87654321 (hexa).
- Octets 16 et 17 : La valeur qui sera écrite dans chaque octet de chaque secteur formaté (E5E5) est une bonne valeur (1101100111011001).

Sortie : D0.L contient 0 si l'opération est réussie et un code d'erreur dans le cas contraire. Le tampon contient en outre une liste (pas nécessairement triée) des secteurs défectueux.

Remarque : Le formatage positionne le disque en mode forcément changé pour les fonctions RWABS ou MEDIACH.

Numéro : B

Inutilisée

Numéro : C

Nom : MIDIWS

Fonction : Ecriture d'une chaîne de caractères sur le PORT MIDI.

Entrée : Le premier mot de 16 bits contient le nombre de caractères à écrire moins un, les quatre octets suivants pointent sur la chaîne de caractères.

Sortie : Rien.

Numéro : D

Nom : MFPOINT

Fonction : Positionne le vecteur d'interruption interne (0 à 15) du MFP à une adresse précise.

TABLE DES FONCTIONS DU BIOS ETENDU

Entrée : Un mot de 16 bits qui contient le numéro du vecteur suivi de l'adresse à positionner sur 4 octets.

Sortie : Rien.

Numéro : E

Nom : IOREC

Fonction : Fournit un pointeur vers un spécificateur de tampon d'un périphérique d'entrée.

Entrée : Un mot de 16 bits qui contient le numéro de périphérique (0 pour le RS232, 1 pour le clavier et 2 pour le MIDI).

Sortie : Le spécificateur de tampon présente la structure suivante :

- Octets 0 à 3 : Adresse du tampon.
- Octets 4 à 5 : Taille du tampon.
- Octets 6 à 7 : Index de début (prochaine position de traitement).
- Octets 8 à 9 : Index de fin (Si égal à l'index de début, le tampon est vide).
- Octets A à B : Marqueur de niveau inférieur (Autorisation de poursuite de la transmission ou XON).
- Octets C à D : Marqueur de niveau supérieur (Inhibition de poursuite de la transmission ou XOFF).

Si le périphérique est un RS232, un descripteur de tampon de sortie avec la même structure suit le descripteur de tampon d'entrée.

Numéro : F

Nom : RSCONF

Fonction : Configuration du PORT RS232.

Entrée : Octets 0 et 1 : Vitesse de transmission (voir tableau suivant)

0 = 19200	1 = 9600
2 = 4800	3 = 3600
4 = 2400	5 = 2000
6 = 1800	7 = 1200
8 = 600	9 = 300
A = 200	B = 150
C = 134	D = 110
E = 75	F = 50

Octets 2 et 3 : Mode de contrôle du flux
0 = Pas de contrôle
1 = XON/XOFF

TABLE DES FONCTIONS DU BIOS ENTENDU

2 = RTS/CTS

3 = XON/XOFF et RTS/CTS

Octets 4 et 5 : Valeur du registre UCR du MFP

Octets 6 et 7 : Valeur du registre RSR du MFP

Octets 8 et 9 : Valeur du registre TSR du MFP

Octets A et B : Valeur du registre SCR du MFP

Sortie : Rien.

Numéro : 10 (hexa)

Nom : KEYTBL

Fonction : Positionne les pointeurs vers les tables de transcodage du clavier.

Entrée : Les 4 premiers octets pointent vers la table de transcodage des touches normales, les 4 suivants vers la table de transcodage des touches utilisées avec SHIFT, et les 4 derniers vers la table de transcodage des touches lorsque le CAPS LOCK est positionné.

Sortie : Retourne dans D0.L l'adresse d'une table qui contient les adresses successives des trois pointeurs. Chaque table présente une longueur de 128 octets.

Numéro : 11 (hexa)

Nom : RANDOM

Fonction : Retourne un nombre pseudo-aléatoire.

Entrée : Rien.

Sortie : D0.L contient un nombre aléatoire sur 24 bits. L'octet le plus significatif vaut 0.

Numéro : 12 (hexa)

Nom : PROTOBT

Fonction : Construit une image prototype du secteur BOOT.

Entrée : Octets 0 à 3 : Adresse d'un tampon de 512 octets.
Octets 4 à 7 : 4 octets qui contiennent un numéro de série.
Octets 8 et 9 : Type du disque ou -1 si le type ne doit pas être modifié.

TABLE DES FONCTIONS DU BIOS ENTENDU

0 = 40 pistes simple face

1 = 40 pistes double face

2 = 80 pistes simple face

3 = 80 pistes double face

Octets A et B : Sémaphore d'exécution (1 = exécutable, 0 = inexécutable).

Sortie : Rien.

Numéro : 13 (hexa)

Nom : FLOPVER

Fonction : Vérifie un ou plusieurs secteurs du disque.

Entrée : Identique à la fonction 8 (lecture secteurs).

Sortie : Identique à la fonction 8 (lecture secteurs).

Numéro : 14 (hexa)

Nom : SCRDMF

Fonction : Produit une copie de l'écran sur imprimante.

Entrée : Rien.

Sortie : Rien.

Numéro : 15 (hexa)

Nom : CURSCONF

Fonction : Lecture ou positionnement des attributs et de la vitesse de clignotement du curseur.

Entrée : Un mot de 16 bits indiquant la vitesse de clignotement ou -1 si cette vitesse doit être lue. Ce mot est suivi d'un autre mot de 16 bits indiquant les attributs du curseur (0 = Clignotement, 1 = curseur valide, 2 curseur inhibé) La vitesse de clignotement est basée sur la fréquence de balayage (50 hertz).

Sortie : D0.L contient la vitesse de clignotement lue ou ancienne sur son mot de 16 bits de poids fort et l'attribut courant ou ancien sur son mot de 16 bits de poids faible.

Numéro : 16 (hexa)

Nom : SETTIME

TABLE DES FONCTIONS DU BIOS ENTENDU

Fonction : Positionne la date et l'heure.

Entrée : Un mot de 32 bits qui contient la date sur les 16 bits les plus significatifs et l'heure sur les 16 bits les moins significatifs.

Sortie : Rien.

Numéro : 17 (hexa)

Nom : GETTIME

Fonction : Lecture de la date et de l'heure.

Entrée : Rien.

Sortie : D0.L contient la date et l'heure.

Numéro : 18 (hexa)

Nom : BIOSKEYS

Fonction : Repositionne les valeurs d'initialisation des tables de troncage du clavier.

Entrée : Rien.

Sortie : Rien.

Numéro : 19 (hexa)

Nom : IKBDWS

Fonction : Ecriture d'une chaîne de caractères vers le contrôleur intelligent de clavier.

Entrée : Un mot de 16 bits qui contient le nombre d'octets à écrire suivi de l'adresse de la chaîne sur 4 octets.

Sortie : Rien.

Numéro : 1A

Nom : JDISINT

Fonction : Empêche l'interruption désignée.

Entrée : Un mot de 16 bits qui contient le numéro de l'interruption à empêcher.

Sortie : Rien.

TABLE DES FONCTIONS DU BIOS ENTENDU

Numéro : 1B

Nom : JENABINT

Fonction : Autorise l'interruption désignée.

Entrée : Un mot de 16 bits qui contient le numéro de l'interruption à autoriser.

Sortie : Rien.

Numéro : 1C

Nom : GIACCESS

Fonction : Ecriture ou lecture d'un registre du générateur sonore.

Entrée : Un octet qui contient la donnée à écrire suivi d'un mot de 16 bits qui contient le numéro du registre du PSG. Ce mot doit avoir le bit 7 à 0 pour une opération de lecture et le bit 7 à 1 pour une opération d'écriture.

Sortie : Si l'opération est une lecture : D0.L contient la donnée lue.

Numéro : 1D

Nom : OFFGIBIT

Fonction : Positionne un bit du port A du PSG à zéro.

Entrée : Un mot de 16 bits qui contient le numéro du bit à positionner.

Sortie : Rien.

Numéro : 1E

Nom : ONGIBIT

Fonction : Positionne un bit du port A du PSG à 1.

Entrée : Un mot de 16 bits qui contient le numéro du bit à positionner.

Sortie : Rien.

Numéro : 1F

Nom : XBTIMER

Fonction : Positionne un TIMER du MFP.

TABLE DES FONCTIONS DU BIOS ENTENDU

Entrée : Octets 0 et 1 : Numéro du timer (0 à 3).
Octets 2 et 3 : Numéro du registre du TIMER à positionner.
Octets 4 et 5 : Donnée à écrire dans le registre du TIMER.
Octets 6 à 9 : pointeur vers un HANDLER d'interruption.

Sortie : Rien.

Numéro : 20 (hexa)

Nom : DOSOUND

Fonction : Positionne le "compteur" sonore sur une table.

Entrée : Un mot de 32 bits qui pointe sur l'adresse de la table de commande pour le compteur sonore.

Structure de la table :

Valeur 00 à 0F : indique un numéro de registre, la valeur qui suit ce nombre est à écrire dans le registre.
80 : registre temporaire, la valeur qui suit ce nombre est écrite dans le registre temporaire.
81 : Les trois octets qui suivent ce nombre sont traités de la façon suivante : le premier indique le numéro du registre, le second est la valeur à ajouter au registre temporaire en complément à deux, le troisième est la valeur terminale. Le registre temporaire est chargé dans le registre défini par le premier octet, l'opération est recommencée jusqu'à ce que le registre temporaire atteigne la valeur indiquée par le troisième octet.
82 à FF : L'octet qui suit indique le temps à attendre avant la prochaine mise à jour. Si cette valeur vaut 0, la production sonore est terminée.

Sortie : Rien.

Numéro : 21 (hexa)

Nom : SETPRT

Fonction : Positionne le type d'imprimante.

Entrée : Un mot de 16 bits qui contient le type d'imprimante.
BIT SI = 0 SI = 1

0	Matricielle	Marguerite (DAISY)
1	Couleur	Monochrome

TABLE DES FONCTIONS DU BIOS ENTENDU

2	ATARI	EPSON
3	DRAFT	TEXTE
4	Parallèle	Série
5	Continue	Feuille à feuille
6 à 14	Réservé	Réservé
15	0	-----

Si le mot vaut - 1 (FFFF), la configuration courante est lue.

Sortie : Si le mot d'entrée vaut -1, D0.L. contient la configuration de l'imprimante.

Numéro : 22 (hexa)

Nom : KBDVBASE

Fonction : Fournit un pointeur vers une table de structure de paramètres du gestionnaire intelligent de clavier.

Entrée : Rien.

Sortie : D0.L. pointe sur la table suivante :

Octets 0 à 3	: Adresse d'une routine d'entrée MIDI.
Octets 4 à 7	: Adresse de traitement d'une erreur clavier.
Octets 8 à B	: Adresse de traitement d'une erreur MIDI.
Octets C à F	: Adresse du gestionnaire d'état clavier.
Octets 10 à 13	: Adresse du gestionnaire d'état souris.
Octets 14 à 17	: Adresse du gestionnaire d'état horloge.
Octets 18 à 1B	: Adresse du gestionnaire d'état manette de jeu.

Numéro : 23

Nom : KBRATE

Fonction : Lecture ou écriture de la vitesse de répétition des touches du clavier.

Entrée : Un mot de 16 bits qui contient le délai initial avant répétition suivi d'un mot de 16 bits qui contient la vitesse de répétition en cinquantièmes de seconde. Si un des paramètres vaut 0, sa valeur n'est pas modifiée.

Sortie : D0.L. contient la valeur lue ou l'ancienne valeur dans le cas d'une modification.

Numéro : 24

Nom : PTRBLK

Fonction : Configuration d'écran pour la copie imprimante.

TABLE DES FONCTIONS DU BIOS ENTENDU

Entrée : Un mot de 32 bits qui pointe sur une table de paramètres.

Table :

- Octets 0 à 3 : Adresse de la vidéoram.
- Octets 4 et 5 : OFFSET dans cette vidéoram.
- Octets 6 et 7 : Largeur de l'écran.
- Octets 8 et 9 : Longueur de l'écran.
- Octets A et B : Marge à gauche.
- Octets C et D : Marge à droite.
- Octets E et F : Résolution écran (0 à 2).
- Octets 10 et 11 : Résolution imprimante.
- Octets 12 à 15 : Adresse de la palette de couleur.
- Octets 16 et 17 : Type d'imprimante.
- Octets 18 et 19 : Interface (0=Parallèle , 1 = Série).
- Octets 1A à 1D : Pointeur sur le masque des ombrages.

Sortie : Rien.

Numéro 25 (hexa)

Nom : VSYNC

Fonction : Attend l'arrivée de la prochaine interruption de BLANKING avant de rendre la main. Cette fonction est utilisée pour synchroniser les opérations graphiques avec les temps morts de balayage écran.

Entrée : Rien.

Sortie : Rien.

Numéro : 26 (hexa)

Nom : SUPEXEC

Fonction : Passage en mode superviseur.

Entrée : Un mot de 32 bits qui contient l'adresse de la routine à exécuter en mode superviseur.

Sortie : Rien.

Numéro : 27 (hexa)

Nom : PUNTAES

Fonction : Sortie du mode AES.

Entrée : Rien.

Sortie : Rien.

VARIABLES SYSTEMES INTERNES

La liste des variables qui suit est garantie par ATARI. Elle restera stable de version en version.

Les autres variables (celles qui ne sont pas documentées) peuvent être modifiées sans préavis. Il ne faut donc pas les utiliser pour des applications viables et commercialisables.

<i>Adresse</i>	<i>Long</i>	<i>Nom</i>	<i>Fonction</i>
0400	4	etv_timer	vecteur de maintenance du TIMER.
0404	4	etv_critic	vecteur de maintenance d'erreur.
0408	4	etv_term	vecteur de fin de processus.
040C	20	etv_xtra	espace libre pour 5 vecteurs.
0420	4	memvalid	nombre 752019F3 qui valide memcntl et indique un BOOT correct.
0424	1	memcntl	Contient la configuration mémoire (voir chapitre I).
0426	4	resvalid	Si cette adresse contient 31415926 au RESET, le système saute à la valeur contenue dans resvector.
042A	4	resvector	Si resvalid contient 31415926 alors cette adresse contient l'adresse d'exécution à l'initialisation.
042E	4	phystop	Adresse de la fin physique de la mémoire RAM.
0432	4	membot	Début de la mémoire disponible. La fonction BIOS "getmpb" utilise cette valeur pour calculer l'adresse de départ de la TPA.
0436	4	memtop	Fin de la mémoire disponible. La fonction BIOS "getmpb" utilise cette valeur pour calculer l'adresse de fin de la TPA.
043A	4	memval2	Contient 237698AA pour indiquer une initialisation correcte.
043E	2	flock	Utilisé pour verrouiller l'usage du circuit DMA.

VARIABLES SYSTEMES INTERNES

Adresse	Long	Nom	Fonction
0440	2	seekrate	Temps de déplacement piste à piste du disque souple pour les deux unités suivant la table ci-dessous : 0 = 6 ms 1 = 12 ms 2 = 2 ms 3 = 3 ms (défaut).
0442	2	timr_ms	Calibrage de l'horloge (TIMER) Contient 14H (20) par défaut.
0444	2	fverify	Sémaphore de vérification. S'il est différent de zéro, chaque écriture fait l'objet d'une vérification.
0446	2	bootdev	Contient le numéro du périphérique d'où le système a été initialisé.
0448	2	palmode	Sémaphore standard vidéo 0 = NTSC 60 Hz : 1 = PAL 50 Hz.
044A	1	defshifmd	Résolution vidéo par défaut utilisée lors du passage du mode monochrome en mode couleur.
044C	2	sshiftmd	Mode vidéo : 0 = 320 x 200 1 = 640 x 200 2 = 640 x 400.
044E	4	v_bas_ad	Pointeur vers l'adresse de début de la mémoire écran.
0452	2	vblsem	Sémaphore pour forcer l'exclusion gestion de l'interruption de BLANKING vertical.
0454	2	nvbls	Nombre d'adresses sur lesquelles vblqueuc pointe (8 par défaut).
0456	4	vblqueuc	Pointeur vers une table de nvbls vecteurs.
045A	4	colorptr	Pointeur vers un vecteur de 16 mots de 16 bits contenant les 16 couleurs à initialiser au prochain BLANKING vertical.

VARIABLES SYSTEMES INTERNES

Adresse	Long	Nom	Fonction
045E	4	screenpt	Pointeur vers une nouvelle adresse de mémoire écran qui sera validée au prochain BLANKING vertical. Si la valeur vaut 0, il n'y a pas de changement de valeur d'adresse.
0462	4	vbclock	Compteur d'interruption de BLANKING
0466	4	frclock	Compteur d'interruption de BLANKING verticaux en cours de traitement.
046A	4	hdv_init	Vecteur d'initialisation du disque dur.
046E	4	hdv_dsb	Vecteur vers la routine qui retourne un bloc d'état du disque dur. La pile doit contenir le numéro du périphérique correspondant au disque dur.
0472	4	hdv_bpb	Vecteur vers la routine qui retourne un BPB pour le disque dur.
0476	4	hdv_rw	Vecteur vers la routine de lecture et d'écriture du disque dur.
047A	4	hdv_boot	Vecteur vers la routine de BOOT depuis le disque dur.
047E	4	hdv_mediach	Vecteur vers la routine qui retourne le résultat d'un changement de disque dur.
0482	2	cmdload	Sémaphore. Si différent de 0, le système essaye de charger le programme COMMAND.PRG. Ce mot peut être positionné par le BOOT.
0484	1	conterm	Attributs de la console : bit 2 = 1 = click clavier bit 1 = 1 = répétition touche bit 0 = 1 = autorisation BELL.
04A2	4	savptr	Pointeur vers un espace de sauvegarde des registres pour les fonctions BIOS.
04A6	2	nflpos	Nombre de lecteurs de disques.

VARIABLES SYSTEMES INTERNES

<i>Adresse</i>	<i>Long</i>	<i>Nom</i>	<i>Fonction</i>
04B2	4	buf11	Pointeur vers une liste tampon.
04B6	4	buf12	Pointeur vers une liste tampon.
04BA	4	hz_200	Compteur du timer 200 hertz.
04C2	4	drvbits	Vecteur retourné par la fonction 10H (DRIVEMAP) du BIOS. Si il y a au moins un disque souple attaché, cette valeur vaut 3.
04C6	4	dskbuf	Pointeur vers un tampon de 1K utilisé par certaines opérations graphiques.
04CE	2	prtent	Initialement à -1, ce mot est incrémenté par la pression sur ALT & HELP. Si la valeur vaut 0, elle déclenche une copie de l'écran sur imprimante. Si la valeur est différente de 0, la copie s'arrête.
04F2	4	sysbase	Pointe vers la base de l'OS.
04F6	4	shell_n	Pointe vers le contexte SHELL.

MFP 68901

Le MFP 68901 est un composant faisant partie de la famille du 68000. C'est un périphérique multifonctions (Multi Function Peripheral) contenu dans un boîtier à 48 broches.

