

R.A.P.L. PROPULSE

au Capital de 50.000 Ffrs

50 RUE PROSPERITE

78200 NANTERRE LA VILLE

B.I.S. Immatriculé à 103 873 182

 **ATARI**

9, rue Saintou - 92150 SURESNES

TECHNIF 47210

Manuel de référence - Tome 2

OUTILS DE DEVELOPPEMENT

ATARI ST

 **ATARI**

TABLE DES MATIERES

CONTENU DES TROIS DISQUETTES	5
CHAPITRE 1: STRUCTURE D'UN PROGRAMME	11
1. GENERALITES	12
2. STRUCTURE D'UN FICHIER EXECUTABLE PLACE SUR SUPPORT.	12
3. STRUCTURE D'UN PROGRAMME EN MEMOIRE	13
4. PROCEDURE DE LANCEMENT	15
5. INITIALISATION DU PROGRAMME APPELE	16
6. RESERVATION DE MEMOIRE	17
CHAPITRE 2: COMPILATION EN LANGAGE C	19
1. LA BIBLIOTHEQUE C STANDARD	20
2. LA BIBLIOTHEQUE DES FONCTIONS VDI: VDIBIND	80
3. LA BIBLIOTHEQUE DES FONCTIONS AES: AESBIND	81
4. LA BIBLIOTHEQUE DES FONCTIONS SYSTEME: OSBIND	82
5. CONSEILS DE PROGRAMMATION EN LANGAGE C	83
CHAPITRE 3: LES OUTILS DE COMPILATION	89
1. INTRODUCTION	90
2. LE PRE-PROCESSEUR CP68	90
3. L'ANALYSEUR SYNTAXIQUE C068	91
4. LE GENERATEUR DE CODE C168	92
5. CONSEILS DE COMPILATION	92
CHAPITRE 4: L'OUTIL D'ASSEMBLAGE AS68	95
1. OPERATION D'ASSEMBLAGE	96
2. INITIALISATION D'AS68	96
3. LANCEMENT DE L'ASSEMBLAGE	96
4. EXEMPLES DE COMMANDES	98
5. DIRECTIVES DU LANGAGE ASSEMBLEUR	98
6. DIFFERENCES ENTRE LANGAGES ASSEMBLEUR	102
7. EXTENSIONS AU LANGAGE	103
8. MESSAGES D'ERREUR (Cf. ANNEXE A)	103
9. SOMMAIRE DU JEU D'INSTRUCTIONS	103
CHAPITRE 5: OUTILS D'EDITION DE LIENS, CONVERSION DE FORMAT.	107
1. INTRODUCTION	108
2. L'EDITEUR DE LIENS LINK68	108
3. L'EDITEUR DE LIENS LO68	112
4. L'UTILITAIRE RELMOD	115

CHAPITRE 6: L'UTILITAIRE DE MISE AU POINT SID68	117
1. LANCEMENT DE SID68	118
2. COMMANDES DE SID68	121
CHAPITRE 7: LES AUTRES UTILITAIRES	135
1. L'UTILITAIRE D'ARCHIVAGE AR68	136
2. L'EDITEUR DE COMMANDES: COMMAND.TOS	141
3. L'UTILITAIRE DE TRAITEMENT PAR LOTS: BATCH	151
4. L'UTILITAIRE D'AFFICHAGE: DUMP	153
5. L'UTILITAIRE D'INFORMATIONS SUR FICHIERS: SIZE68 ...	154
6. L'UTILITAIRE DE FORMAT "S-RECORD": SEND	156
7. L'UTILITAIRE D'AFFICHAGE DE SYMBOLES: NM68	159
8. L'UTILITAIRE DE REFERENCES: MMXREF	161
9. L'UTILITAIRE D'EFFACEMENT DE FICHIER: RM	162
10. L'UTILITAIRE DE TEMPORISATION: WAIT	162
ANNEXE A: LES MESSAGES D'ERREUR	163
A.1. LES MESSAGES D'ERREUR DE CP68	165
A.2. LES MESSAGES D'ERREUR DE C068	169
A.3. LES MESSAGES D'ERREUR DE C168	181
A.4. LES MESSAGES D'ERREUR D'AS68	183
A.5. LES MESSAGES D'ERREUR DE LINK68	190
A.6. LES MESSAGES D'ERREUR DE LO68	194
A.7. LES MESSAGES D'ERREUR DE AR68	197
A.8. LES MESSAGES D'ERREUR DE DUMP	199
A.9. LES MESSAGES D'ERREUR DE SIZE68	200
A.10. LES MESSAGES D'ERREUR DE SEND	201
A.11. LES MESSAGES D'ERREUR DE SID68	202
A.12. LES MESSAGES D'ERREUR DE COMMAND	205
ANNEXE B: CODES D'ERREUR CPM-68K	209
L'EDITEUR DE RESSOURCES "RCS"	211
1. Introduction à GEM RCS	211
2. Tutoriel de GEM RCS	220
3. Manuel de référence de GEM RCS	233
4. RSCREATE et autres informations techniques	260
Index alphabétique des thèmes	265