Le mot multifonctions n'est pas exagéré au vu de ses performances. Les voici résumées en quelques lignes :

- une interface série intégrée ;
- quatre timers (chronomètres) universels programmables individuellement ;
- un port parallèle de 8 bits ;
- chaque bit du port parallèle est programmable séparément en entrée ou en sortie ;
- ils peuvent chacun être utilisé comme source d'interruption ;
- seize origines d'interruptions sont possibles ;
- possibilité de chaînage des interruptions avec d'autres 68901.

Programmation du circuit

Le MFP 68901 comporte 24 registres différents accessibles par le programmeur. Ce nombre peut paraître élevé, mais leur arrangement logique constitue une aide efficace à la programmation du MFP. Nous allons décrire chacun de ces registres.

Le registre 1 : GPIIP (General Purpose I/O Interrupt Port)

Il donne accès aux huit bits du port parallèle. Les données qui s'y trouvent écrites seront transmises aux bornes de sortie du port. De même, en lecture, il reflète l'état des signaux d'entrée I0 à I7.

Le registre 2 : AER (Active Edge Register)

Lorsque les bits du port parallèle sont utilisés comme entrée d'interruption, ce registre permet de déterminer sur quel type de niveau elle se produira. Si

MFP 68901

le bit est à 1, l'interruption est générée lors du passage d'un niveau bas à un niveau haut, et inversement si le bit est à 0.

Le registre 3 : DDR (Data Direction Register)

La programmation de chaque bit du port parallèle, en entrée ou en sortie, se fait à travers ce registre. Un 1 programme la broche correspondante en sortie, et un 0 en entrée.

Les registres 4 et 5 : IERA, IERB (Interrupt Enable Register)

Ces deux registres de huit bits autorisent ou non chacune des 16 sources d'interruption. Une interruption est permise lorsque le bit correspondant est mis à 1. Chaque bit des deux registres utilise les événements suivants :

IERA :

- bit 7 : bit 7 du port parallèle, priorité la plus élevée
- bit 6 : bit 6 du port parallèle
- bit 5 : timer A
- bit 4 : tampon de réception plein
- bit 3 : erreur de réception
- bit 2 : tampon d'émission vide
- bit 1 : erreur d'émission
- bit 0 : timer B

IERB :

- bit 7 : bit 5 du port parallèle
- bit 6 : bit 4 du port parallèle
- bit 5 : timer C
- bit 4 : timer D
- bit 3 : bit 3 du port parallèle
- bit 2 : bit 2 du port parallèle
- bit 1 : bit 1 du port parallèle
- bit 0 : bit 0 du port parallèle, priorité la plus faible.

Les registres 6 et 7 : IPRA et IPRB (Interrupt Pending Register)

Lorsqu'une interruption s'est produite, le bit correspondant est mis à un dans les registres dont la représentation des bits est la même que pour les registres IER.

Si le système d'interruption vectorisée est utilisé, le bit est automatiquement remis à zéro. Sinon, le bit est remis à zéro en envoyant un octet dont tous les bits sont à un, sauf celui reflétant l'interruption.

Les registres 8 et 9 : ISRA et ISRB (Interrupt In-Service Register)

Le fonctionnement de ce registre dépend de l'état du bit 3 du registre n° 12 (Vector Register). Ce bit sélectionne le mode de travail "Software End of Interrupt" (SEI) ou "Automatic End of Interrupt" (AEI).

Dans le mode AEI, lorsque le processeur reçoit le vecteur d'interruption du MFP, les bits IPR et ISR relatifs à cette interruption sont remis à zéro. De plus, d'autres événements peuvent provoquer une interruption même si la routine précédente n'est pas achevée.

Par contre, si on travaille dans le mode SEI, lors de la demande du vecteur par le processeur, le bit ISR est mis à un. Il sera remis à zéro par l'écriture d'un octet dont seul ce bit sera à zéro. Mais, tant que ce bit reste à un, toutes les interruptions de priorité inférieure ne seront pas traitées.

Les registres 10 et 11 : IMRA et IMRB (Interrupt Mask Register)

Ces deux registres permettent de démasquer séparément les différentes sources d'interruption et de les signaler au registre IPR.

Le registre 12 : VR (Vector Register)

Lorsqu'on travaille en mode d'interruption vectorisée, le 68901 peut générer un vecteur d'interruption différent pour chaque source. Les seize sources sont codées en fonction de leur priorité dans les quatre bits de poids faible du numéro du vecteur. Les quatre bits de poids fort sont recopiés à partir du registre VR. Le bit 3 de ce registre programme le mode "Software End of Interrupt" s'il est mis à un, sinon, le mode "Automatic End of Interrupt" est sélectionné.

Les registres 13 et 14 : TACR et TBCR (Timer A/B Control Register)

Avant d'expliquer l'utilisation de ces registres, examinons les fonctions possibles des timers :

Les timers A et B sont tout à fait semblables. Ils possèdent tous deux un registre de données, un diviseur de fréquence et un compteur. Celui-ci est décrémenté d'une unité lors de chaque impulsion d'horloge. Lorsque le compteur passe de 1 à 0, la sortie du compteur correspondant change d'état et le timer est chargé avec la valeur contenue dans le registre de données. Si la condition a été programmée (le bit IER à 1), une interruption est générée. Le décomptage est synchronisé sur la fréquence de l'entrée XTAL1.

Ce mode est disponible sur les timers C et D également. C'est le mode de délai (Delay Mode).

Un autre mode est le mode de comptage d'événements (Event Count). Il s'utilise sur les timers A et B. Dans ce mode, le MFP 68901 compte les impulsions externes présentes sur les entrées TAI et TBI. La fréquence de comptage ne peut être supérieure au quart de celle de fonctionnement du MFP.

MFP 68901

La particularité de ce mode de programmation réside dans le fait que les entrées TAI et TBI sont reliées à I3 et I4 du port parallèle. Une interruption peut être délivrée par l'intermédiaire d'une impulsion sur l'entrée du timer. Cependant, les bornes I3 et I4 peuvent être utilisées sans la possibilité des interruptions.

Le troisième mode d'utilisation des timers A et B est la mesure de la largeur des impulsions. Les timers peuvent être activés ou non à l'aide des entrées TAI et TBI.

L'utilisation de ces deux registres ne nécessite que l'emploi de cinq des huit bits possibles. Les bits 0 à 3 indiquent le mode de fonctionnement.

Fonction	Bits			
	3	2	1	0
Arrêt du timer	0	0	0	0
Mode délai, le diviseur divise par 4	0	0	0	1
Mode délai, le diviseur divise par 10	0	0	1	0
Mode délai, le diviseur divise par 16	0	0	1	1
Mode délai, le diviseur divise par 50	0	1	0	0
Mode délai, le diviseur divise par 64	0	1	0	1
Mode délai, le diviseur divise par 100	0	1	1	0
Mode délai, le diviseur divise par 200	0	1	1	1
Mode comptage d'événements	1	0	0	0
Mode mesure de largeur d'impulsion, le diviseur divise par 4	1	0	0	1
Mode mesure de largeur d'impulsion, le diviseur divise par 10	1	0	1	0
Mode mesure de largeur d'impulsion, le diviseur divise par 16	1	0	1	1
Mode mesure de largeur d'impulsion, le diviseur divise par 50	1	1	0	0
Mode mesure de largeur d'impulsion, le diviseur divise par 64	1	1	0	1
Mode mesure de largeur d'impulsion, le diviseur divise par 100	1	1	1	0
Mode mesure de largeur d'impulsion, le diviseur divise par 200	1	1	1	1

Le bit 4 de ce registre permet de mettre la sortie du timer correspondant à zéro.

Le registre 15 : TCDCR (Timer C et D Control Register)

Seul le mode de délai est utilisable avec les timers C et D. La programmation de chacun d'eux se réalise à l'aide d'un seul octet, les bits 0 et 2 sont réservés au timer C, tandis que les bits 4 à 6 servent au timer D.

Fonction	Timer C : bits		
	2	1	0
	Timer D : bits		
	6	5	4
Arrêt du timer	0	0	0
Mode de délai, le diviseur divise par 4	0	0	1
Mode de délai, le diviseur divise par 10	0	1	0
Mode de délai, le diviseur divise par 16	0	1	1
Mode de délai, le diviseur divise par 50	1	0	0
Mode de délai, le diviseur divise par 64	1	0	1
Mode de délai, le diviseur divise par 100	1	1	0
Mode de délai, le diviseur divise par 200	1	1	1

Les registres 16 à 19 : TADR, TBDR, TCDR et TDDR (Timer Data Register)

Ces quatre registres contiennent les valeurs à décompter.

Le registre 20 : SCR (Synchronous Character Register)

Lors d'une transmission synchrone, le caractère de synchronisation est chargé dans le registre. Dans le cas d'une réception synchrone, le premier caractère reçu est comparé à ce registre. S'il y a concordance, les caractères suivants sont chargés dans le tampon de réception.

Le registre 21 : UCR (Usart Control Register)

Ce registre permet de fixer les paramètres de la transmission à l'exception de la vitesse. Ces bits sont les suivants :

- bit 0 : non utilisé
- bit 1 : 0 = parité impaire
1 = parité paire
- bit 2 : 0 = pas de parité (le bit 1 est ignoré)
1 = parité correspondant au bit 1
- bits 3 et 4 : programment le nombre de bits de départ et d'arrêt, ainsi que le format désiré.

Bits 3	4	Start bit	Stop bit	Format
0	0	0	0	synchrone
0	1	1	1	asynchrone
1	0	1	1,5	asynchrone
1	1	1	2	asynchrone

- bits 5 et 6 : indiquent la longueur des données à transférer.

Bits 5	6	longueur
0	0	8 bits
0	1	7 bits
1	0	6 bits
1	1	5 bits

- bit 7 : 0 = la fréquence utilisée pour la transmission est celle présente sur les broches RC et TC ;

1 = la fréquence de la broche TC est divisée par 16.

Le registre 22 : RSR (Receiver Status Register)

Le RSR indique l'état de la réception lors d'une transmission série. La fonction des différents bits est expliquée ci-après :

- bit 0 Receiver Enable Bit : lorsque ce bit est à 1, le récepteur fonctionne normalement, sinon, tous les autres sont automatiquement mis à zéro.
- bit 1 Synchronous Strip Enable : ce bit permet de déterminer si un caractère semblable à celui contenu dans le registre SCR a été transféré dans le registre de réception.
- bit 2 Match / Character in progress : dans le cas d'une transmission synchrone, ce bit indique que le caractère synchrone reçu est compatible avec celui contenu dans le SCR. Dans le mode asynchrone, ce bit est mis à un lorsque le bit de départ a été reconnu. Il est mis à zéro par le bit d'arrêt.
- bit 3 Found-Search / Break Detected : dans le mode synchrone, ce bit est mis à un lorsqu'un caractère reçu correspond à celui mémorisé dans le registre SCR. Cette condition peut créer une interruption. Dans le mode de fonctionnement asynchrone, ce bit est mis à un

- lors de la réception d'un BREAK, c'est-à-dire, lorsque la ligne reste à zéro pour une durée supérieure à un caractère normal.
- bit 4 Frame Error : ce type d'erreur se produit lorsqu'un bit de stop n'a pas été détecté.
 - bit 5 Parity Error : cette erreur indique que la parité programmée dans le registre UCR n'a pas été respectée pour le dernier caractère reçu.
 - bit 6 Overrun Error : ce bit est mis à un lorsqu'un caractère reçu est transmis dans le registre de réception alors que celui-ci contient encore un caractère qui n'a pas été lu. Ce bit peut générer une interruption.
 - bit 7 Buffer Full : ce bit est à un si un caractère complet est disponible pour le processeur. Il est remis à zéro par la lecture de l'octet.

Le registre 23 : TSR (Transmitter Status Register)

Comme le RSR fournit des renseignements sur l'état de la transmission, le TSR fournit des renseignements sur l'état de l'émission.

Ses bits ont les fonctions suivantes :

- bit 0 Transmitter Enable. Lorsque ce bit est à zéro, l'émission est désactivée. Les bits 4 et 5 du registre sont mis respectivement à zéro et à un.
- bits 1 et 2 High et Low bit. Ils permettent au programmeur d'activer la sortie émettrice lorsque l'émetteur est désactivé. Si les deux bits sont à zéro, la sortie est en haute impédance. Si seul le bit L (bit 2) est à un, la sortie passe au niveau bas. Si seul le bit H (bit 1) est à un, la sortie passe au niveau haut. S'ils sont tous deux à un, la sortie de l'émetteur est bouclée sur l'entrée du récepteur. C'est le mode LOOP-BACK qui permet de tester l'interface série du 68901. La sortie est alors à l'état un.
- bit 3 Break : ce bit permet d'envoyer une condition de Break sur la ligne de transmission en mode asynchrone seulement.
- bit 4 End of Transmission. Ce bit est mis à un si la partie émettrice est interdite et ce, après transmission complète du dernier caractère.
- bit 5 Auto Turnaround. Si ce bit est à un, le récepteur est automatiquement activé lorsque l'émetteur est mis hors circuit et qu'un dernier caractère éventuel a été transmis.
- bit 6 Underrun Error. Lorsqu'un caractère complet a été envoyé et qu'aucun autre n'est inscrit dans le registre d'émission, ce bit est mis à un.
- bit 7 Buffer Empty. Ce bit à un signifie qu'un caractère a été lu à partir du registre de réception.

Le registre 24 : UDR (Usart Data Register)

Les données à recevoir et à envoyer sont transférées par ce registre.

MFP 68901

Le MPF 68901 dans l'ATARI ST

Les 24 registres du MFP 68901 occupent les adresses mémoire allant de la location \$FFFA00 à la location \$FFFA2F.

Chaque registre occupe la partie basse (octet impair) d'un mot de 16 bits dont voici les adresses :

FFFA00	GPIP	FFFA18	TACR
FFFA02	AER	FFFA1A	TBCR
FFFA04	DDR	FFFA1C	TCDCR
FFFA06	IERA	FFFA1E	TADR
FFFA08	IERB	FFFA20	TBDR
FFFA0A	IPRA	FFFA22	TCDR
FFFA0D	IPRB	FFFA24	TEDR
FFFA0E	ISRA	FFFA26	SCR
FFFA10	ISRB	FFFA28	UCR
FFFA12	IMRA	FFFA2A	RSR
FFFA14	IMRB	FFFA2C	TSR
FFFA16	VR	FFFA2E	UDR

Le port d'entrée/sortie est utilisé de la manière suivante :

- bit 0 : gestion du Centronics ;
- bit 1 : transfert des données de détection d'entrée du RS232 ;
- bit 2 : effacement de l'émetteur d'entrée du RS232 ;
- bit 3 : réservé ;
- bit 4 : interruption du clavier et du MIDI ;
- bit 5 : interruption du FDC et du HDC ;
- bit 6 : indicateur du RS232 ;
- bit 7 : détection de moniteur monochrome.

Le timer A n'est pas utilisé dans l'ATARI. Son entrée timer a été associée à la broche "BUSY" de l'interface Centronics.

Le timer B est utilisé pour compter les lignes de l'écran.

La sortie du timer D est reliée aux entrées TC et RC de séquençement des transmissions liées à l'interface série intégrée du MFP.

Généralités

Le générateur sonore **AY3-8910** de General Instruments, en abrégé **PSG** (Programmable Sound Generator), est un circuit LSI (circuit à haute intégration) qui peut produire une grande variété de sons complexes sous le contrôle d'un programme approprié.

Le **PSG** est un composant relativement facile à interfacer avec n'importe quel système à microprocesseur. Sa flexibilité est telle qu'il est souvent utilisé dans toutes sortes d'applications les plus diverses.

Comme les synthétiseurs musicaux, les alarmes et les signalisations sonores ou les modems utilisent la technique **FSK** (Frequency Shift Keying).

La sortie sonore analogique (c'est-à-dire un signal non digital) est effectuée par l'intermédiaire d'un convertisseur **DAC** (Digital Analog Converter) sur quatre bits ; celui-ci permet une grande variété d'effets sonores.

Une des caractéristiques principales du circuit est qu'une fois ses commandes de génération reçues, il génère lui-même les effets sonores, laissant le processeur libre pour continuer d'autres tâches. Ainsi, le **PSG** peut produire des sons relativement longs en n'affectant en rien la vitesse d'exécution du programme.

Le **PSG** possède trois voies mixables et permet la sortie de trois sons simultanés, la création d'un accord musical simple, majeur, ou mineur.

Le **PSG** est un coprocesseur dont la gestion se fait au moyen de différents registres programmables. Ces registres sont au nombre de 16 et seront décrits en détail dans le présent chapitre.

La structure interne du PSG

Le **PSG** se compose des éléments suivants :

- **GENERATEURS SONORES** : au nombre de trois, ils sont respectivement appelés canal A, canal B et canal C. Ils produisent un signal carré dont la fréquence est programmable.
- **LE GENERATEUR DE BRUIT** : il produit une modulation de fréquence aléatoire.
- **LE MELANGEUR (MIXER)** : il permet de combiner les sorties des trois canaux A, B et C du générateur sonore avec le générateur de bruit.
- **LE CONTROLEUR D'AMPLITUDE** : il fournit au D/A (Digital Analog convert) la possibilité de contrôler l'amplitude par un modèle fixe ou variable. Le modèle d'amplitude fixe est contrôlé par le microprocesseur lui-même, tandis que le modèle d'amplitude variable est obtenu par l'utilisation du générateur d'enveloppe.

GENERATEUR SONORE AY3-8910

- LE GENERATEUR D'ENVELOPPE : il produit un modèle de variation d'amplitude appelé enveloppe qui peut être utilisé pour moduler la sortie de chaque mixer.
- LES CONVERTISSEURS DIGITAUX/ANALOGIQUES (D/A) : les trois convertisseurs D/A produisent un signal de sortie sur 16 niveaux déterminés par le contrôleur d'amplitude.
- LES PORTS D'ENTREE/SORTIE : le générateur sonore AY3-8910 possède deux ports d'entrée/sortie. Ces ports ne jouent aucun rôle dans la production sonore.

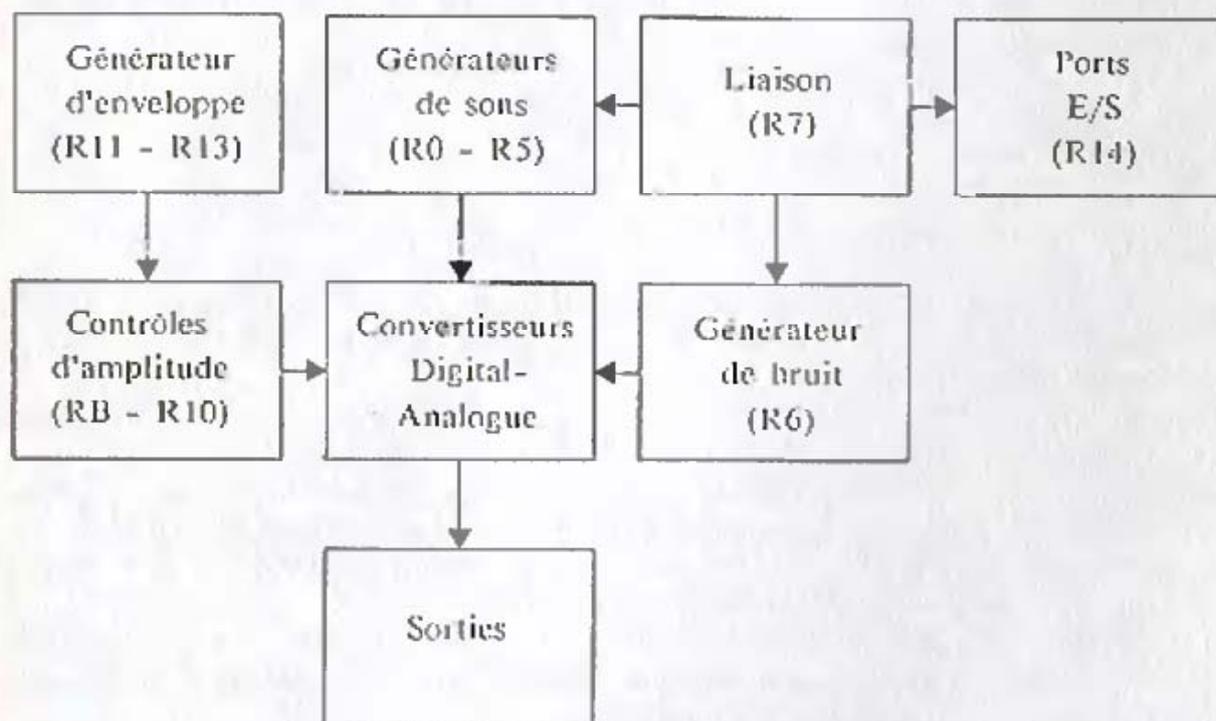
Les différents registres du PSG

Les registres qui permettent la programmation du PSG sont au nombre de 16, numérotés de R0 à R15.

Les registres R14 et R15 servent à la gestion des ports d'entrée/sortie et seront analysés par la suite.

Pour produire un son, une combinaison des registres R0 à R15 doit être introduite dans le PSG. Chaque son doit être analysé de façon à séparer les différents paramètres qui le définissent. C'est-à-dire : la composante du bruit, de son, la fréquence, la forme et la durée des enveloppes.

Une fois cette analyse terminée, les registres peuvent être chargés et le son produit. Le schéma bloc figurant ci-après montre les interactions entre les différentes sections du PSG.



Registres R0 à R5

Les registres R0 à R5 définissent la fréquence du son émis. Ils sont divisés en trois paires : R0-R1 pour le canal A, R2-R3 pour le canal B et R4-R5 pour le canal C.

Les registres R0, R2 et R4 sont les registres de réglage fin de la fréquence et les 8 bits sont utilisés.

Les registres R1, R3 et R5 sont les registres de réglage grossier, seuls les 4 bits de poids faible LSB sont utilisés.

Les valeurs chargées dans les registres R0, R2 et R4 sont donc comprises entre 0 et 15 (00 et FF en hexadécimal).

Dans le PSG, la fréquence d'un son est déterminée en divisant premièrement la fréquence de l'horloge interne (2 MHz) par 16 et puis par F, qui est la fréquence à programmer.

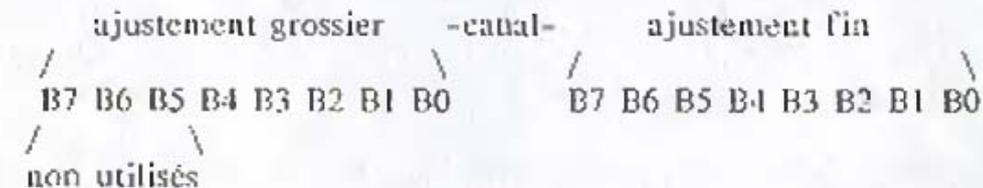
On applique la formule suivante :

$$PT = 2\,000\,000 / (16 * F) \text{ où } PT \text{ est la période de ton.}$$

La valeur résultante des paires de registres étant codée sur 12 bits, PT doit être arrondi à l'unité avant d'être exprimé sur 12 bits au moyen de la fonction :

$$PT = \text{BINS}(F, 12) = \begin{array}{ccc} \text{XXXX} & \text{XXXX} & \text{XXXX} \\ \backslash\text{---}/ & \backslash\text{-----}/ & \\ \text{RH} & & \text{RL} \end{array}$$

Les huit bits de droite sont transmis dans les registres R0, R2 ou R4, et les quatre bits de gauche dans les registres R1, R3 ou R5.



PT11 PT10 PT9 PT8 PT7 TP6 PT5 PT4 TP3 TP2 PT1 PT0

12 bits pour le générateur sonore.

Une autre façon de procéder pour charger les paires de registres consiste à calculer RL et RH comme suit :

$$RL = PT - (\text{INT}(PT/256) * 256) \text{ et}$$

$$RH = \text{INT}(PT/256) \text{ ou encore : } RH = PT \backslash 256 \text{ (C'est } \backslash \text{ et non /)}$$

GENERATEUR SONORE AY3-8910

Il suffit alors de transmettre RL dans R0, R2 ou R4, et RH dans R1, R3 ou R5.

Exemple :

Soit $F = 440$ Hz (le LA international).

PT = $2\,000\,000 / (16 * 440)$
PT = $2\,000\,000 / 7\,040$
PT = 284,092
on arrondit PT = 284
on calcule RL = $284 - (\text{INT}(284/256) * 256)$
RL = $284 - (\text{INT}(1,109) * 256)$
RL = $284 - (1 * 256)$
RL = 28
RH = $\text{INT}(284/256)$
RH = 1

Si c'est le canal A qui doit être programmé, R0 sera égal à 28, et R1 à 1.

Détermination de F minimum et de F maximum

Comme PT (période de ton) est exprimé sur 12 bits, la valeur que l'on peut charger ne peut excéder 4095, c'est-à-dire ($2 \text{ exp } 12 - 1$), et la valeur minimale est égale à 1.

1 donne Fmax et 4095 donne Fmin.

$1 = 2000000 / (16 * F_{\text{max}})$; $F_{\text{max}} = 2000000 / (16 * 1) = 135\,000$ Hz
 $4095 = 2000000 / (16 * F_{\text{min}})$; $F_{\text{min}} = 2000000 / (16 * 4095) = 30,52$ Hz

Il est évident qu'une fréquence Fmax de l'ordre de 135 000 Hz est imperceptible par l'oreille humaine. De plus, la bande passante des petits amplis audio ou de télévision dépasse rarement les 5000 Hz. Dès lors, nous nous fixerons cette valeur comme valeur maximale de fréquence.

PT max = $2\,000\,000 / (16 * 5000) = 25$

Les valeurs de PT seront donc comprises entre 25 et 4095.

Registre R6

Le registre R6 du PSG permet de programmer une fréquence de bruit. Seuls les cinq bits de poids faible du registre R6 sont utilisés pour programmer le générateur de bruit.

Registre de période de bruit :

B7 B6 B5 B4 B3 B2 B1 B0
/ \ / \
non utilisés 5 bits pour le générateur de bruit

GENERATEUR SONORE AY3-8910

La formule appliquée pour trouver la période de bruit est semblable à celle utilisée pour calculer la période de ton (PT).

Soit PB pour période de bruit :

$$PB = 2\ 000\ 000 / (16 * Fb) \text{ où } Fb \text{ est la fréquence de bruit désirée.}$$

Comme les cinq premiers bits de R6 sont utilisés pour déterminer PB, les valeurs de PBmax et de PBmin seront comprises entre 1 et $(2 \exp 5 - 1)$, soit 31.

Par la formule, on détermine Fb max et Fb min :

$$Fb_{max} = 2\ 000\ 000 / (16 * PB_{min})$$

$$Fb_{max} = 125\ 000 \text{ Hz}$$

$$Fb_{min} = 2\ 000\ 000 / (16 * PB_{max})$$

$$Fb_{min} = 4\ 032 \text{ Hz}$$

Il est à noter que les restrictions applicables pour Fmax sont aussi à observer pour Fb max (fréquence de bruit maximale).

Registre R7

Le registre R7 permet de contrôler la fonction de mixage entre les trois générateurs sonores et les trois générateurs de bruits. Le mixage autorise la combinaison de chaque canal A, B ou C avec un générateur de bruits. Le registre R7 contrôle également les ports d'entrée/sortie A et B dont nous parlerons par la suite.

Remarque : nous rappelons que dans l'AY3-8910, seul le port A existe.

Registre de contrôle du mixage :

	B7 B6	B5 B4	B3	B2 B1 B0
I/O PORT	B A	C B	A	C B A
fonction	-----	-----	-----	-----
	INPUT ENABLE	NOISE ENABLE		TONE ENABLE

Remarques : mettre un canal sur OFF ne suffit pas pour arrêter l'émission de celui-ci. Il faut écrire 0 dans le registre de contrôle d'amplitude (voir ci-dessous). ATTENTION, les bits du registre R7 sont actifs bas.

Exemple

Je désire produire sur le canal A du son et pas de bruit ; sur le canal B, du bruit et pas de son et sur le canal C, du bruit et du son.

Je dois donc charger le registre R7 avec la valeur suivante :

Valeur décimale :	128	64	32	16	8	4	2	1
BIT :	B7	B6	B5	B4	B3	B2	B1	B0
	X	X	0	0	1	0	1	0

X = sans importance.

R7 = 10 en base 10 ou R7 = 0A11 en base 16.

GENERATEUR SONORE AY3-8910

Tableau résumant les effets du registre R7 :

Bit	Bit = 0	Bit = 1
B0	SON sur le canal A (ON)	SON sur le canal A (OFF)
B1	SON sur le canal B (ON)	SON sur le canal B (OFF)
B2	SON sur le canal C (ON)	SON sur le canal C (OFF)
B3	BRUIT sur le canal A (ON)	BRUIT sur le canal A (OFF)
B4	BRUIT sur le canal B (ON)	BRUIT sur le canal B (OFF)
B5	BRUIT sur le canal C (ON)	BRUIT sur le canal C (OFF)
B6	PORT A en entrée	PORT A en sortie
B7	PORT B en entrée	PORT B en sortie

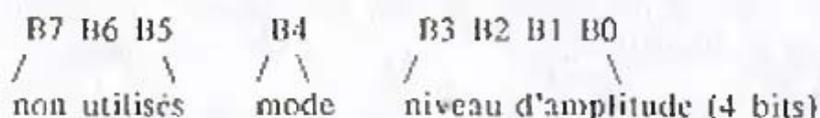
Registres R8 à R10

L'amplitude du signal sonore émis par les trois convertisseurs D/A (un pour chacun des canaux A, B et C) est déterminée par le contenu du registre R8 pour le canal A, du registre R9 pour le canal B et du registre R10 pour le canal C.

Seuls les quatre bits les moins significatifs (B3-B0) de chacun des registres sont utilisés. Ils permettent donc les valeurs comprises entre 0 et 15. La valeur 0 chargée dans l'un des registres correspond à un volume (nul). L'amplitude maximale est obtenue en chargeant le registre avec la valeur 15. Le cinquième bit (B4) est utilisé pour sélectionner le mode de fonctionnement du contrôle d'amplitude.

Si le bit B4 est égal à 0, l'amplitude ne varie pas. Si B4 est égal à 1, la variation d'amplitude est confiée au générateur d'enveloppe (voir ci-après).

Registre de contrôle d'amplitude



Registres R11 et R12

Les registres R11 et R12 permettent au PSG de contrôler la période de l'enveloppe. Un calcul similaire à celui des registres R0 à R5 fournit les valeurs à charger dans les registres R11 et R12.

Soit PE pour la période d'enveloppe et Fe pour la fréquence d'enveloppe :

$$PE = 2\ 000\ 000 / (256 * Fe)$$

GENERATEUR SONORE AY3-8910

Les 8 bits des registres R11 et R12 sont utilisés. La valeur résultante est donc codée sur 16 bits et peut varier de 0 à 65535 ($2 \text{ exp } 16 - 1$).

Le registre R11 du PSG définit l'ajustement fin de la période d'enveloppe tandis que R12 permet un ajustement grossier.

Ajustement grossier R12								Ajustement fin R11							
B7	B6	B5	B4	B3	B2	B1	B0	B7	B6	B5	B4	B3	B2	B1	B0
PE15 à PE8								PE7 à PE0							

PE (16 bits pour le générateur d'enveloppe).

Par la formule, on peut calculer $P_e \text{ max}$ et $P_e \text{ min}$ des enveloppes possibles :

$$F_e = 2\,000\,000 / (256 * PE)$$
$$F_e = 1/P_e = 2\,000\,000 / (256 * PE)$$
$$P_e = (256 * PE) / 2\,000\,000$$

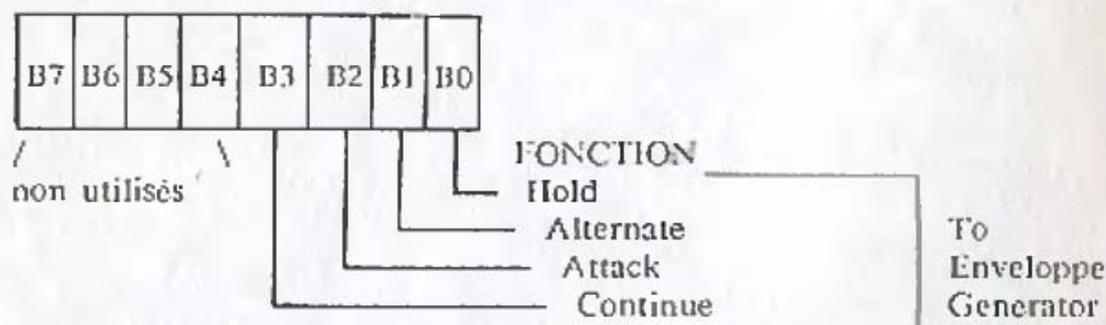
$P_e \text{ max} = (256 * 65535) / 2\,000\,000 = 8,389$ secondes, ce qui correspond à une fréquence d'enveloppe $F_e = 0,0298$ Hz.

$P_e \text{ min} = (256 * 1) / 2\,000\,000 = 0,000\,128$ secondes, ce qui correspond à une fréquence d'enveloppe $F_e = 1\,953,145$ Hz.

Registre R13

Le registre R13 contrôle les différentes formes de modulation utilisées par le PSG. Si le bit (B4) décrit dans les registres R8 à R10 est à 1, la modulation a lieu ; sinon, la programmation du registre 13 est ignorée. Seuls les quatre bits les moins significatifs sont utilisés. Ils déterminent chacun un paramètre de modulation.

R13 : contrôle de forme de modulation :



Registres R14 et R15

Nous avons vu, lors de l'étude du registre R7, que les bits B6 et B7 définissent le sens de la transmission des données (0 = entrée, 1 = sortie) sur les ports d'entrée/sortie.

GENERATEUR SONORE AY3-8910

Le registre R14 est utilisé pour l'écriture et la lecture du PORT A. Le registre R15 est utilisé pour l'écriture et la lecture du PORT B.

Configuration du PSG dans le système ST

Les 16 registres du PSG ne sont pas directement accessibles, l'accès doit se faire par l'intermédiaire du registre de sélection de registres situé à l'adresse FF8800 en y introduisant le numéro du registre à lire ou à écrire (0 à 15).

Cette même adresse sert d'adresse de lecture pour les PORTS A et B commandés par les registres 14 et 15.

Enfin, l'adresse FF8802 permet d'écrire dans un registre précis ou dans les PORTS A et B.

Structure des PORTS

Port A

- BIT 0 : Sélection de la face pour l'unité de disque souple.
- BIT 1 : Sélection de l'unité disque souple 0.
- BIT 2 : Sélection de l'unité disque souple 1.
- BIT 3 : Commande du signal RTS du RS232.
- BIT 4 : Commande du signal DTR du RS232.
- BIT 5 : STROBE (écriture) du port imprimante parallèle.
- BIT 6 : Sortie libre.
- BIT 7 : Réservee.

Port B

Le PORT B est utilisé comme PORT parallèle bidirectionnel. En sortie, il sert principalement à la commande de l'imprimante parallèle de type Centronics.

CONTROLEUR DE DISQUE WD 1772

Le WD 1772 est un composant LSI de la firme Western Digital contenant un contrôleur de disquettes. Au point de vue programmation, il est compatible avec ses prédécesseurs de la série FD 179X/279X. Il possède un séparateur de données et une logique d'écriture et de précompensation intégrés. Il est capable de supporter des formats de données aux standards IBM 3740 (simple densité FM) et IBM 34 (double densité MFM).

Description générale

Le contrôleur de disquettes comporte divers registres internes réalisant l'interface avec le processeur et la disquette.

Registre de sérialisation des données

Il met les données en forme pour les envoyer vers la disquette et inversement, de la disquette vers le processeur.

Registre de données

Reçoit les données du registre de sérialisation lors d'une lecture de la disquette ou les lui transmet lors d'une écriture. Lors d'une commande de recherche de piste, ce registre conserve l'adresse de la piste désirée.

Registre de piste

Il contient la position courante de la tête de lecture/écriture. Il est incrémenté ou décrémenté par pas de un, en fonction du déplacement de la tête. Son contenu est comparé avec le numéro de la piste de l'enregistrement lors des opérations de lecture, d'écriture ou de vérification. Il est accessible en lecture et en écriture.

Registre de secteur

Il contient le numéro du secteur recherché lors des opérations de lecture, d'écriture ou de vérification. Il peut également être lu et chargé.

Registre de commande

Il contient la commande en cours d'exécution. Le contrôleur peut réaliser 11 fonctions différentes que nous détaillerons par la suite. Il ne peut être lu.

Registre d'état (status)

Il fournit des renseignements sur les différents états du contrôleur. Ce registre ne peut être que lu. La signification de chacun de ses bits est la suivante :

B7	MOTOR ON	reflète l'état de la broche MO.
B6	WRITE PROTECT	indique que la disquette est protégée en écriture.

CONTROLEUR DE DISQUE WD 1772

- B5 RECORD TYPE/SPIN-UP s'il est à 1, la séquence de mise en marche du moteur est terminée (6 révolutions) pour la commande de type 1. Pour la commande de type 2 et 3, il donne une indication sur le type d'enregistrement.
- B4 RECORD NOT FOUND ce bit signifie que la piste, le secteur ou la face n'a pas été trouvé.
- B3 CRC ERROR si B4 est mis à 1, B3 indique une erreur dans un champs d'identification. Sinon, l'erreur se trouve dans le champs des données.
- B2 LOST DATA/TRACK00 le processeur n'a pas répondu assez rapidement au signal DRQ. Pour une commande de type 1, ce bit reflète l'état de la broche TRACK00.
- B1 DATA REQUEST INDEX ce bit fournit l'état de la broche DRQ et, pour la commande de type 1, celui de la broche IP.
- B0 BUSY quand il est à 1, une commande est en cours d'exécution.