DISQUE NUMERO 1 VOLUME: MASTER1

Catalogue de A:\MASTER1.DEV

CARDS	<DOSSIER>
DOCUMENT	<DOSSIER>
DOODLES	<DOSSIER>
EMACS	<DOSSIER>
FORMDO	<DOSSIER>
ICON	<DOSSIER>
ICONA	<DOSSIER>
MENU	<DOSSIER>
RSCREAT	<DOSSIER>
TEMPLATE	<DOSSIER>
RCS2	<DOSSIER>

Catalogue de A:\MASTER1.DEV\CARDS

CARDS.C	7968	Exemple de programme utilisant les
CARD.H	465	fonctions AES de boîte de dialogue.
CARDS.H	376	
CTYPE.H	1607	
CTYPE.O	356	
CARDS.PRG	4536	
CARD.RSC	1096	
MAKECARD.BAT	249	

Catalogue de A:\MASTER1.DEV\DOCUMENT

NEWFORM.DOC	4927	Source de deux fonctions d'adressage
DISK.TXT	4211	du WD 1772 par l'intermédiaire du
		contrôleur DMA.

Catalogue de A:\MASTER1.DEV\DOODLES

LINKDO.BAT	79	Exemple d'un programme de dessin
DOODLE.C	54146	sous GEM.
DOODRSC.C	7481	
DOODLE.H	1358	
DOS.H	941	
TREEADDR.H	653	
DOOSTART.S	11348	
DOOSTART.O	1486	
DOODLE.PRG	20786	
DOODLE.RSC	3355	

Catalogue de A:\MASTER1.DEV\EMACS

ME.TTP	27782	Editeur de texte
TEXT1	896	
TEXT2	467	
TEXT3	4522	
TUTOR.DOC	127513	avec sa documentation anglaise.

Catalogue de A:\MASTER1.DEV\FORMDO

FORM.DEF	34	Exemple de programme utilisant la
FORM.H	75	fonction AES 'form_do'.
FORM.RSC	134	
FRMTST.C	1608	
FRMTST.PR	2525	

Catalogue de A:\MASTER1.DEV\ICON

ATBALL.C	2446	Exemple de programme.
ATBALL.S	2009	Source assembleur du programme précédent.
ATBALL.SHP	1604	
SDASM.PR	3685	
SDASM.S	15728	Exemple de programme.
SDC.C	7448	Exemple de programme.
SDC.PR	4368	Editeur d'images et d'icônes.
SDCST.S	13646	Exemple de programme.
SE.PR	31156	Editeur d'images et d'icônes.
SE.RSC	2896	
SHIC.DOC	20690	

Catalogue de A:\MASTER1.DEV\ICONA

ICED.PR	49020	Editeur d'icônes.
ICED.RSC	6948	
CLOCK.ICN	775	

Catalogue de A:\MASTER1.DEV\MENU

MLINK.BAT	99	
MENTST.C	2174	Exemple de programme utilisant les
MENTST.DEF	82	routines de gestion des menus déroulants.
MENTST.H	202	
MENTST.RSC	744	

Catalogue de A:\MASTER1.DEV\RSCREAT

DOC	699	
RSC.O	1408	
RSRCLIB.H	2669	
STCREATE.C	4435	Exemple de génération de
STCREATE.INP	44	fichiers RSC.

Catalogue de A:\MASTER1.DEV\TEMPLATE

TEMPLATE.C	611	Squelette d'initialisation d'une application GEM.
------------	-----	---

Catalogue de A:\MASTER1.DEV\RCS2

DEF2DFN.PR	13971	
RCS2.DFN	3264	
RCS2.PR	96083	Générateur de ressources
RCS2.RSC	21860	(création des fichiers '.RSC').
README	1058	

DISQUE NUMERO 2 VOLUME: MASTER2

Catalogue de A:\MASTER2.DEV

SHELL.UNX	<DOSSIER>	
ACSKEL.C	7958	Exemple d'un programme accessoire.
APSKEL.C	7259	Exemple d'une application GEM.
AR68.PR	20968	Utilitaire d'archivage (Chapitre 7, à1.).
DUMP.PR	10112	Utilitaire d'affichage (Chapitre 7, à4.).
FIND.PR	11648	Utilitaire de recherche de chaîne.
HIGH.PR	133	Passage en haute résolution.
LOW.PR	155	Passage en basse résolution.
NM68.PR	11904	Utilitaire d'affichage des symboles (Chapitre 7, à7.).
SID.TOS	28800	Utilitaire d'aide à la mise au point (Chapitre 5).
SIZE68.PR	12672	Utilitaire d'informations sur fichiers (Chapitre 7, à5.).
ACCSTART.S	4591	Source assembleur du lanceur d'accessoire (Chapitre 1, à5.).
APSTART.S	5296	Source assembleur d'un lanceur d'application GEM (Chapitre 1, à5.).
GEMSTART.S	24234	Source assembleur d'un lanceur d'application GEM (Chapitre 1, à5.).
COMMAND.TOS	22041	Interpréteur de commandes (Chapitre 7, à2.).
ENTETE.S	712	Source assembleur d'un lanceur TOS.
MMXREF.TTP	11487	Utilitaire de références (Chapitre 7, à8.).
LARGE.S	5995	Source de routines IBM.PC
SEND.PR	8192	Utilitaire pour format "S-RECORD" (Chapitre 7, à6.).
KERMIT.PR	44674	Programme de communication.
KERMIT.RSC	11856	
KERMIT.OPT	62	
LAESBIND	2342	

LGEMLIB	3978
LLIBF	1414
LVDIBIND	4065

Catalogue de A:\MASTER2.DEV\SHELL.UNX

SHELL.ACC	29130	Interpréteur de commandes en
RC.BAT	150	accessoire.
RCS.BAT	182	
SHELL.DOC	15429	
PROFILE	55	
RCLOS.BAT	304	
RS.BAT	87	

DISQUE NUMERO 3 VOLUME: MASTER3

Catalogue de A:\MASTER3.DEV

AESBIND	28416	Bibliothèque GEM-AES (Chapitre 2, à3..)
GEMLIB	84834	Bibliothèque des fonctions C standards (Chapitre 2, à1..)
LIBF	22338	Bibliothèque mathématique (Chapitre 2, à1..)
VDIBIND	55092	Bibliothèque GEM-VDI (Chapitre 2, à2..)
C.BAT	129	Fichier BATCH de compilation.
CLINK.BAT	81	Fichier BATCH d'édition de liens pour les programmes utilisant GEMLIB.
LINKACC.BAT	89	Fichier BATCH d'édition de liens pour les accessoires.
LINKAP.BAT	88	Fichier BATCH d'édition de liens pour les applications GEM.
LINKIO.BAT	99	Fichier BATCH d'édition de liens pour les applications GEM utilisant GEMLIB.
ACCSTART.O	858	Fichier objet du lanceur d'accessoires.
APSTART.O	928	Fichier objet du lanceur d'applications.
GEMSTART.O	2242	Fichier objet du lanceur d'applications utilisant GEMLIB.
OSBIND.O	180	Fichier objet d'accès au système (TRAP1, TRAP13, TRAP14).
LINK68.PRg	35072	Editeur de liens (Chapitre 5, à 2..)
LO68.PRg	19712	Editeur de liens (Chapitre 5, à 3..)
RELMOD.PRg	13805	Utilitaire de conversion (Chapitre 5, à 4..)
AS68INIT	6272	Fichier d'initialisation de l'assembleur.
AS68SYMB.DAT	6246	Fichier de paramétrage de l'assembleur.
CTYPE.H	1608	Fichier des fonctions tests de caractères.
DEFINE.H	692	Fichier des fonctions de base.
ERRNO.H	814	Fichier des fonctions d'erreur.
GEMBIND.H	11315	Fichier des définitions GEM.
GEMDEFS.H	4080	Fichier des définitions GEM de base.
MACHINE.H	5743	Fichier de portabilité machine.