Logique du CRC

Il génère et contrôle le code CRC (Cyclic Redundancy Check) qui est un polynôme :

$$(X^{16} + X^{12} + X^5 + 1)$$

Unité arithmétique et logique

Elle réalise les opérations de comparaison, de décrémentation et d'incrémement sur les différents registres.

Le processeur accède à certains registres au moyen des signaux A0, A1 et R/ \bar{W} :

A1	A0	READ (R/ \bar{W} = 1)	WRITE (R/ \bar{W} = 0)
0	0	registre d'état	registre de commandes
0	1	registre de piste	registre de piste
1	0	registre de secteur	registre de secteur
1	1	registre de données	registre de données

Programmation du contrôleur

Le contrôleur de disquettes accepte onze commandes différentes qui peuvent être divisées en quatre groupes. Ces commandes sont chargées dans le registre de commandes lorsque le bit "BUSY" du registre d'état est à zéro.

CONTROLEUR DE DISQUE WD 1772

Type	Commande	
I	Restore	: recherche de la piste 0
I	Seek	: recherche de piste
I	Step	: avance sur une piste dans la direction précédente
I	Step-In	: déplacement d'une piste vers le centre du disque
I	Step-Out	: déplacement d'une piste vers l'intérieur du disque
II	Read Sector	: lire un secteur
II	Write sector	: écrire sur un secteur
III	Read address	: lire l'identificateur
III	Read track	: lire une piste entière
III	Write track	: écrire sur toute une piste
IV	Force Interrupt	: délivrer une interruption

Commandes de type I

Ces commandes sont relatives au positionnement de la tête de lecture/écriture. Les mots servant à programmer les cinq fonctions sont les suivants :

	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0
restore	0	0	0	0	M	V	R1	R0
seek	0	0	0	I	M	V	R1	R0
step	0	0	1	U	M	V	R1	R0
step in	0	1	0	U	M	V	R1	R0
step out	0	1	1	U	M	V	R1	R0

Chaque commande contient des bits variables. Ainsi, les bits R1 et R0 déterminent le temps entre deux impulsions step. Les combinaisons possibles sont les suivantes :

R1	R0	Temps
0	0	2 millisecondes
0	1	3 millisecondes
1	0	5 millisecondes
1	1	6 millisecondes

CONTROLEUR DE DISQUE WD 1772

Le bit **V** constitue le drapeau de vérification (verify flag). Lorsqu'il est positionné (=1) dans une commande, le contrôleur effectue une vérification pour chaque opération. Le numéro de piste contenu dans le premier champ d'identification est comparé avec celui du registre de piste. S'il y a concordance et que le CRC est correct, la commande est alors exécutée. Si le CRC n'est pas valide, le bit 3 du registre d'état est mis à 1.

Le contrôleur commande directement les moteurs du lecteur de disquettes. Il faut attendre un certain temps après le démarrage pour que la vitesse de rotation nominale (300 tours/min) soit atteinte. Cette séquence est appelée le SPIN-UP.

Le bit **M** indique si cette séquence est terminée. S'il est à zéro avant d'exécuter la commande, le contrôleur vérifie l'état de la broche MO (MOTOR ON). Si elle est à zéro, le contrôleur la met à un et il attend six tours avant d'exécuter la commande. Mais si le moteur est déjà en marche (MO=1), l'opération est réalisée sans délai d'attente en supposant que la vitesse nominale est déjà atteinte.

Le bit **U**, lorsqu'il est mis à 1, permet l'enregistrement de chaque modification de la position de la tête dans le registre de piste.

Commandes de type II

Ces commandes concernent les opérations relatives aux secteurs. Elles sont au nombre de deux :

	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0
Read sector	1	0	0	S	M	E	0	0
Write sector	1	0	1	S	M	E	P	A0

Le bit **M** est le même que pour la commande précédente.

Le bit **E** permet de générer un délai d'attente de 30 millisecondes avant d'exécuter la commande. Ce temps permet aux oscillations mécaniques dues aux mouvements de la tête de s'estomper. Si ce bit est à zéro, on ne viendra lire ou écrire qu'un seul secteur.

Lors de la lecture des secteurs, les bits **CR1** et **CR0** doivent être à zéro. Mais, lors d'une écriture, le bit **A0** indique si on doit écrire sur une marque de donnée d'adresse (Data Address Mark) normale ou effacée.

Le bit **P** indique si la précompensation est utilisée ou non lors de l'écriture de données. Elle permet d'augmenter la sécurité au niveau des données sur la piste inférieure de la disquette.

CONTROLEUR DE DISQUE WD 1772

Lorsque $S=1$, plusieurs secteurs sont lus ou écrits de telle façon que le contrôleur incrémente le registre de secteur, après chaque opération.

Si $S \leftrightarrow 0$, seul un secteur est accédé.

Commandes de type III

La commande de lecture de l'identificateur donne des informations sur le prochain champ ID. Celui-ci contient en six octets les informations suivantes :

Adresse piste	Numéro face	Adresse secteur	Longueur secteur	CRC 1	CRC 2
1	2	3	4	5	6

La commande lecture de piste (Read Track) transmet dans le registre de données les octets écrits lors du formatage de la disquette et donc, également, les données qui y étaient présentes et qui ne seront plus accessibles car elles sont écrites entre les secteurs.

La commande écriture de piste formate la piste en vue d'une mémorisation des données.

	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0
Read address	1	1	0	0	M	E	0	0
Read track	1	1	1	0	M	E	0	0
Write track	1	1	1	1	M	E	P	0

Les bits M, E et P ont la même signification que pour les autres commandes.

Commandes du type IV

Ce type ne contient qu'une commande. Elle permet d'interrompre n'importe quelle opération en cours. Les valeurs de ses bits sont les suivantes :

	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0
Force interrupt	1	1	0	1	13	12	0	0

CONTROLEUR DE DISQUE WD 1772

Ils indiquent les conditions qui doivent générer une interruption. Si I2 vaut 1, une interruption est engendrée lors de chaque impulsion d'index pour, par exemple, déterminer la vitesse par logiciel (software).

Lorsque I3 est égal à 1, une interruption est envoyée sans interrompre la commande en cours.

Cette commande permet de terminer une écriture ou une lecture sur plusieurs secteurs consécutivement ou bien de lire le registre d'état.

Contrôleur dans l'ATARI ST

L'interface avec le lecteur de disquettes est réalisée à travers le contrôleur de DMA (Accès Direct Mémoire). Les commandes sont envoyées au contrôleur de disquette en sélectionnant son registre de commandes par le registre de contrôle de mode du DMA.

L'accès au contrôleur se fait donc de manière indirecte.

Deux adresses permettent la programmation du contrôleur :

\$FF8604 : les 8 bits de poids faible de cette adresse donnent accès aux registres du contrôleur. La sélection du registre se fait par le registre de mode du DMA.

\$FF8606 : en lecture, les deux bits de cette adresse donnent des renseignements sur l'état du contrôleur.

Le bit 1, lorsqu'il est à zéro, indique que le compteur de secteurs est à zéro, et inversement s'il est à 1.

Le bit 2 reflète l'état du signal Data request.

Les accès en écriture à cette adresse influent sur le registre de mode du DMA dont certains bits concernent le contrôleur de disquette.

Les bits 1 et 2 actionnent respectivement les broches A0 et A1 et sélectionnent donc les registres du contrôleur.

Le bit 3 à zéro donne accès au WD1772.

Le bit 4 à zéro permet l'accès aux registres internes. S'il est à un, il sélectionne le compteur de secteurs.

Le bit 7 à un accède au WD1772.

Le bit 8 à zéro autorise la lecture des registres, tandis qu'à un il permet l'écriture dans les registres.

Généralités

Le MC6850 est un ACIA (Asynchrone Communication Interface Adapter) fabriqué par Motorola. Il réalise la mise en forme des données et contrôle la transmission série asynchrone entre le microprocesseur et un périphérique.

Registres de l'ACIA

L'ACIA est constitué de quatre registres accessibles par programme, en deux adresses seulement, à l'aide des signaux R/\bar{W} et RS.

Registres	RS	R/\bar{W}	Accès
registre de contrôle	0	0	écriture
registre d'état	0	1	lecture
registre d'émission	1	0	écriture
registre de réception	1	1	lecture

- Le *registre d'émission* reçoit, via le bus de données, les caractères qui seront mis en forme pour être transmis sur la liaison série.
- Le *registre de réception* contient la donnée en provenance du périphérique après que le nombre de bits programmé ait été reçu.
- Le *registre de contrôle* permet de fixer le type de transmission et la possibilité de travailler en mode d'interruption ou non.
- Les bits 0 et 1 déterminent le diviseur de la fréquence d'horloge d'émission et de réception. Ils génèrent également la fonction Master Reset étant donné que le composant n'a pas de broche Reset.

CR1	CR0	Fonction
0	0	- 1
0	1	- 16
1	0	- 64
1	1	Master Reset

- Les bits 2 à 4 sont les bits de sélection des mots. Ils fixent la parité, le nombre de bits par mot et la quantité de bits d'arrêt.

CR4	CR3	CR2	Fonction	
0	0	0	7 bits	- parité paire - 2 bits d'arrêt
0	0	1	7 bits	- parité impaire - 2 bits d'arrêt
0	1	0	7 bits	- parité paire - 1 bit d'arrêt
0	1	1	7 bits	- parité impaire - 1 bit d'arrêt
1	0	0	8 bits	- sans parité - 2 bits d'arrêt
1	0	1	8 bits	- sans parité - 1 bit d'arrêt
1	1	0	8 bits	- parité paire - 1 bit d'arrêt
1	1	1	8 bits	- parité impaire - 1 bit d'arrêt

- Les bits 5 et 6 contrôlent la sortie RTS et la possibilité d'envoyer une interruption lorsque le registre d'émission est vide. Ils permettent également l'émission d'une longue suite de zéros sur la ligne, ce qui correspond à l'envoi d'un signal BREAK.

CR6	CR5	Fonction
0	0	RTS = 0 : interruption sur la transmission interdite
0	1	RTS = 0 : interruption sur la transmission autorisée
1	0	RTS = 1 : interruption sur la transmission interdite
1	1	RTS = 0 : interruption sur la transmission interdite envoi d'un BREAK.

- Le bit 7, lorsqu'il est mis à 1, autorise les interruptions à la réception. Les causes d'interruption sont : le registre de réception est plein, le signal DCD est passé de 0 à 1, erreur d'OVERRUN ; c'est-à-dire qu'un caractère reçu n'a pas été lu à temps et a été écrasé par le suivant.
- Le registre d'état fournit des renseignements sur l'état d'avancement de la transmission.
 - Le bit 0 indique que le registre de réception contient un caractère qui peut être lu. Ce bit est remis à zéro lorsque la donnée est lue, lors d'un Master Reset ou lorsque le signal DCD passe à 1.
 - Le bit 1 se positionne à 1 lorsque le registre d'émission est vide et qu'il est prêt à recevoir une donnée à transmettre au périphérique.
 - Le bit 2 reflète l'état du signal DCD, et, si elle est autorisée, une interruption est envoyée au microprocesseur. Il est remis à zéro par la lecture du registre d'état et du registre de réception. Si après ces lectures le bit reste à 1, c'est que le signal DCD est toujours à 1.

- Le bit 3 fournit l'état du signal CTS. Ce bit n'est pas remis à zéro par un Master Reset.
- Le bit 4 indique une erreur de format, c'est-à-dire que le récepteur n'a pas détecté de bit d'arrêt ou qu'il a reçu un signal BREAK.
- Le bit 5 renseigne les erreurs d'OVERRUN. Il est remis à zéro par la lecture du registre de réception.
- Le bit 6 permet de savoir si la parité du caractère reçu est conforme à celle qui a été programmée ; dans le cas contraire, il est remis à zéro.
- Le bit 7 indique l'état de la broche IQR. Lorsque cette dernière est à 0, le bit 7 est à 1.

ACIAs dans l'ATARI ST

L'ATARI ST comporte deux ACIAs 6850 : un pour réaliser la communication avec le clavier, et l'autre pour le fonctionnement de l'interface MIDI (Musical Instrument Digital Interface). Ils ne disposent que de deux adresses rassemblées en un seul bloc :

\$FFFC00	registre de contrôle de l'ACIA du clavier.
\$FFFC02	registre d'émission de l'ACIA du clavier.
\$FFFC04	registre de contrôle de l'ACIA du MIDI.
\$FFFC06	registre d'émission de l'ACIA du MIDI.

Pour la transmission avec le clavier, les paramètres sont :

- 8 bits de données ;
- 1 bit d'arrêt ;
- pas de parité ;
- 7812,5 bauds (500 kHz/64).

Les paramètres de l'interface MIDI sont identiques sauf la vitesse de transmission qui est de 31250 bauds (500 kHz/16), valeur fixée par les spécifications standard du MIDI.

CIRCUIT DMA

Le circuit DMA est un circuit 40 broches spécialement développé pour la série ST.

Le canal d'accès direct à la mémoire dynamique (RAM) est divisé pour permettre l'accès à vitesse lente (250 à 500 Kbits par seconde) et à vitesse rapide (jusqu'à 12 Mbits par seconde).

Le circuit DMA possède un registre adresse de base pour l'adresse d'accès des opérations de lecture et d'écriture. Le circuit possède un seul registre de ce type et donc, une seule opération de DMA est possible à la fois.

L'opération de DMA est effectuée au travers d'une pile FIFO (FIRST IN FIRST OUT ou Premier entré premier sorti) de 32 octets. La pile FIFO est programmée par l'intermédiaire du registre de contrôle de mode et du registre compteur de secteur.

La réussite ou l'échec d'un accès DMA est rapporté au travers du registre d'état qui peut être effacé par la transition du mode écriture vers le mode lecture dans le registre de contrôle de mode ML.

L'accès au BUS est assuré par la technique du premier arrivé premier servi. L'accès demeure effectif jusqu'à la fin de l'opération ou jusqu'à la production d'une erreur.

CIRCUIT GLUE

Le circuit GLUE se présente sous la forme d'un gros cube muni de 68 pattes et contient toute la logique annexe classique d'un ordinateur. Ce circuit prend en charge la gestion de tous les tampons et presque tous les décodages d'adresses du système. Il sert aussi de diviseur de fréquence pour alimenter les horloges des ACIAs, et différents signaux nécessaires à la production de l'image vidéo par le SHIFTER.

Le circuit GLUE n'est pas accessible par les adresses du système. Son utilisation est donc réservée aux bidouilleurs de matériel qui désirent réaliser des interfaces de digitalisation ou d'incrustation vidéo.

CIRCUIT MMU

Le MMU est un circuit semblable au GLUE par sa forme et l'impossibilité de l'atteindre par des fonctions logicielles extérieure. Il offre donc peu d'intérêt pour le programmeur.

Signalons cependant que ses principales fonctions consistent à gérer le multiplexage d'adresses des mémoires dynamiques, à compléter la gestion du DMA et à aider le SHIFTER à produire les signaux vidéo indispensables à la production de l'image.

CIRCUIT SHIFTER

Le SHIFTER est le circuit principal responsable de la production de l'image du ST. Il s'occupe du décodage de 32 K de mémoire et de la division éventuelle de ces 32 K en 2 ou 4 plans pour permettre la sortie d'une image couleur.

Le SHIFTER possède un registre (16 bits), base adresse de départ de la mémoire centrale, qui permet de positionner la mémoire vidéo n'importe où dans l'espace adressable disponible par pas de 256 octets.

Il possède, en outre, un registre adresse courante qui lui permet d'atteindre successivement tous les points à afficher, un registre de mode qui détermine son mode de fonctionnement (monochrome ou couleur) et, enfin, un registre de mode de synchronisation qui permet l'adaptation des fréquences aux normes américaines ou européennes.

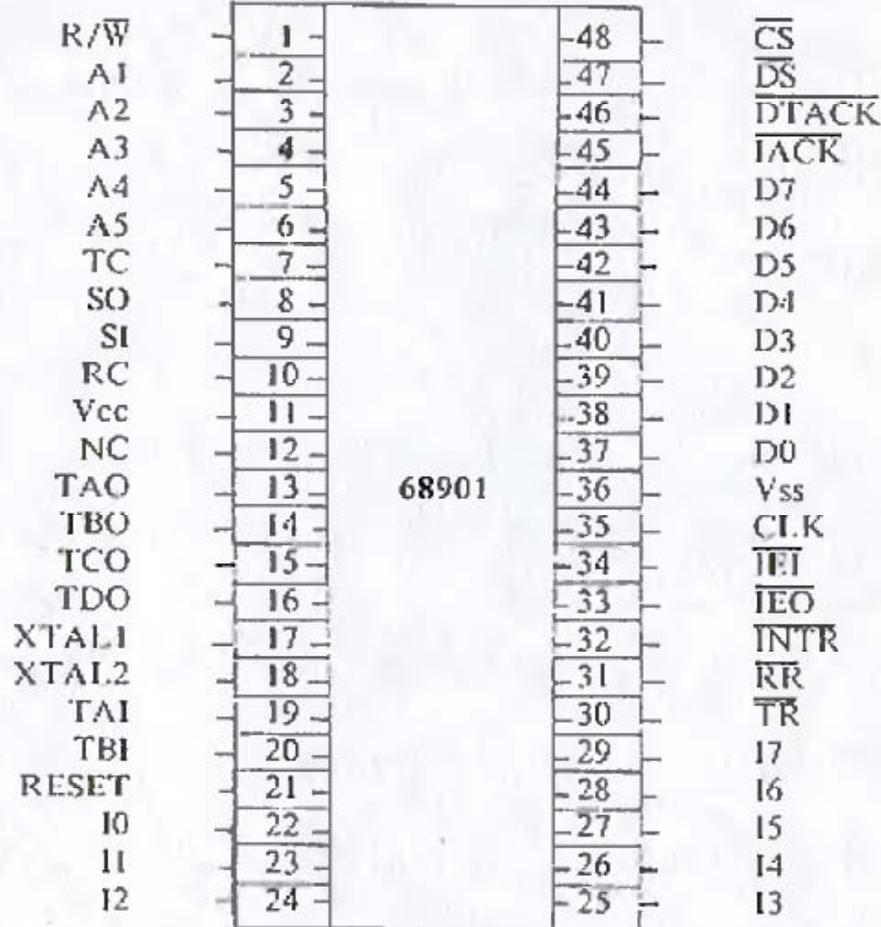
BROCHAGE DU 68000

D4	1	64	D5
D3	2	63	D6
D2	3	62	D7
D1	4	61	D8
D0	5	60	D9
AS	6	59	D10
UDS	7	58	D11
LDS	8	57	D12
R/W	9	56	D13
DTACK	10	55	D14
BG	11	54	D15
BGACK	12	53	Vss
BR	13	52	A23
VDD	14	51	A22
CLK	15	50	A21
VSS	16	49	VDD
HALT	17	48	A20
RESET	18	47	A19
VMA	19	46	A18
E	20	45	A17
VPA	21	44	A16
BERR	22	43	A15
IPL2	23	42	A14
IPL1	24	41	A13
IPL0	25	40	A12
FC2	26	39	A11
FC1	27	38	A10
FC0	28	37	A9
A1	29	36	A8
A2	30	35	A7
A3	31	34	A6
A4	32	33	A5