MATH.H	1333	Fichier des déclarations des fonctions mathématiques.
OBDEFS.H	4614	Fichier des définitions pour objets GEM.
OSBIND.H	3906	Fichier des fonctions système.
OSIF.H	12201	Fichier des fonctions DOS.
OSIFERR.H	358	Fichier des fonctions erreurs système.
PORTAB.H	1974	Fichier de portabilité.
SETJMP.H	1491	Fichier de la fonction 'setjmp'.
STDIO.H	3392	Fichier standard.
STRING.H	969	Fichier des déclarations des fonctions chaînes.
TADDR.H	1015	Fichier des fonctions d'adresse d'arbres.
TOSDEFS.H	2209	Fichiers des fonctions TOS.
VDIBIND.H	2068	Fichier des déclarations des fonctions VDI.
AS68.PRg	52864	Assembleur (Chapitre 4).
CO68.PRg	51968	L'analyseur syntaxique (Chapitre 3, à3..).
CL68.PRg	47104	Le générateur de codes (Chapitre 3, à4..).
CP68.PRg	22144	Le pré-processeur (Chapitre 3, à2..).
WAIT.PRg	264	Temporisation (Chapitre 7, à10..).
RM.PRg	576	Effacement de fichier (Chapitre 7, à9..).
BATCH.TTP	2014	Traitement par lots (Chapitre 7, à3..).

EXEMPLES DE COMPILATION

ATTENTION: Faites immédiatement des copies de sauvegarde de vos trois disquettes de développement.

Sur votre disquette de travail n°3, créez un fichier 'desktop.inf' en cliquant l'option "Sauvegarder le Bureau".

Chargez le fichier 'desktop.inf' à partir d'un éditeur (EMACS par exemple et ajoutez en fin de fichier la ligne:

(tp 03 04 *.TTP @

De cette manière, tous les fichiers possédant le suffixe '.TTP' seront considérés comme des programmes "TOS avec paramètres". (Il est également possible de modifier les suffixes '.TTP' en suffixe '.APP' et de déclarer l'application comme "TOS avec paramètres"). Faites un RESET pour que le nouveau fichier 'desktop.inf' soit pris en compte.

Placez sur le disque n°3 le fichier source C 'APPSKEL.C' (présent sur le disque n°2). Le disque ne doit pas être protégé contre l'écriture.

Double-cliquez sur le programme 'BATCH.TTP'. Une boîte de dialogue s'ouvre pour l'entrée des paramètres.

Tapez la ligne suivante: c appskel
 → ATTENTION, n'entrez pas les suffixes des noms de fichiers. Cette ligne de commande lance le fichier 'C.BAT' avec comme argument le nom du fichier à compiler.

Après quelques minutes, la compilation est terminée et le programme attend l'appui d'une touche pour revenir au Bureau.

Après la lecture des différents chapitres traitant des utilitaires, il vous sera facile de modifier les fichiers '.BAT'. Il est par exemple utile de placer le fichier à compiler sur un disque différent ('B:' si vous possédez un second lecteur, ou sur un disque virtuel 'C:'). Référez vous au chapitre 3.à5 et au chapitre 5.à2 pour un stockage des fichiers temporaires hors du disque par défaut. Naturellement, les possesseurs d'un disque dur n'ont pas à se soucier de ces manipulations; la partition du disque d'où est lancée la compilation doit cependant disposer d'un espace disque suffisant.

1) Batch
Cn (nom de fichier)) fichier - 0

29) Balk
linkage n (nom. de liaison.) Pictures - PRG
executable
acc pour accessible
ap pour prog.