BROCHAGE DU 68000

<i>Nom de broche</i>	<i>Description</i>	<i>Type</i>
A1-A23	Bus d'adresses	Output
D0-D15	Bus de données	Bidirectionnel
\overline{AS}	Adresse valide	Output
\overline{UDS}	Sélection de l'octet de donnée de poids fort	Output
\overline{LDS}	Sélection de l'octet de donnée de poids faible	Output
R/ \overline{W}	Lecture / écriture	Output
\overline{DTACK}	Transfert de données réalisé	Input
\overline{BR}	Demande d'accès au bus	Input
\overline{BG}	Attribution du bus	Output
\overline{BGACK}	Détection de l'attribution du bus	Input
$\overline{IPL0}$ - $\overline{IPL2}$	Niveau de priorité des interrupts	Input
\overline{BERR}	Erreur bus	Input
\overline{RESET}	Initialisation du processeur	Bidirectionnel
\overline{HALT}	Signal d'arrêt	Bidirectionnel
E	Validation (horloge)	Output
\overline{VPA}	Validation adresse périphérique	Input
\overline{VMA}	Validation adresse mémoire	Output
FC0-FC2	Signaux d'état du processeur	Output
CLK	Horloge	Input

BROCHAGE DU MFP 68901

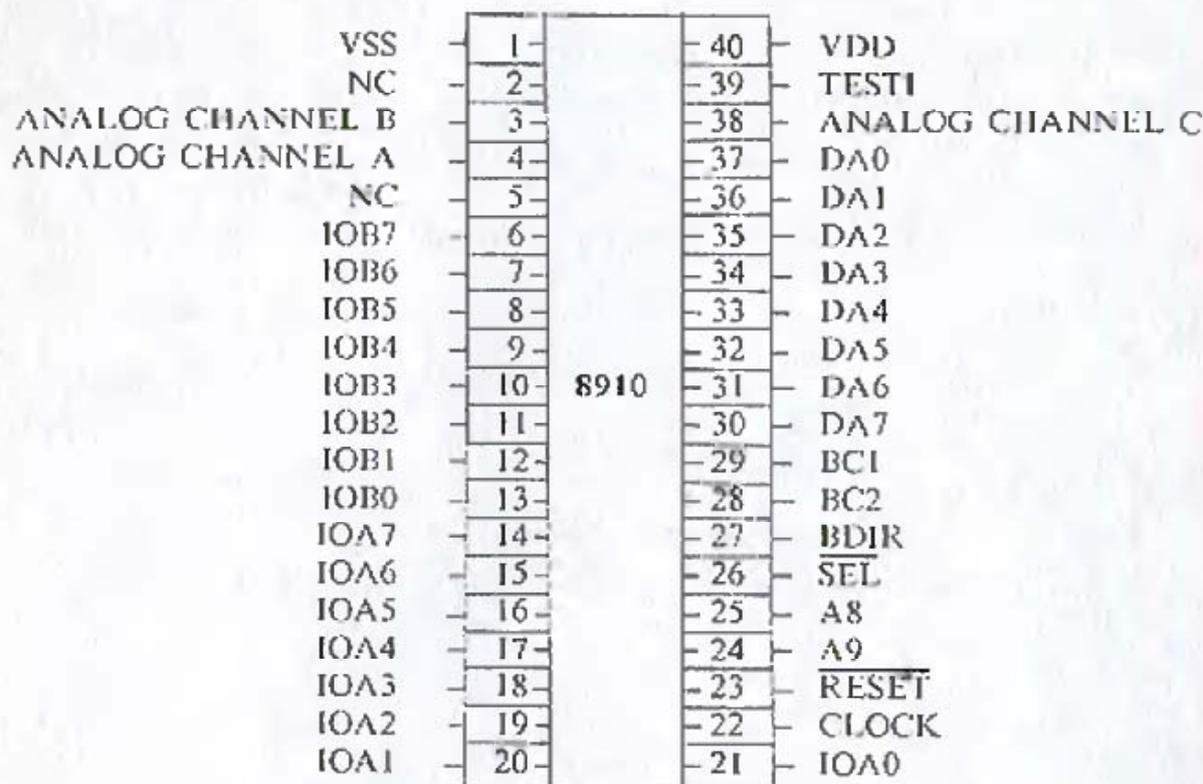


Nom de broche	Description	Type
Vcc	Alimentation	Input
Vss	Masse de l'alimentation (0 volt)	Input
CLK	Horloge	Input
D0 - D7	Bus de données	Bidirectionnel
CS	Sélection du circuit	Input
DS	Sélection de l'octet de donnée	Input
DTACK	Fin d'un cycle d'écriture ou de lecture	Output

BROCHAGE DU MFP 68901

<i>Nom de broche</i>	<i>Description</i>	<i>Type</i>
A1 - A5	Sélection des registres internes	Input
RESET	Remise à zéro	Input
$\overline{\text{INTR}}$	Demande d'interruption	Output
$\overline{\text{TACK}}$	Autorisation de l'interruption	Input
$\overline{\text{IEO}}$	Sortie de validation d'interrupt	Output
$\overline{\text{IEI}}$	Entrée de validation d'interrupt	Input
I0 - I7	Port d'entrée/sortie	Bidirectionnel
XTAL1, XTAL2	Entrées du quartz	Input
TAI, TBI	Entrée des horloges des compteurs A et B	Input
TAO, TBO, TCO, TDO	Sortie des compteurs	Output
SI	Entrée série	Input
SO	Sortie série	Output
RC	Horloge de réception	Input
TC	Horloge d'émission	Input
$\overline{\text{RR}}$	Etat du registre de réception des données	Output
$\overline{\text{TR}}$	Etat du registre d'émission des données	Output
R/ $\overline{\text{W}}$	Lecture / écriture	Input

BROCHAGE DU CIRCUIT AY3-8910

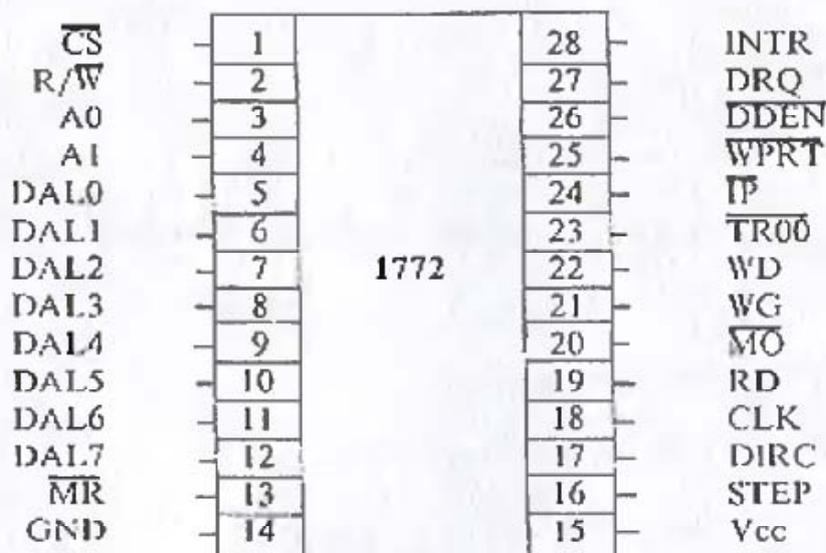


Nom de broche	Description	Type
VSS	Masse	
NC	Non connecté	
ANALOG CHANNEL	Canal de sortie	Output
IOB7----IOB0	Port de données 8 bits	Bidirect
IOA7----IOA0	Port de données 8 bits	Bidirect
CLOCK	Horloge de référence pour la fréquence, le bruit et l'enveloppe	Input
RESET	Entrée REZ.	Input
A9	Fixé à 0	Input
A8	Fixé à 1	Input

BROCHAGE DU CIRCUIT AY3-8910

<i>Nom de broche</i>	<i>Description</i>	<i>Type</i>
$\overline{\text{SEL}}$	Sélection de la fréquence d'horloge	Input
BDIR	Contrôle des opérations internes	Input
BC2, BC1		Input
DA7----DA0	Entrées et sorties de données	Bidirect
TEST1	Broche de test	
VDD	Alimentation +5 volts	

BROCHAGE DU FDC WD1772



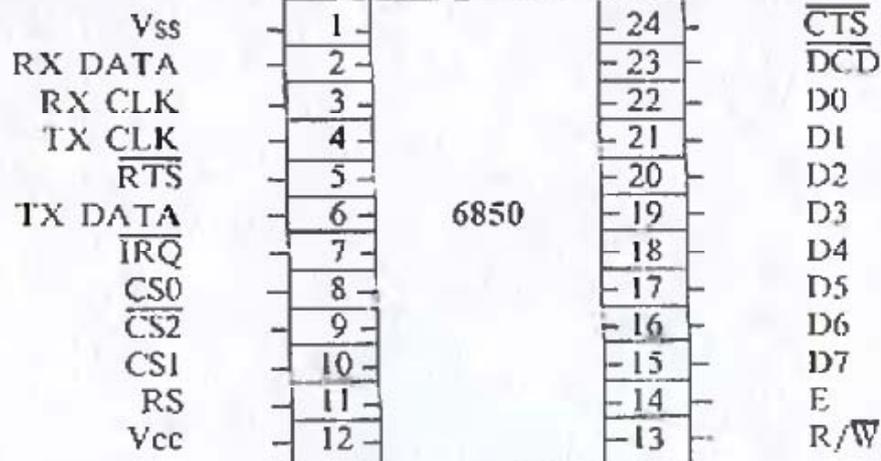
<i>Nom de broche</i>	<i>Description</i>	<i>Type</i>
Vcc	Alimentation (+ 5 volts)	Input
GND	Masse de l'alimentation (0 volt)	Input
\overline{MR}	Remise à zéro	Input
DAL0-DAL7	Bus de données	Bidirect
\overline{CS}	Sélection du circuit	Input
R/ \overline{W}	Lecture / écriture	Input
A0-A1	Sélection des registres internes	Input
DRQ	Etat du registre de données	Output
INTR	Demande d'interruption	Output
CLK	Horloge	Input
STEP	Pas du moteur de la tête	Output
DIRC	Direction du déplacement de la tête	Output

BROCHAGES ET CONNECTEURS

BROCHAGE DU FDC WD 1772

<i>Nom de broche</i>	<i>Description</i>	<i>Type</i>
\overline{RD}	Réception des données	Input
MO	Commande de mise en marche du moteur	Output
WG	Verrouillage de la logique d'écriture	Output
WD	Emission des données	Output
$\overline{TR00}$	Indication de la piste 0	Input
\overline{IP}	Détection du trou d'index	Input
\overline{WPRT}	Protection contre l'écriture	Input
\overline{DDEN}	Simple ou double densité	Input

BROCHAGE DE L'ACIA 6850



Nom de broche	Description	Type
Vss	Dvoltage masse électrique	Input
RX DATA	Réception des données	Input
RX CLK	Horloge de réception	Input
TX CLK	Horloge d'émission	Input
$\overline{\text{RTS}}$	Demande d'émission	Output
TX DATA	Emission des données	Output
$\overline{\text{IRQ}}$	Demande d'interruption	Output
CS0- $\overline{\text{CS2}}$	Sélection du circuit	Input
RS	Sélection des registres internes	Input
Vcc	Alimentation (+ 5 volts)	Input
R/ $\overline{\text{W}}$	Lecture / écriture	Input
E	Validation (horloge)	Input
D0-D7	Bus de données	Bidirectionnel
DCD	Détection de porteuse	Input
$\overline{\text{CTS}}$	Prêt à émettre	Input

BROCHAGE DU CIRCUIT DMA

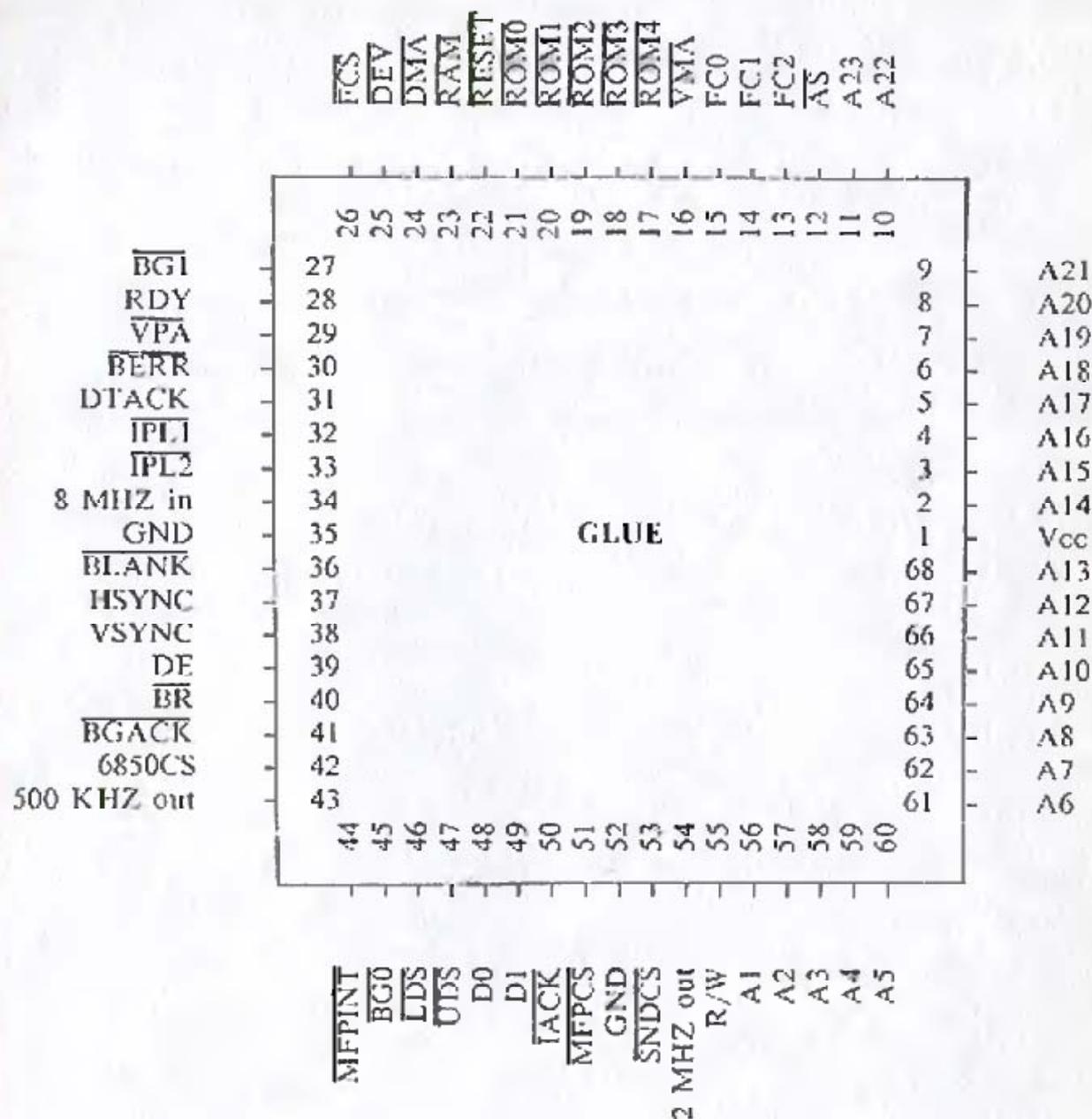
R/W	1	40	Vcc
A1	2	39	CLK
$\overline{\text{FCS}}$	3	38	$\overline{\text{RDY}}$
D0	4	37	$\overline{\text{ACK}}$
D1	5	36	CD0
D2	6	35	CD1
D3	7	34	CD2
D4	8	33	CD3
D5	9	32	CD4
D6	10	31	CD5
D7	11	30	CD6
D8	12	29	CD7
D9	13	28	GND
D10	14	27	CA2
D11	15	26	CA1
D12	16	25	$\overline{\text{CR/W}}$
D13	17	24	$\overline{\text{HDCS}}$
D14	18	23	HDRQ
D15	19	22	$\overline{\text{FDCS}}$
GND	20	21	FDRQ

<i>Nom de broche</i>	<i>Description</i>	<i>Type</i>
R/ $\overline{\text{W}}$	Lecture / écriture	Input
A1	Sélection d'un des 2 registres adresse de base	Input
$\overline{\text{FCS}}$	Sélection circuit	Input
D0-D15	Bus de données 16 bits	Bidirectionnel
GND	Masse	
Vcc	Alimentation (+ 5 volts)	Input
CLK	Horloge	Input
RDY	Prêt	Input
ACK	Autorisation (acknowledge)	Output
CD0-CD7	Bus de données 8 bits	Bidirectionnel

BROCHAGE DU CIRCUIT DMA

<i>Nom de broche</i>	<i>Description</i>	<i>Type</i>
CA1-CA2	Choix des registres de sélection	Output
CR/W	Sens de transfert des données	Input
$\overline{\text{HDCS}}$ - $\overline{\text{FDCS}}$	Sélection du circuit de transfert	Output
HDRQ-FDRQ	Réception des demandes de transfert	Output

BROCHAGE DU CIRCUIT GLUE

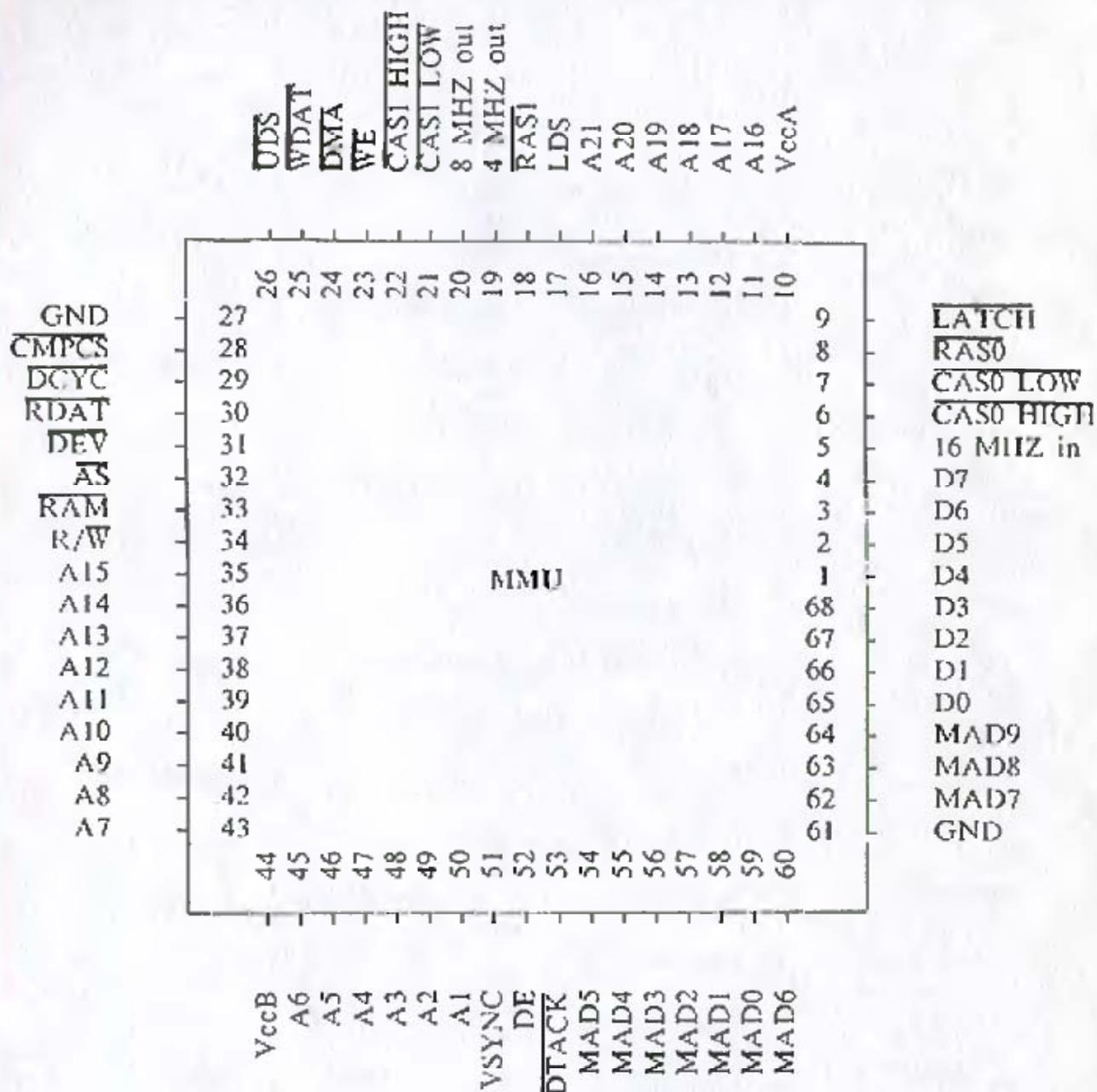


Nom broche	Description	Type
Vcc	Alimentation (+5 volts)	
A1-A23	Bus d'adresses	Input
<u>AS</u>	Adresse valide	Input
FC0 - FC2	Signaux d'état du processeur	Input
<u>VMA</u>	Validation adresse mémoire	Input
<u>ROM0</u> - <u>ROM4</u>	Sélection d'adresses pour l'activation des ROM	Output
<u>RESET</u>	Initialisation	Input
<u>DEV</u>	Raccordé au DEV du MMU	

BROCHAGE DU CIRCUIT GLUE

<i>Nom de broche</i>	<i>Description</i>	<i>Type</i>
$\overline{FCS} - \overline{IACK}$		
$\overline{MFPCS} - RDY$	Connexions au 68901	Output
BG1	Attribution d'un bus	Input
VPA	Validation adresse périphérique	Output
\overline{BERR}	Erreur bus vers le 68000	Output
\overline{DTACK}	Transfert de données réalisé	Output
$\overline{IPL1} - \overline{IPL2}$	Niveaux de priorité interruptions	Output
8MHZ in	Horloge	Input
GND	Masse	
\overline{BLANK}	Sortie pour la gestion vidéo	Output
HSYNC-VSYNC	Sorties pour la gestion vidéo	Output
DE		Output
BR	Demande d'accès au bus	Output
\overline{BGACK}	Détection de l'attribution du bus	Output
6850CS	Connexion à l'ACIA	
500KHZ out	Activation des ACIA	Output
\overline{MFPINT}	Entrée en provenance du 68901	Input
BG0	Non utilisé	
\overline{LDS}	Sélection de l'octet de donnée de poids faible	Input
\overline{UDS}	Sélection de l'octet de donnée de poids fort	Input
D0 - D1	Données	Bidirectionnel
\overline{SNDCS}	Activation du générateur sonore	Output
2MHZ out	Activation générat. sonore et ACIA	Output
R/ \overline{W}	Lecture / écriture	Input

BROCHAGE DU CIRCUIT MMU



Nom de broche	Description	Type
D0 - D7	Bus de données	Bidirectionnel
16MHz in	Horloge	Input
CAS - RAS	Signaux d'activation de la mémoire vive (RAM)	Output

BROCHAGE DU CIRCUIT MMU

<i>Nom de broche</i>	<i>Description</i>	<i>Type</i>
LATCH - RDAT - WDAT	LATCH, READ et WRITE DATA: lecture, écriture et verrouillage des données	Output
Vcc	Alimentation (+ 5 Volts)	
A1 - A21	Bus d'adresses	Input
GND	Masse	
CMPCS	Raccordé au CS du Shifter	Output
DCYC - DE		Output
AS	Adresse valide	Input
R/W	Lecture / écriture	Input
MAD0 - MAD9	Activation des adresses mémoire	Output
DTACK	Transfert de données réalisé	Output

BROCHAGE DU CIRCUIT SHIFTER

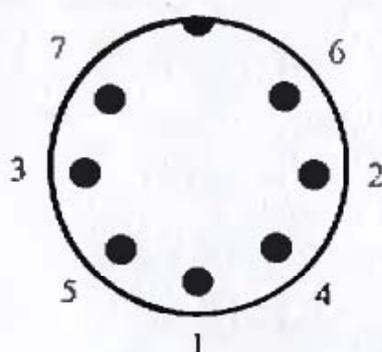
XLT 0	1	40	Vcc
XLT 1 32MHzin	2	39	16MHz out
D0	3	38	CS
D1	4	37	DE
D2	5	36	A1
D3	6	35	A2
D4	7	34	A3
D5	8	33	A4
D6	9	32	A5
D7	10	31	R/W
LOAD	11	30	MONO
D8	12	29	R0
D9	13	28	R1
D10	14	27	R2
D11	15	26	G0
D12	16	25	G1
D13	17	24	G2
D14	18	23	B0
D15	19	22	B1
GND	20	21	B2

<i>Nom broche</i>	<i>Description</i>	<i>Type</i>
XLT	Horloge	Input
D0-D15	Bus de données	Bidirectionnel
LOAD	Chargement depuis DCYC du MMU	Input
GND	Masse	
Vcc	Alimentation (+ 5 volts)	Input
CS	Sélection de circuit	Input
DE		Output
A1-A5	Bus d'adresses	Input

BROCHAGE DU CIRCUIT SHIFTER

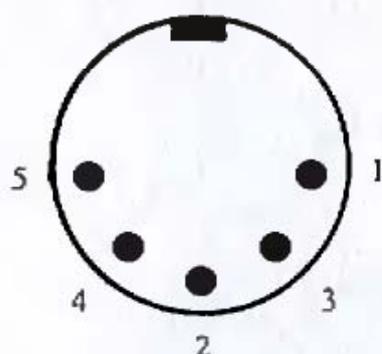
<i>Nom broche</i>	<i>Description</i>	<i>Type</i>
R/ \bar{W}	Lecture / écriture	Input
MONO	Signal vidéo monochrome	Output
R0-R2	Signal vidéo rouge	Output
G0-G2	Signal vidéo vert	Output
B0-B2	Signal vidéo bleu	Output
16MHZ out	Sortie HORLOGE divisée par 2	Output

CONNECTEURS - ALIMENTATION



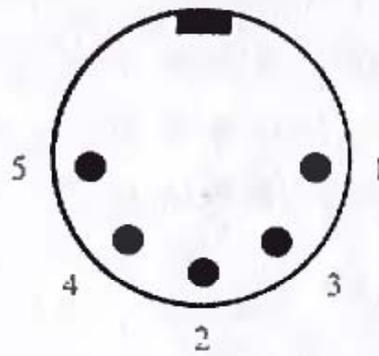
<i>N° broche</i>	<i>Fonction</i>
1	+ 5 volts
2	Open
3	Masse
4	+ 12 volts
5	- 12 volts
6	+ 5 volts
7	Masse

CONNECTEURS - SORTIE MIDI



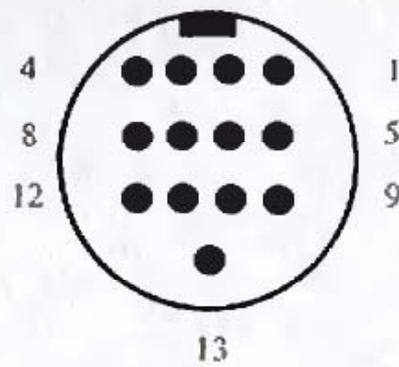
<i>N° broche</i>	<i>Fonction</i>
1	Transmission des données THRU
2	Masse
3	Masse THRU
4	Transmission des données OUT
5	Masse OUT

CONNECTEURS - ENTREE MIDI



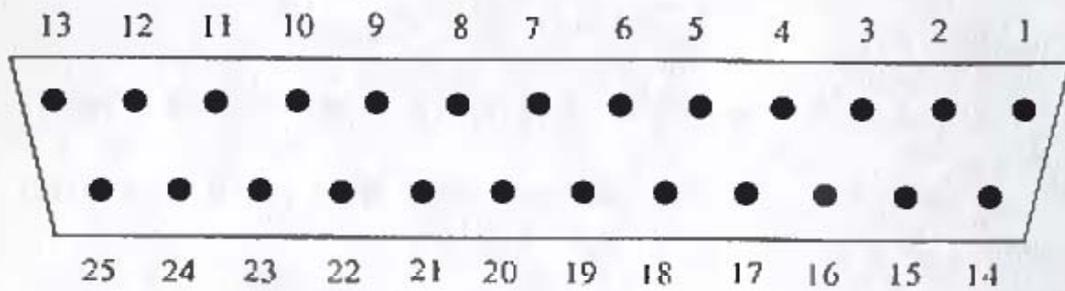
<i>N° broche</i>	<i>Fonction</i>
1	Transmission des données THRU
2	Masse
3	Masse THRU
4	Réception des données IN
5	Masse IN

CONNECTEURS - MONITEUR



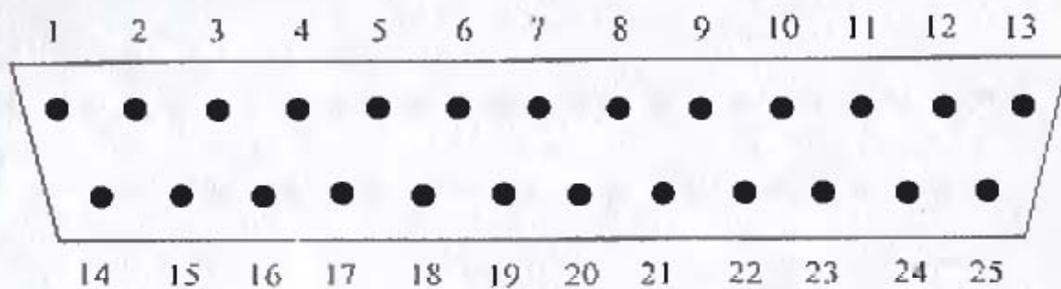
<i>N° broche</i>	<i>Fonction</i>
1	Sortie son
2	Réservée
3	Sortie à usage général
4	Détection de présence d'un moniteur vidéo monochrome
5	Entrée son
6	Vert
7	Rouge
8	Masse
9	Synchronisation horizontale
10	Bleu
11	Monochrome
12	Synchronisation verticale
13	Masse

CONNECTEURS - IMPRIMANTE



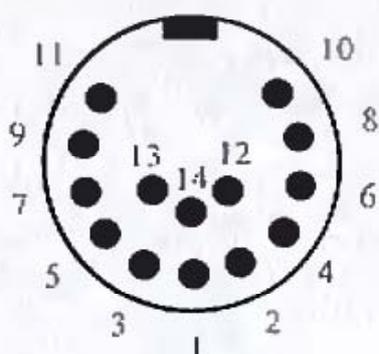
<i>N° broche</i>	<i>Fonction</i>
1	Centronics STROBE
2	Data 0
3	Data 1
4	Data 2
5	Data 3
6	Data 4
7	Data 5
8	Data 6
9	Data 7
10	Non connecté
11	Centronics BUSY
12 - 17	Non connectés
18 - 25	Masse

CONNECTEURS - SORTIE RS-232 (MODEM)



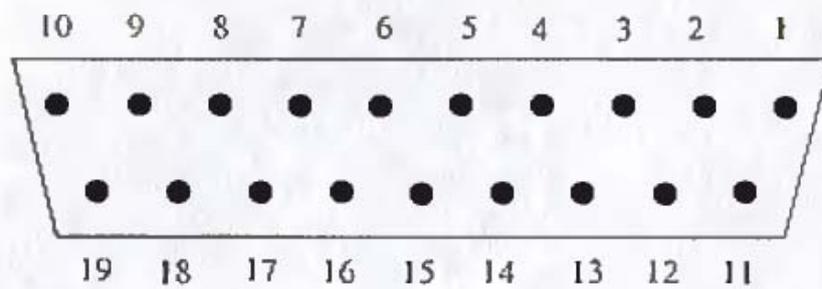
<i>N° broche</i>	<i>Fonction</i>
1	Masse châssis
2	Données transmises
3	Données reçues
4	RTS (Request To Send) permis d'envoi
5	CTS (Clear To Send) prêt à envoyer
6	Non connecté
7	Masse
8	DCR (Data Carrier Repeat)
9 - 19	Non connectés
20	DTR (Data Terminal Ready)
21	Non connecté
22	Ring indicator
23 - 25	Non connectés

CONNECTEUR POUR LECTEUR DE DISQUETTES



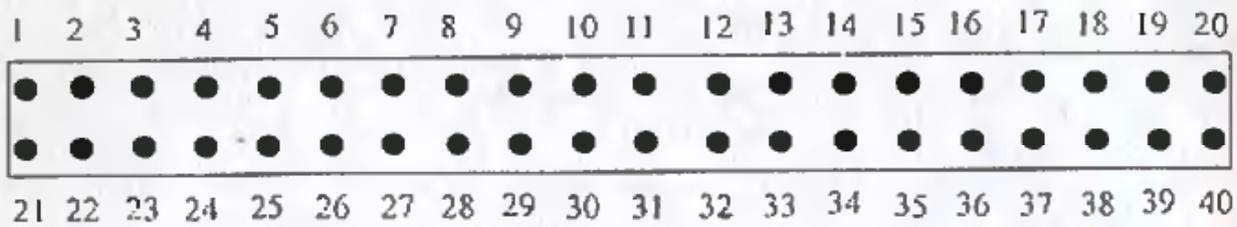
N° broche	Fonction
1	Lecture des données
2	Sélection de la face 0
3	Masse
4	Impulsion
5	Sélection du lecteur 0
6	Sélection du lecteur 1
7	Masse
8	Moteur ON
9	Direction IN
10	Step (pas)
11	Ecriture des données
12	Ecriture de la GATE
13	Piste 00
14	Protection en écriture

CONNECTEUR POUR DISQUE DUR



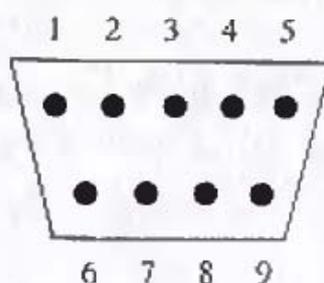
<i>N° broche</i>	<i>Fonction</i>
1	Data 0
2	Data 1
3	Data 2
4	Data 3
5	Data 4
6	Data 5
7	Data 6
8	Data 7
9	Select
10	Demande d'interruption
11	Masse
12	Reset
13	Masse
14	Accusé de réception
15	Masse
16	A1
17	Masse
18	Lecture / Ecriture
19	Demande de données

CONNECTEUR POUR CARTOUCHES



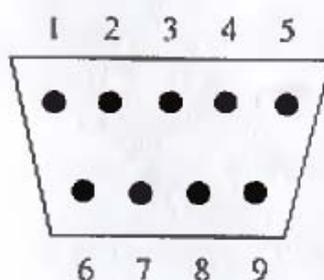
<i>N° broche</i>	<i>Fonction</i>
1	Alimentation + 5 Volts
2	Alimentation + 5 Volts
3	Data 14
4	Data 15
5	Data 12
6	Data 13
7	Data 10
8	Data 11
9	Data 8
10	Data 9
11	Data 6
12	Data 7
13	Data 4
14	Data 5
15	Data 2
16	Data 3
17	Data 0
18	Data 1
19	Adresse 13
20	Adresse 5
21	Adresse 8
22	Adresse 14
23	Adresse 7
24	Adresse 9
25	Adresse 6
26	Adresse 10
27	Adresse 5
28	Adresse 12
29	Adresse 11
30	Adresse 4
31	ROM sélect 3
32	Adresse 3
33	ROM sélect 4
34	Adresse 2
35	Validation de la donnée supérieure
36	Adresse 1
37	Validation de la donnée inférieure
38 - 40	Masse

CONNECTEUR POUR SOURIS OU MANETTE DE JEUX



<i>N° broche</i>	<i>Fonction</i>
1	Haut/XB
2	Bas/XA
3	Gauche/YA
4	Droite/YB
5	Non connecté
6	Feu / bouton de tir gauche
7	+ 5 volts
8	Masse
9	Feu du bouton de tir droit de la manette 1

CONNECTEUR POUR MANETTE DE JEUX



<i>N° broche</i>	<i>Fonction</i>
1	Haut
2	Bas
3	Gauche
4	Droite
5	Réservé
6	Bouton de tir
7	+ 5 volts
8	Masse
9	Non connecté

INSTRUCTION DEF FN ET FONCTION FN

Syntaxe

Déclaration : DEF FN NN(P1,P2,...,PN) = fonction BASIC

Utilisation : FN NN(p1,p2,...,pn)

Comme des milliers d'utilisateurs du Basic ST, vous n'utilisez jamais les fonctions définies par l'utilisateur. Utiliser de telles fonctions n'apparaît pas immédiatement nécessaire au programmeur débutant et les exemples des manuels ne montrent généralement pas leur utilité.

Pourtant, ces fonctions permettent des techniques de programmation particulièrement intéressantes.

Avantages

Les variables utilisées dans la fonction ne sont pas affectées par un appel.

Les fonctions peuvent être définies n'importe où dans le programme à condition que la logique du programme rencontre la définition, au moins une fois avant un appel.

On peut redéfinir une fonction autant de fois que nécessaire.

On peut définir une fonction qui utilise d'autres fonctions définies.

Inconvénient

Une fonction définie ne peut pas contenir d'instruction.

Exemple d'utilisation

- On veut réaliser une fonction qui donne la valeur hexadécimale d'une adresse mémorisée sous la forme classique (2 octets) à une autre adresse X.

INSTRUCTION DEF FN ET FONCTION FN

On peut écrire :

```
ADS = HEX$(PEEK (X) + 256 * PEEK (X+1))
```

On peut aussi écrire :

```
DEF FN AD$(X) = HEX$(PEEK (X) + 256 * PEEK (X+1))
```

De cette manière, chaque fois que l'on devra faire appel à la fonction, il suffira d'écrire : FN AD\$(Z) où Z sera soit la valeur de l'adresse qui contient l'adresse à rechercher, soit une variable qui contient cette valeur.