Que 1st
OK

CHAPITRE I

STRUCTURE D'UN PROGRAMME

- ✓ 1. GENERALITES
- ✓ 2. STRUCTURE D'UN FICHIER EXECUTABLE PLACE SUR SUPPORT
- ✓ 3. STRUCTURE D'UN PROGRAMME EN MEMOIRE
- ✓ 4. PROCEDURE DE LANCEMENT
- ✓ 5. INITIALISATION DU PROGRAMME APPELE
- ✓ 6. RESERVATION DE MEMOIRE

1. Généralités

L'une des tâches essentielles d'un système d'exploitation consiste à lire une application quelconque sur un support de masse, à la placer en mémoire vive et à lui donner le contrôle du processus.

La zone de mémoire disponible pour le stockage d'un fichier est appelée TPA ("Transient Program Area" ou zone de programme transitoire), et comprend tout l'espace non occupé par le système d'exploitation ou la mémoire écran.

Un programme chargé en mémoire est constitué de trois segments principaux:

- * La zone TEXTE qui contient tous les codes d'instructions du programme.
- * La zone DATA qui contient toutes les données initialisées du programme.
- * La zone BSS (Block Storage Segment) qui est réservée pour le stockage des données non-initialisées.

2. Structure d'un fichier exécutable placé sur support

Sur ATARI ST, un fichier exécutable stocké sur support comporte un certain nombre de segments permettant au système de disposer de points de repères. Le système d'exploitation GEMDOS "reconnait" les différentes zones suivantes:

- * L'en-tête du fichier

Zone de 28 octets contenant les informations suivantes:

\$00	\$60	\$1A
\$02	Nombre d'octets dans la zone TEXT	
\$06	Nombre d'octets dans la zone DATA	
\$0A	Nombre d'octets dans la zone BSS	
\$0E	Nombre d'octets dans la table des symboles	
\$12	Réservé, doit être à zéro	
\$16	Réservé, doit être à zéro	
\$1A	Réservé	

Le nombre \$601A dénote un fichier relogeable avec des zones TEXT, DATA et BSS localisées dans des segments contigus. Seuls les fichiers disposant de ce nombre "magique" pourront être exécutés par la routine de lancement du GEMDOS.

- * La zone TEXT

Contient les codes d'instructions machine du programme. Les opérandes des instructions ne désignant pas encore des adresses définies, le programme n'est pas encore exécutable. Il doit en premier lieu être relogé, c'est à dire adapté à l'emplacement mémoire dans lequel il est situé.

- * La zone DATA

Contient toutes les données initialisées du programme.

- * La table des symboles (optionnelle)

Contient les noms génériques des routines et variables du fichier source. Elle n'est utile que pour les logiciels de mise au point (SID par exemple, Cf. chapitre 6).

- * La table de relocation

Contient toutes les informations nécessaires pour transformer les adresses relatives du programme en adresses fixes. C'est par cette table qu'est assurée la relogeabilité.

Un fichier exécutable placé sur support a la structure suivante:

EN-TETE DE FICHIER
ZONE TEXTE (Codes des instructions)
ZONE DATA (Données initialisées)
TABLE DES SYMBOLES (Optionnelle)
TABLE DE RELOCATION

3. Structure d'un programme en mémoire

Lorsqu'un fichier exécutable est lu sur un disque afin d'être exécuté, la routine de lancement le stocke en mémoire selon un format spécifique comportant quatre zones distinctes:

- 1) La Page de Base

A partir notamment de l'en-tête du fichier, la routine de lancement crée la PAGE DE BASE du programme, stucture de 256 octets décrivant l'environnement général.

Cette Page de Base est constituée des éléments suivants:

FORMAT DE LA PAGE DE BASE

Adresse Relative	Longueur en octets	Description
\$00	4	Adresse inférieure de la TPA.
\$04	4	Adresse supérieure + 1 de la TPA.
\$08	4	Adresse du début de la zone TEXT (codes).
\$0C	4	Longueur en octets de la zone TEXT.
\$10	4	Adresse du début de la zone DATA.
\$14	4	Longueur en octets de la zone DATA.
\$18	4	Adresse du début de la zone BSS.
\$1C	4	Longueur en octets de la BSS.
\$20	4	Adresse de la DTA (Cf. 'Fsfirst').
\$24	4	Adresse de la Page de Base du programme appelant.
\$28	4	Réservé.
\$2C	4	Adresse de la chaîne d'environnement.
\$30	7	Identificateurs des fichiers standards.
\$37	1	Numéro du disque actif (0, 1, ...).
\$38	8	Réservés.
\$40	40	Numéros des chemins du catalogue.
\$68	24	Zone de sauvegarde des registres D0,A3,A4, A5,A6,A7 avant l'entrée dans le TRAP 1.
\$80	1	Longueur de la chaîne de commande.
\$81	127	Chaîne de commande si elle existe.