- On veut réaliser une fonction qui centre une chaîne de caractères dans un espace de N caractères.

Il suffit d'écrire :

```
DEF FN CT$(A$,N) = STRING$(  
N/2 - LEN (A$) / 2 - .5," ") + A$
```

L'appel s'effectue grâce à la fonction : FN CT\$(Z\$,I) où Z\$ est soit la variable, soit le texte à centrer, et I le nombre de caractères du champ de centrage.

- Calcul du jour courant de l'année :

```
DEF FN JC(J,M,A) =(M-1)*28  
VAL(MID$("000303060811131619212426",  
(M-1)*2+1,2)) - ((M>2) AND ((A AND NOT -4) = 0)) + J
```

Remarque : respectez les blancs entre A AND NOT -4.

Utilisation : pour déterminer le numéro du jour courant dans l'année correspondant au 15 octobre 1985, écrire :

```
PRINT FN JC(15,10,1985)  réponse : 288
```

Le 15/10/1985 est donc le 288^e jour de l'année.

- Calcul de la date interne :

```
DEF FN DT(J,M,A) = A * 365 + INT ((A-1)/4) + (M-1) * 28  
+ VAL (MID$ ("000303060811131619212426", (M-1) * 2 + 1, 2))  
- ((M > 2) AND ((A AND NOT -4)=0)) + J
```

Utilisation : identique à l'exemple 3, mais le nombre fourni représente le nombre de jours écoulés depuis un référentiel. Cette fonction est valable pour toutes les dates comprises entre 1901 et 2099.

INSTRUCTION DEF FN ET FONCTION FN

- Calcul du jour de la semaine :

Pour cette fonction, on utilise d'abord la précédente pour calculer la date interne, ensuite, la date interne est transformée en un jour de la semaine (LUNDI à DIMANCHE).

```
DEF FN JS(N) = MID$ ("VENDREDISAMEDI..DIMANCHELUNDI..  
MARDI..MERCREDIJEUDI..", (N-INT(N/7)*7)*8+1,8)
```

Remarque : en tapant cette phrase, remplacez les points par des blancs.

Utilisation : quel jour tombe le 1^{er} janvier 1986 ?

- calcul de la date interne :

```
PRINT FN DT(1,1,1986) fournira 725387
```

- calcul du jour :

```
PRINT FN JS(725387) fournira MERCREDI
```

EMPECHER LE TRAÇAGE DES FENETRES PAR LE BASIC

Lors de chaque appel d'une routine de la librairie des événements, le Basic retrace les fenêtres.

Il est possible d'empêcher certains traçages pour accélérer le fonctionnement du système. Pour cela, il suffit de modifier l'octet relatif 24 de la table SYSTAB. Cet octet est appelé GEMFLAG. L'écriture de la valeur 1 à cette adresse suspend l'appel des événements et, par conséquent, le traçage des fenêtres.

Pour remettre le système dans son état initial, il suffit de repositionner l'octet à 0.

`POKE SYSTAB+24,1` et `POKE SYSTAB+24,0`

Ce sémaphore désactive l'interactivité entre GEM et Basic. Les menus ne sont alors plus accessibles. Seul Ctrl-C permet de sortir du programme. Veillez à bien réactiver ce sémaphore à l'issue du programme.

SIMULATION DES FONCTIONS HEX\$, BIN\$ ET &B

Le Basic ATARI souffre de nombreuses lacunes. Parmi celles-ci, nous avons noté, tout spécialement, l'absence de la fonction classique de conversion en binaire BIN\$ et de sa soeur jumelle &B qui réalise la conversion inverse. La fonction HEX\$ quand à elle existe bel et bien, cependant, elle ne fonctionne pas correctement et retourne systématiquement le même résultat (50) quelle que soit la valeur de la variable considérée.

Pour remédier à cet état de choses, nous vous proposons trois routines qui vous permettront de simuler ces trois fonctions.

Routine HEXA.BAS

```
10      REM CONVERSION EN HEXADECIMAL
20      CLEARW 2
30      INPUT A
40      GOSUB 1000
50      PRINT"HEX$(;"A;)"=";HE$
60      END
1000    REM SOUS-ROUTINE HEXA
1010    HES=""
1020    BS="0123456789ABCDEF"
1030    X=0
1040    B=A
1050    C=B/16
1060    IF C<1 THEN X=1
1070    C=(C-INT(B/16))*16
1080    HES=MID$(BS,C+1,1)+HE$
1090    IF X=1 THEN RETURN
1100    B=INT(B/16)
1110    GOTO 1050
```

Routine BIN.BAS

```
10      REM CONVERSION EN BINAIRE
20      CLEARW 2
30      INPUT A
40      GOSUB 1000
50      PRINT"BIN$(;"A;)"=";BI$
60      END
1000    REM SOUS-ROUTINE BINAIRE
1010    BI$=""
1020    B$="01"
1030    X=0
1040    B=A
1050    C=B/2
1060    IF C<1 THEN X=1
1070    C=(C-INT(B/2))*2
1080    BI$=MID$(B$,C+1,1)+BI$
```

SIMULATION DES FONCTIONS HEX\$,BIN\$ ET &B

```
1090 IF X=1 THEN RETURN
1100 B=INT(B/2)
1110 GOTO 1050
```

Routine DECIM.BAS

```
10 REM CONVERSION BINAIRE DECIMAL
20 CLEARW 2
30 INPUT A$
40 GOSUB 1000
50 PRINT"VALEUR DECIMALE DE ";A$;"=";DE
60 END
1000 REM ROUTINE
1005 DE=0
1010 FOR I=1 TO LEN(A$)
1020 J=LEN (A$)+1-I
1030 IF MID$(A$,J,1)="0" THEN 1050
1040 DE=DE+2^(I-1)
1050 NEXT I
1060 RETURN
```

MANIPULATIONS DE L'ECRAN EN BASIC

La structure d'implantation de l'écran en mémoire centrale permet de réaliser, même en Basic, quelques effets intéressants ou simplement originaux.

La mémoire écran en Basic se trouve à l'adresse 78000H pour la version 520 ST standard et 260 ST. Pour les systèmes 1040 ST et 520 ST+, la mémoire écran se trouve à l'adresse F8000H.

La mémoire écran occupe 32K et va donc de l'adresse 78000 à 7BFFF ou F8000 à FBFFF.

Sauvegarde de l'image écran sur disque

Pour sauvegarder l'image écran sur disque, il suffit d'utiliser la commande BSAVE sous la forme :

```
BSAVE "ECRAN.BIN",&H78000,32768 (520 ST et 260 ST)
```

ou

```
BSAVE "ECRAN.BIN",&HF8000,32768 (520 ST+ et 1040 ST)
```

Inversion de l'écran

Ce petit programme qui inverse logiquement chaque octet de l'écran permet de présenter l'écran en blanc sur fond noir. Bien sûr, la vitesse d'exécution du Basic le rend un peu lent mais, nous sommes sûrs qu'avec un peu d'entraînement, il est très facile de réaliser rapidement un programme semblable en langage machine.

```
10 FOR I=&H78000 TO &H7BFFF STEP 2  
20 POKE I,NOT(PEEK(I))  
30 NEXT I
```

Retournement d'écran

Ce programme très simple n'a pas d'autre utilité que de retourner l'écran pour réaliser un petit effet idiot mais amusant.

```
10 FOR I=0 TO 30000 STEP 2  
20 POKE &H78000+30000-I,PEEK(&H78000+I)  
30 NEXT I
```

Lecture de l'écran dans une variable tableau

Le programme suivant permet de sauver l'écran dans un tableau entier de façon à pouvoir le relire lorsque ce sera utile.

MANIPULATIONS DE L'ECRAN EN BASIC

Une limite, dans la définition des tableaux entiers, empêche de définir les 16 384 variables nécessaires pour contenir les 32 768 octets de l'écran. Cependant, les variables disponibles suffisent amplement, car l'écran n'occupe pas tout à fait 32 768, mais bien 30 000 octets (600 x 400, soit 240 000 bits ou 30 000 octets).

```
10 DEFINT A
15 REM SAUVEGARDE
20 DIM A(16370)
30 FOR I=0 TO 16370
40 A(I)=PEEK(&H78000+I*2)
50 NEXT I
60 REM EFFACEMENT
70 FOR I=0 TO 16370
80 POKE &H78000+I*2,0
90 NEXT I
100 REM RAPPEL
110 FOR I=0 TO 16370
120 POKE &H78000+I*2,A(I)
130 NEXT I
```

Hardcopy automatique

Ce truc concerne à la fois l'écran et l'imprimante.

Pour produire une copie physique de l'écran (en mode graphique), il suffit, si l'on dispose d'une imprimante ATARI ou IBM compatible, de tenir conjointement les touches ALT et HELP. Cependant, cette commande nécessite l'utilisation du mode interactif.

Il est possible de réaliser cette commande par un appel du BIOS (*voir truc sur l'appel du BIOS*) ou, plus simplement encore, en modifiant l'état de la variable qui sert de sémaphore de copie située à l'adresse 1062.

Ainsi : POKE 1062,0 déclenche la copie de l'écran.

Les trucs et astuces en Logo pourraient remplir le présent volume.

Le Logo n'est pas le langage le plus utilisé des possesseurs de ST, ce qui est peut-être regrettable, car il est performant et nettement plus achevé que le Basic.

Traduction des primitives en français

La procédure suivante vous permettra de traduire tous les mots clés (primitives) du Logo dans la langue de votre choix. Elle utilise la gestion des propriétés et plus particulièrement la propriété .PRM.

```
TO TRADUIT :ANGLAIS :FRANCAIS
  PPROP : :FRANCAIS ".PRM GPROP :ANGLAIS ".PRM
  END
```

L'utilisation est simple, il suffit d'écrire par exemple :

```
TRADUIT FD AVANCE
```

La primitive FD possède maintenant un équivalent français qui peut être utilisé directement dans un commande du type :

```
AVANCE 100
```

Production de bruits divers

La primitive SOUND du Logo est difficile à utiliser. Nous vous proposons deux exemples qui vous permettront de produire, respectivement, le bruit d'un coup de feu et le bruit du sac et du ressac des vagues sur les rochers.

Le second exemple sera utilisé dans l'exemple d'appel du BIOS étendu en Basic et vous pourrez remarquer que la vitesse d'exécution de l'assembleur permettra de transformer le doux bruit des vagues en un vacarme infernal du genre de celui produit par les pales d'un hélicoptère.

```
TO BANG
  SOUND [0 0 1 0 2 0 3 0 4 0 5 0 6 15 7 199 8 16 9 16 10 16 11 0 12
        17 13 9] END
```

```
TO VAGUE
  SOUND [0 0 1 0 2 0 3 0 4 0 5 0 6 15 7 199 8 16 9 16 10 16 11 0 12
        31 13 14]
  END
```

APPEL DU GEM EN BASIC

Pour illustrer l'appel d'une fonction du GEM VDI depuis Basic, nous avons décidé de vous présenter un exemple supplémentaire d'appel de fonction.

Pour tirer parti de la fonction VDISYS, il est nécessaire de bien connaître l'interface d'appel du VDI (ou de l'AES).

Cette routine qui fournit la position de la souris sur l'écran, lorsque l'on appuie sur le bouton gauche, permet de réaliser des menus de sélection en Basic.

Les explications complémentaires sur la fonction 124 utilisée ici vous seront fournies dans le tome 2 du présent ouvrage.

```
10 POKE CONTRL,124
20 POKE CONTRL+2,0
30 POKE CONTRL+6,0
40 VDISYS
50 IF PEEK(INTOUT)=0 THEN GOTO 40
60 ?PEEK (PTSOUT),PEEK(PTSOUT+2)
70 GOTO 40
```

APPELS DU BIOS ET DU BDOS

Pour illustrer l'appel des fonctions du BDOS et du BIOS du chapitre 5, voici trois exemples d'appel de fonction depuis le Basic. Le premier utilise la fonction BDOS 8 qui lit un caractère en provenance du clavier sans produire d'écho à l'écran (C_NECIN). Le deuxième utilise la fonction 14 du BIOS étendu et permet d'enclencher le processus de HARDCOPY. Le troisième est plus complet et utilise la fonction 1C du BIOS étendu qui permet d'adresser directement les registres du PSG.

Exemple 1 - Lecture d'un caractère : programme CALLBDOS.BAS

L'appel n'utilise pas de paramètre d'entrée et fournit le code ASCII du caractère lu dans D0.

Le programme assembleur utilisé s'écrit :

2A 48	MOVEA.L A0,A5	Sauvegarde A0 dans A5 A0 contient l'adresse courante de la routine.
3F 3C 00 08	MOVE.W #8,-(A7)	Pousse le mot de 16 bits 8 (numéro de la fonction) dans la pile et décrémente la pile.
4E 41	TRAP #1	TRAP d'appel du BDOS
54 8F	ADDQ.L #2,A7	Ajoute 2 à la pile pour la rééquilibrer.
3B 40 00 10	MOVE.W D0,\$10(A5)	Sauve D0 à l'adresse 10H relative à A5 qui depuis la première instruction pointe sur le début du programme.
4E 75	RTS	Retour au BASIC
00 00		Valeur par défaut de l'adresse 10.

Les octets sont groupés par deux pour constituer des mots de 16 bits qui seront chargés dans un tableau d'entiers.

On obtient donc :

```
2A48 3F3C 0008 4E41 548F 3B40 0010 4E75 0000
```

Ce qui fait 9 mots. Il faut donc dimensionner le tableau avec 8 éléments (il existe un élément 0).

L'adresse d'implantation de la routine est donnée en ligne 80 par le VARPTR de l'élément 0 du tableau.

Programme \CALLBDOS.BAS

```
10 DEFINT M
20 DIM M(8)
```

APPELS DU BIOS ET DU BDOS

```
30 FOR I=0 TO 8
40 READ XS
50 Z=VAL("&H"+X$)
60 M(I)=Z
70 ADR=0
80 ADR=VARPTR(M(0))
90 CALL ADR
100 ?M(8)
110 GOTO 90
150 DATA 2A48,3F3C,0008,4E41
160 DATA 548F,3B40,0010,4E75,0000
```

Exemple 2 - Hardcopy

Cet exemple illustre un appel du BIOS (TRAP #14).

Le programme est identique à celui de l'exemple 1, à l'exception du numéro de la routine et du numéro du TRAP.

```
2A 48      MOVEA.L A0,A5
3F 3C 00 14  MOVE.W  #$14,-(A7)
4E 4E      TRAP   #14
54 8F      ADDQ.L  #2,A7
3B 40 00 10  MOVE.W  D0,$10(A5)
4E 75      RTS
00 00
```

Programme *HARDCOPY.BAS*

```
10 DEFINT M
20 DIM M(8)
30 FOR I=0 TO 8
40 READ XS
50 Z=VAL("&H"+X$)
60 M(I)=Z
65 NEXT I
70 ADR=0
80 ADR=VARPTR(M(0))
90 CALL ADR
100 ?M(8)
150 DATA 2A48,3F3C,0014,4E4E
160 DATA 548F,3B40,0010,4E75,0000
```

Exemple 3 - Appel du PSG

L'exemple 3 est plus complexe : la routine utilisée du BIOS demande deux arguments de 16 bits. Le premier doit contenir le numéro du registre et le second la donnée à y écrire.

APPELS DU BIOS ET DU BDOS

Le programme assembleur s'écrit :

```
2A 48      MOVEA.L A0,A5
3F 3C 00 00 MOVE.W  #0,-(A7)    0 pour le registre au départ
3F 3C 00 00 MOVE.W  #0,-(A7)    0 pour la donnée au départ
3F 3C 00 1C MOVE.W  #$1C,-(A7) Fonction 1C du BIOS
4E 4E      TRAP    #14
5C 8F      ADDQ.L  #6,A7      Réajuste de 6 octets - 3 mots
3B 40 00 18 MOVE.W  D0,$18(A5) sauvegarde de D0
4E 75      RTS          Retour au BASIC
00 00
```

Le programme charge la valeur du registre dans M(2) qui correspond aux octets 4 et 5 du programme (premier couple de 00). M(4) reçoit la donnée à écrire (deuxième couple de 00).

Remarque : le numéro du registre doit être augmenté de 80H (128) pour forcer l'écriture du registre.

Le programme contient quelques exemples de données qui permettent de simuler le bruit d'un hélicoptère.

Grâce à cette routine, vous pouvez adresser directement le PSG et tous les effets sonores sont dès lors possibles (sirène, explosion, laser galactique...).