2) La zone TEXT: Zone des codes d'instructions

3) La zone DATA: Zone des données initialisées

4) La zone BSS: Zone des données non-initialisées

Les différentes zones du programme s'agencent en mémoire de la manière suivante:

HAUT DE LA MEMOIRE DISPONIBLE (HAUT DE LA TPA)

PILE UTILISATEUR
MEMOIRE DISPONIBLE
ZONE BSS (Données non-initialisées)
ZONE DATA (Données initialisées)
ZONE TEXTE (Codes des instructions)

BAS DE LA MEMOIRE DISPONIBLE (BAS DE LA TPA)

4. Procédure de lancement

Le chargement et l'exécution d'un programme est effectuée par la routine GEMDOS dont le nom générique est 'Pexec()' appelée par l'intermédiaire du TRAP 1 (fonction \$4B).

L'organigramme simplifié d'une procédure de lancement ressemble à ceci:

NOTE: Est décrite ici la procédure à partir du Bureau GEM, mais il est naturellement loisible à tout programme d'exécuter la routine 'Pexec()'.

BUREAU GEM: Le contrôle du processus appartient au SCREEN MANAGER, c'est à dire le programme gérant le Bureau.

- 1) Un "double-clic" est détecté sur un objet contenu dans un catalogue. L'utilisateur indique ainsi son désir d'ouvrir ou d'exécuter un fichier.
- 2) Le SCREEN MANAGER teste le type du programme.
 - a) Le fichier n'a pas le suffixe d'un programme exécutable: Afficher la boîte de dialogue "visualisation ou impression" et exécuter la routine correspondante.
 - b) Le fichier est exécutable et de type GEM:
 - Effacer les objets du Bureau GEM (barre des menus, icônes),
 - Mettre la souris en forme d'abeille,
 - Placer dans la barre des menus le titre du programme lancé,
 - Aller au point '3'.
 - c) Le fichier est exécutable et de type "TOS avec paramètres":
 - Ouvrir la boîte de dialogue d'entrée de paramètres et stocker les caractères entrés au clavier,
 - Désactiver la souris,
 - Nettoyer l'écran,
 - Aller au point '3'.
 - d) Le fichier est exécutable et de type "TOS":
 - Désactiver la souris,
 - Nettoyer l'écran,
 - Aller au point '3'.
- 3) Libération de l'espace mémoire réservé au SCREEN MANAGER (environ 40 Koctets)
- 4) Exécution de la routine de lancement (voir ci-dessous).

ROUTINE DE LANCEMENT 'Pexec()'

Lors de l'exécution de cette routine, les tâches successives suivantes sont accomplies:

- 1) Recherche de la première adresse mémoire disponible,
- 2) Réservation de 256 octets pour la création de la page de base,
- 3) Lecture des 28 premiers octets du fichier exécutable (en-tête de fichier).
- 4) Test des deux premiers octets: le nombre magique \$601A est-il présent? Si non présent, retour avec code d'erreur.

- 5) Création de la page de base, avec placement des arguments de la commande s'ils existent.
- 6) Remise à zéro de la mémoire depuis le haut de la page de base jusqu'au début de la mémoire d'écran.
- 7) Chargement du reste du fichier, c'est à dire la zone programme (TEXTE), la zone des données initialisées (DATA).
- 8) Chargement de la table de relocation et réécriture des adresses relogeables du programme.
- 9) Réserve de toute la mémoire disponible.
- 10) Pointeur de pile utilisateur placé en haut de la mémoire.
- 11) Contrôle donné à la première instruction du programme.

A la fin de l'exécution du programme appelé, le contrôle est redonné au programme appelant, c'est à dire, dans cet exemple, le Bureau GEM.

5. Initialisation du programme appelé

Les premières instructions exécutées dans un programme appartiennent généralement à un module objet distinct des autres modules du programme source. Il représente le premier fichier '.O' placé dans la ligne de commande lors d'une édition de liens.

Avec le pack de développement, trois modules objets de ce type (on les appelle des "lanceurs") sont fournis. Il s'agit de:

```
(
- ACCSTART.O
- APPSTART.O
- GEMSTART.O
```

* ACCSTART

Comme son nom le laisse supposer, ce lanceur est chargé d'exécuter diverses instructions relatives aux accessoires du Bureau GEM. Tout programme accessoire écrit devra donc être "lié" à ce module. Une ligne de commande de l'éditeur de liens LINK68 (Cf. chapitre 5, à2.) aurait l'apparence suivante:

```
link68 [u,s] acc.68k= accstart, accessoire, vdibind, aesbind
```

* APPSTART

'Appstart' est un lanceur pour les applications n'utilisant pas la librairie C standard 'GEMLIB'. Son intérêt réside dans la faible place mémoire qu'il occupe. Comme pour 'accstart', il représente le premier module à "lier".

```
link68 [u,s] app.68k = appstart, programme, vdibind, aesbind
```

* GEMSTART

Ce lanceur est nécessaire pour les applications utilisant les fonctions de la librairie C, nécessitant ainsi une édition de liens avec la librairie 'GEMLIB'. Par exemple:

```
link68 [u,s] prog.68k = gemstart, programme, vdibind, gemlib, libf
```

Ces lanceurs, après les instructions d'initialisation effectuées, se branchent à la routine 'main' de votre programme (soit 'main()' pour un source C, '-main' pour un source assembleur).

Vous pourrez être amené à écrire vous-même votre lanceur pour des raisons particulières (ainsi le programme DOODLE fourni comme exemple d'application GEM utilise son propre lanceur). Quoiqu'il en soit, il est utile de connaître l'une des tâches effectuées par chacun de ces trois lanceurs.

Au paragraphe 1.4 nous avons vu que la routine de lancement 'Pexec()' réservait la totalité de la mémoire et plaçait la pile utilisateur en haut de la TPA. Les trois lanceurs déjà mentionnés ont la charge de libérer la mémoire (par l'appel de 'Mshrink()', TRAP 1 fonction \$4A), de réserver l'emplacement mémoire de la zone 'BSS' (zone des données non-initialisées), ainsi qu'un Kilo-octets de pile utilisateur en initialisant le pointeur de pile 1024 octets au-dessus de la zone BSS. L'utilisation de ces lanceurs entraîne donc la configuration de mémoire présentée ci-dessous:

HAUT DE LA MEMOIRE DISPONIBLE (HAUT DE LA TPA)

MEMOIRE DISPONIBLE	
PILE UTILISATEUR: 1024 octets	pointeur de pile
ZONE BSS (Données non-initialisées)	
ZONE DATA (Données initialisées)	
ZONE TEXTE (Codes des instructions)	

BAS DE LA MEMOIRE DISPONIBLE (BAS DE LA TPA)

Il vous faudra, si vous pensez être limité par la taille de la pile, modifier ou réécrire le lanceur de votre application (les fichiers sources assembleur 'appstart.S', 'accstart.S' et 'gemstart.S' sont présents sur la disquette 'MASTER2' et peuvent vous fournir une aide précieuse).

6. Réserve de mémoire

Votre programme devra dans la plupart des cas réserver une partie de la mémoire pour son usage exclusif. La zone libre se situe depuis le haut de la pile jusqu'au bas de la mémoire d'écran. Pour ce faire, l'emploi de 'Malloc()' (TRAP 1, fonction \$48) est le plus fréquent puisque la fonction de la librairie C 'malloc()' n'autorise pas les réservations de plus de 64 Ko.

L'appel de 'Malloc()' avec -1L comme argument retourne la place mémoire disponible; l'appel de 'Malloc()' avec en argument

le nombre d'octets à réserver alloue à votre application la mémoire désirée.