Programme \GIA.BAS

```
1 REM HELICO
10 DEFINT M
20 DIM M(12)
30 FOR I=0 TO 12
40 READ X$
50 Z=VAL("&H"+X$)
60 M(1)=Z
70 NEXT I
80 FOR I=0 TO 13
90 M(2)=I+128
100 READ D
110 M(4)=D
120 ADR=0
130 ADR=VARPTR(M(0))
140 CALL ADR
150 NEXT I
160 DATA 2A48,3F3C,0000,3F3C,0000,3F3C,001C,4E4E
170 DATA 5C8F,3B40,0018,4E75,0000
200 DATA 0,0,0,0,0,0,15,199,16,16,16,0,6,12
```

ANNEXE

TABLE DES CODES CLAVIER RETOURNEE
PAR LE GESTIONNAIRE CLAVIER

Hexa	touche	Hexa	touche	Hexa	touche
01	Esc	24	J	47	HOME
02	I	25	K	48	FLECHE HAUT
03	2	26	L	49	NON UTILISE
04	3	27	M	4A	CLAV NUM -
05	4	28	ù	4B	FLECHE GAUCHE
06	5	29	π@	4C	NON UTILISE
07	6	2A	SHIFT gauche	4D	FLECHE DROITE
08	7	2B	<>	4E	CLAV NUM +
09	8	2C	W	4F	NON UTILISE
0A	9	2D	X	50	FLECHE BAS
0B	0	2E	C	52	INSERT
0C	.	2F	V	51	NON UTILISE
0D	-	30	B	53	DELETE
0E	BS	31	N	54	NON UTILISE
0F	TAB	32	?,	5F	NON UTILISE
10	A	33	;	60	TOUCHE ISO
11	Z	34	/:	61	UNDO
12	E	35	←=	62	HELP
13	R	36	SHIFT droit	63	CLAV NUM (
14	T	37	NON UTILISE	64	CLAV NUM)
15	Y	38	ALT	65	CLAV NUM /
16	U	39	ESPACE	66	CLAV NUM *
17	I	3A	CAPS LOCK	67	CLAV NUM 7
18	O	3B	F1	68	CLAV NUM 8
19	P	3C	F2	69	CLAV NUM 9
1A	[3D	F3	6A	CLAV NUM 4
1B]S*	3E	F4	6B	CLAV NUM 5
1C	RET	3F	F5	6C	CLAV NUM 6
1D	CNTL	40	F6	6D	CLAV NUM 1
1E	Q	41	F7	6E	CLAV NUM 2
1F	S	42	F8	6F	CLAV NUM 3
20	D	43	F9	70	CLAV NUM 0
21	F	44	F10	71	CLAV NUM .
22	G	45	NON UTILISE	72	CLAV NUM ENTER
23	H	46	NON UTILISE		

INDEX

68000	13, 25, 127, 275	BF	95
ABCD	135	BIOS	179, 220, 238
ABORT	178	BIOSKEYS	233
ABS	63, 94	BK	95
ACC	170	BL	95
ACIA	24, 28, 265, 279	BLOAD	42, 74, 81
ADD	132, 135	BOOT	175, 180, 189, 231, 238
ADDA	135	bootdev	239
ADDI	132, 135	BOX	95
ADDQ	133, 136	BPB	181, 182, 224
ADDRIN	89	BPS	182
ADDROUT	89	BRA	138
ADDX	136	BRA.S	182
ADRB	35	BREAK	42, 72, 171, 249
ADRH	35	BSAVE	42
AER	243	BSET	133, 138
AES	88	BSR	139
ALT	225	BSS	184, 186
AND	40, 94, 133, 136	BIST	139
ANDI	132, 136	buf11 - buf12	240
.APV	93	.BUR	93
ARC	94	BURY	95
ARCHIVE	194	BUTFIRST	95
ARCTAN	94	BUTLAST	95
ASC	63	BYE	95
ASCII	94	C	170
ASL	137	CALL	42, 79
ASM	170	CAPS LOCK	225, 231
ASR	137	CAT	171
ATN	63	CATCH	95, 119
AUTO	42	cartouches	27, 295
AUX	222	CCP	179, 184
AY3-8910	13, 251, 275	CD	171
BACK	95	CDBL	63
BACKUP	194	centronics	14, 250, 258
BAS	170	CHAIN	42
BAT	170	CHANGEF	96
BATCH	177	CHAR	96
BCC	137	CHECKSUM	183, 221
BCHG	138	CHK	139
BCLR	138	CHMOD	172, 178
BCONIN	223	CINT	63
BCONOUT	223	CIRCLE	43, 96
BCONSTAT	222	CLEAN	96
BCOSTAT	224	CLEAR	43
BDOS	179, 198	CLEARSCREEN	96
		CLEARTEXT	96
		CLEARW	43
		CLOSE	43

INDEX

CLOSEW	43	C_PRNOS	204
CLR	139	C_PRNOUT	201
CLS	172	C_RAWCIN	202
CLUSTER	176, 189, 193	C_RAWIO	202
cmdload	240		
CMP	139	DATA	44, 79
CMPA	140	D_Bcc	140
CMPI	140	DCD	266
CMPM	140	DDR	244
CO	96	DEF	170
COLOR	43	.DEF	93
colorptr	239	DEFDBL	39, 45
COMMAND.TOS	177, 179	DEF FN	45, 72
COMMAND.PRG	180	DEFINE	96
COMMON	44	DEFINEDP	97
CON(SOLE)	222, 223	DEFINT	39, 45
constantes	38	DEF SEG	45, 84
CONT	44, 72	defshifmd	239
conterm	240	DEFSNG	39, 45
.CONTENTS	93	DEFSTR	39, 45
CONTRL	63, 87	DEFUSR	45, 78
CONTROL	89	DEL	172
CONTROT	225	DELETE	46
COPY	172	DEGREES	97
COPYDEF	96	.DEPOSIT	93
COPYOFF	96	DIM	46, 71
COPYON	96	DIR	46, 97, 173
COS	63, 96	DIRECTORY	194
COUNT	96	DIVS	140
CRC	260	DIVU	141
CS	96	DMA	238, 264, 268,
CSNG	63		280
CT	96	DOC	170
CTRL - *	117	DOSOUND	235
CTS	231, 267	drvbits	240
CURSCONF	232	DRVMAP	225
CVD	63	dskbuf	241
CVI	63	DTA	195, 197
CVS	63	D_CREATE	209
C_AUXIS	204	D_DELETE	209
C_AUXIN	201	D_FREE	208
C_AUXOS	205	D_GETDRV	205
C_AUXOUT	201	D_GETPATH	215
C_CONIN	200	D_SETDRV	203
C_CONIS	203	D_SETPATH	209
C_CONOS	204	ECHO	173
C_CONOUT	201	ED	97
C_CONRS	203	EDALL	97
C_CONWS	202	EDF	97
C_NECIN	202		

EDIT	46, 97	FLOAT	64
EDNS	97	flock	238
EDPS	97	FLOPFMT	229
ELLIPSE	46, 97	FLOPRD	228
EMPTYP	97	FLOPVER	232
END	47, 98	FLOPWR	228
.ENL	93	.FMT	93
ENV	173	FNAME	181
EOF	64	FOLLOW	48, 99
EOR	133, 141	FOR	48, 75
EORI	132, 141	FORMAT	190
EQUALP	98	FORWARD	99
EQV	40	.FPT	93
ER	98	FPUT	99
ERA	47, 173	frclock	240
ERALL	98	FRE	64
ERASE	47, 98	FUIW	49
ERASEFILE	98	fverify	239
FRL	64	F_ATTRIB	212
ERN	98	F_CLOSE	211
ERNS	98	F_CREATE	210
ERPS	98	F_DATIME	218
ERR	64, 178	F_DELETE	212
ERRACT	98	F_DUP	215
ERROR	47, 98	F_FORCE	215
ESC V	176	F_GETDTA	207
ESC W	175	F_IOCTL	213
etv	238	F_OPEN	210
.EXAMINE	93	F_READ	211
EXECFLG	181	F_RENAME	218
EXG	141	F_SEEK	212, 215
EXIT	173	F_SETDTA	205
EXP	64, 98	F_SFIRST	217
EXT	141	F_SNEXT	218
		F_WRITE	211
FALSE	99	GB	64, 89
FAT	174, 189	GEM	87, 185
FATBUF	181	GEMDOS	167, 179, 184, 186, 189
FCB	188, 195	GEMSYS	49, 89
FD	99	GET	49, 73, 173
FDC	15, 250, 277	GETBPB	224
FENCE	99, 120	GETMPB	222
FIELD	47, 73	GETREZ	227
FIL	170	GETTEXT	99
FILL	47, 99	GETTIME	233
FILLATTR	99	GFILL	99
FILLER	182	GIACCESS	234
FIRST	99		
FIX	64		

INDEX

GLIST	100	ISR	245
GLOBAL	89	ISRA - ISRB	244
GLUE	269, 282	ITEM	101
GO	100		
GOSUB	49, 71	JDISINT	233
GOTO	49	JENABINT	234
GOTOX	50	JMP	142
GPIP	243	JSR	142
GPROP	100		
GRAPHICS	100		
		KBDVBASE	236
HANDLER	188, 210, 219	KBRATE	236
hdv_....	240	KBSHIFT	225
HDC	250	KEYP	101
HEADING	100	KEYTBL	231
HELP	174	KILL	51, 73
HEXS	64		
HIDDEN	194	LABEL	101, 119
HIDETURTLE	100	LAST	101
HLP	170	LC	102
HOME	100	LDADDR	181
horloge	30, 34	LEA	142
HT	100	LEFT	101
hz_200	240	LEFTS	65
		LEN	66
IERA - IERB	244	LET	51
IF	50, 100	LIB	170
IFFALSE	100	LINEATTR	101
IFTRUE	101	LINEF	51
IKBD	222	LINE INPUT	51
IKBDWS	233	LINK	142
ILLEGAL	142	LIST	51, 101
IMP	40	LISTP	101
imprimante	235, 236, 291	LLIST	52
IMRA - IMRB	245	LMODE	181
INIT	174	LOAD	52, 102
INITMOUS	226	LOADPIC	102
INP	64	LOC	66
INPUT	50	LOCAL	102
INPUT#	51	LOF	66
INPUT\$	65	LOG	66, 102, 170
INSTR	65	LOG10	66, 102
INT	65, 101	LOGBASE	227
INTIN	65, 87, 89	LOWERCASE	102
INTOUT	65, 87, 89	LPOS	66
IOREC	230	LPRINT	52
IPR	245	.LPT	93
IPRA - IPRB	244	LPUT	102

INDEX

LS	174	M_START	222
LSET	52	NAME	53, 103
LSL	142	NAMEP	103
LSR	142	NBCD	144
LT	101	NDIR	183
		NEG	145
MAKE	102	NEGX	145
manettes de jeux	29, 33, 296	NEW	53
MC 6850	24, 28, 265, 279	NEXT	53, 75
MD	174, 222	NFAT	174
MEDIA	183	nflpos	240
MEDIACH	225	NIID	183
mem....	238	NODES	103
MEMBERP	103	NOFORMAT	103
MERGE	52	NOP	145
MFP 68901	13, 14, 23, 25, 229, 230, 243, 273	NOT	40, 103, 132, 145
MFPOINT	229	NOTRACE	103
MIDS	52, 66	NOWATCH	103
MID	222	NOWRAP	174
MIDI	16, 222, 229, 230, 236, 250, 267, 289	NSECTS	183
MIDIWS	229	NSIDES	183
MKDS	66	NUMBERP	104
MKIS	67	nvbls	239
MKSS	67		
MMU	13, 180, 269, 284	OBJ	170
MOD	40	OCTS	67
MOUSE	103	OEM	181
MOVE	132, 133, 143, 174	OFFGIBIT	234
MOVEA	132, 143	OLD	53
MOVEM	143	ON ERROR	53
MOVEP	144	ONGIBIT	234
MOVEQ	133, 144	ON...GOSUB	53, 74
MULS	144	ON...GOTO	53, 74
MULU	144	OP	104
MPB	222	OPEN	54, 73
MP_MFL., MAL., ROVER	222	OPENW	55
M_ALLOC	216	opérateurs	39, 92
M_FREE	216	OPTION BASE	55
M_LENGTH	222	OR	40, 104, 145
M_LINK	222	ORI	132, 145
M_OWN	222	OUT	56
M_SHRINK	216	OUTPUT	104
		OVR	170
		.PAK	93
		PACKAGE	104
		PAL	104

INDEX

PALETTE	104	.PSG	22
palmode	239	PIRBLK	236
PAN	120	PTSIN	67, 87
PAS	170	PTSOUT	67, 87
PATH	104, 175	PU	105
PAUSE	104, 175	PUNTAES	237
PCIRCLE	56	PUT	58, 73, 175
PD	104	PUTBOOT	175
PE	104	PX	105
PEA	146	P_EXEC	216
PEEK	67, 83	P_TERM	217
PELLIPSE	56	P_TERM0	200
PENDOWN	104	P_TERMRES	208
PENERASE	104		
PENREVERSE	105	QUIT	58
PENUP	105	QUOTIENT	107
PHYSBASE	226		
phystop	238	RADIANS	107
PI	105	RAM	19
PIECE	105	RANDOM	107, 231
.PKG	93	RANDOMIZE	58
PKGALL	105	RC	108
PLIST	105	RD	175
PO	105	READ	58, 71
POALI.	106	READCHAR	108
POCALL	106	READLIST	108
POKE	56, 79, 83	READ ONLY	194
POLY	106	READQUOTE	108
PONS	106	RECYCLE	108
POPKG	106	REDEFP	108, 119
POPS	106	RELMOD	186
POREF	106	REM	58, 176
POS	67, 106	.REM	94
POTL	106	REMAINDER	108
POTS	106	REMPROP	108
PPROP	106	REN	176
PPS	107	RENUM	58
PR	107	REPEAT	109
PRG	170	REPLACE	59
PRIMITIVEP	107	.REPLACE	94
PRINT	56, 107	.REPTAIL	94
PRINT #	57	RERANDOM	109
PRINT USING	57	RES	182
.PRM	93	RESET	59, 146, 180, 238
PROCLIST	107	RESTORE	59
PRODUCT	107	RESUME	59, 72
PROTOBT	231	resvalid	238
PRT	222	resvector	238
prtnt	241		
PSG	234, 251		

INDEX

RETURN	59, 71	SETPALETTE	227
RIGHT	109	SETPAN	111
RIGHTS	67	SETPATH	111
RL	108	SETPC	112
RM	176	SETPEN	112
RND	67	SETPOS	112
ROL	146	SETPRT	235
ROM	19, 27	SETSCREEN	227
ROR	146	SETSCRUNCH	112
ROUND	109	SETTEXT	112
ROXL	146	SETTIME	232
ROXR	146	SETX	112
RQ	108	SETY	112
RS-232	14, 222, 230, 250, 292	SETZOOM	113
RSC	170	SF	109
RSCONF	230	SGN	68
RSET	59	shell_n	241
RSR	231, 248	SHIFT	225, 231
RT	109	SHIFTER	13, 14, 18, 269, 286
RTE	147	SHOW	113, 176
RTR	147	SHOWTURTLE	113
RTS	147, 231	SHUFFLE	113
RUN	60, 109	SIN	68, 113
RWABS	223	SORT	113
		souris	28, 31, 225, 226, 296
SAVE	60, 109	SPACES	68
SAVEPIC	109	SPC	68
savptr	240	.SPC	94, 182
SBCD	147	SPF	183
SCC	148	SPT	183
SCR	231, 247	SQR	69
SCRDMP	232	SQRT	113
SCREENFACTS	109	SSBRK	226
screenpt	239	SSECT	181
SE	110	sshiftmd	239
SECTCNT	181	ST	113
seekrate	239	STEP	60
SEI	245	STOP	60, 113, 148
SENTENCE	110	STR\$	68
SERIAL	182	STRINGS	68
SETBG	110	SUB	148
SETCOLOR	228	SUBA	148
SETEXC	224	SUBI	148
SETFILL	110	SUBQ	148
SETH	110	SUBX	149
SETHEADING	110	SUM	113
SETLINE	111	SUPERVISOR	205
SETPAL	111	SUPEXEC	237

INDEX

SWAP	60, 149	UCR	231, 247
sysbase	241	UDR	249
SYSTAB	69	UNBREAK	61
SYSTEM	60, 194	UNBURY	115
S_GETVEC	208	UNFOLLOW	61
S_SETVEC	206	UNLK	150
S_VERSION	207	UNTRACE	61
		UPPERCASE	115
		USR	78
TAB	69		
TACR - TBCR	245	VAL	69
TADR -...- TDDR	247	variables	39
TAI - TBI	246	VARPTR	70, 83
TAN	114	v_bas_ad	239
TAS	149	vbclock	240
TCDRC	247	vblqueue	239
TEST	114	vblsem	239
TEXT	114	VDI	87
TF	114	VDISYS	61, 87
THING	114	VERSION	176
THROW	114	VIDEORAM	13, 18
TICKAL	224	VOLUME LABEL	194
TIMER	238	VR	245
timr_ms	239	VSYNC	237
TMP	170		
TO	114	WAIT	61
TOPLEVEL	114	WATCH	115
TOS	170, 185	WAVE	61
TOWARDS	114	WD 1772	13, 15, 259, 277
TPA	179, 184	WEND	61, 75
TRACE	60, 114	WHERE	115
TRAP	26, 149, 220	WHILE	62, 75
TRAPV	149	WIDTH	62
TROFF	60	WINDOW	115
TRON	60	WORD	115
TRUE	114	WORDP	116
TSR	231, 249	WRAP	116, 120, 176
TST	149	WRITE	62
TT	115	WRITE#	62
TURTLEFACTS	114		
TURTLETEXT	115	XBTIMER	234
TV	83	XCOR	116
TYPE	115	XOFF	230, 231
T_GETDATE	206	XON	230, 231
T_GETTIME	207	XOR	40
T_SETDATE	206		
T_SETTIME	207	YCOR	116
UC	115		

CONSEILS DE LECTURE

Pour approfondir vos connaissances en BASIC, et mieux connaître le système des Atari 520 et 1040 ST, P.S.I. vous propose une palette d'ouvrages utiles.

Pour maîtriser le BASIC de l'Atari ST

- 102 programmes pour Atari ST, par Jacques Deconchat (Editions du P.S.I.)

102 jeux interactifs et graphiques vous font découvrir le BASIC des Atari 520 et 1040 ST, niveau par niveau ; pour débutants complets.

A paraître :

- La pratique de l'Atari ST, par Daniel-Jean David (Editions du P.S.I.)

Une prise en main complète du BASIC de l'Atari et de l'environnement des ST (GEM, TOS).

Pour mieux connaître le système des Atari ST

- Clefs pour Atari ST, 2. GEM, par Daniel Martin (Editions du P.S.I.)

Description détaillée de GEM et des appels à GEM; indispensable à tout programmeur sur Atari ST.

- Guide du programmeur de l'Atari ST, par K.D. Prael (Editions du P.S.I.)

Un ouvrage très complet et de haut niveau sur le système de base du ST et sur GEM : le livre de chevet de tous les programmeurs de l'Atari ST.

Achévé d'imprimer en février 1987
sur les presses de l'imprimerie Laballery
58500 Clamecy
Dépôt légal : février 1987

N° d'impression : 702022
N° d'édition : 86595-351-2
ISBN : 2-86595-351-3

Votre avis nous intéresse

Pour nous permettre de faire de meilleurs livres, adressez-nous vos critiques sur le présent ouvrage.

— Ce livre vous donne-t-il toute satisfaction ?

.....
.....
.....
.....

— Y a-t-il un aspect du problème que vous auriez aimé voir abordé ?

.....
.....
.....
.....

Si vous souhaitez des éclaircissements techniques, écrivez-nous, nous ne manquerons pas de vous répondre directement.

Où avez-vous acheté ce livre ?

- cadeau librairie autres
 exposition boutique micro

Comment en avez-vous eu connaissance ?

- publicité catalogue autres
 exposition conseils d'un ami

Avez-vous déjà acquis des livres P.S.I. ?

Lesquels ?

qu'en pensez-vous ?

Nom Prénom Age.....

Adresse

Profession

Centre d'intérêt

CATALOGUE GRATUIT

Vous pouvez obtenir un catalogue complet des ouvrages PSI, sur simple demande, ou en retournant cette page remplie à votre librairie, à votre boutique micro ou aux

Editions PSI
5, place du Colonel-Fabien
75491 PARIS CEDEX 10

mémento

Un mémento qui s'ouvre à la bonne page et vous permet d'accéder à toutes les informations dont vous avez besoin : configuration complète du matériel et de la mémoire, instructions et fonctions des langages Basic et Logo, interfacage avec GEM, commandes du système d'exploitation, organisation interne des disques, programmation et brochage des circuits spécialisés.

Mais "Clefs pour Atari ST" est aussi un recueil d'astuces qui vous apprendront à utiliser directement l'environnement matériel de votre machine. Des programmes vous feront découvrir toute l'originalité de votre Atari ST : comment simuler certaines instructions inopérantes, comment utiliser les routines internes de GEM, les fonctions définies par l'utilisateur, comment gérer les sons les plus divers, etc.

CLEFS POUR ATARI ST Système de base



9 782865 953516

Editions PSI
Diffusé par
P.C.V. Diffusion
BP 86
77402 Lagny s/Marne
Cedex
France

ISBN : 2.86595.351.3