Il est tentant de confisquer tout l'espace libre pour votre application. Mais nous ne saurions trop vous recommander de ne prendre que ce qui vous est nécessaire et, en tous cas, de TOUJOURS laisser un minimum de 10 Kilo-octets de mémoire disponible. Deux raisons justifient ce conseil:

- * un accessoire tournant en même temps que votre application peut avoir besoin de mémoire lors de son exécution (c'est à dire lorsque l'utilisateur clique l'item de l'accessoire).
- * certaines fonctions GEM font également une réservation de mémoire lors de leur exécution (ainsi 'fset_input' nécessite un minimum de 4 Ko).

CHAPITRE II

LA COMPILATION EN LANGAGE C

1. LA BIBLIOTHEQUE C STANDARD ✓
2. LA BIBLIOTHEQUE DES FONCTIONS VDI: VDIBIND ✓
3. LA BIBLIOTHEQUE DES FONCTIONS AES: AESBIND ✓
4. LA BIBLIOTHEQUE DES FONCTIONS SYSTEME: OSBIND ✓
5. CONSEILS DE PROGRAMMATION EN LANGAGE C ✓
 - 5.1. LA MODULARITE ✓
 - 5.2. CONVENTIONS DE CODE ✓

1. Fonctions de la librairie C standard

Cette section fournit une liste commentée des fonctions de la bibliothèque C standard. Cette liste est ordonnée de façon pseudo-alphabétique. Cela veut dire qu'en premier lieu, les fonctions proches ont été regroupées (ainsi 'index' et 'rindex', 'printf' et 'sprintf', etc...), les groupes obtenus ayant ensuite été classés par ordre alphabétique.

Pour le compilateur ALCYON de DIGITAL RESEARCH, toutes les fonctions explicitées ici sont contenues dans le fichier 'GEMLIB' et 'LIBF'. Pour certaines fonctions, la bibliothèque 'GEMLIB' fait elle-même référence à la bibliothèque mathématique 'LIBF'.

L'utilisation de ces bibliothèques impose d'une façon générale:

* l'inclusion dans le source C des fichiers <stdio.h> et <portab.h> (par la directive '#include').

* une édition de liens faisant référence aux bibliothèques précitées. L'appel type doit respecter la forme suivante:

```
link68 [?,?,?] ???68k=gemstart,???, ..., gemlib,libf
```

Les points d'interrogation représentent les données modifiables ou optionnelles.

Retenez que:

- * le lanceur 'gemstart' doit débiter la liste des fichiers à "lier" ('appstart' peut être utilisé lorsque la bibliothèque 'libf' est seule appelée).
- * 'libf', lorsqu'il est nécessaire, doit être placé derrière 'gemlib' et ne peut être suivi d'aucun autre fichier. (Dans de rares exceptions, 'libf' doit cependant précéder 'gemlib').

Note:

Dans toutes les explications qui suivent, il a été choisi de fournir, comme type d'argument, les substitutions symboliques classiques du langage C. Ces dernières devraient faciliter la portabilité des programmes source. Gardez cependant à l'esprit que les types 'WORD' ou 'BYTE' ne sont pas connus du compilateur C ALCYON. Le pré-processeur CP68 aura la charge, après lecture du fichier <portab.h>, de modifier ces symboles en un type reconnu.

Ainsi 'WORD' sera transformé en 'int', BYTE en 'char', etc...

N.d.T.:

L'intérêt de la bibliothèque C est évidemment d'assurer une portabilité maximum des programmes. Cependant, si vous décidez de développer une application sous GEM (faisant ainsi le choix d'une portabilité uniquement sous cet environnement), il est préférable d'utiliser, pour des fonctions équivalentes, l'appel aux routines de GEMDOS. Celles-ci offrent en effet un gain de place mémoire et accélèrent la rapidité d'exécution des programmes.

abort

La fonction 'abort' provoque l'arrêt du programme avec code d'erreur. L'erreur est dépendante du système. Le 68000 utilise un TRAP d'instruction illégale. Le contrôle est redonné au programme appelant (Bureau GEM, COMMAND programme, SID, etc...).

Séquence d'appel:

```
WORD code;

abort(code);
```

Argument:

code chargé dans le registre D0 avant arrêt

Retour:

Cette fonction ne revient jamais.

abs

La fonction 'abs' retourne la valeur absolue d'un argument entier. Elle est définie comme macro dans le fichier <stdio.h>. L'envoi d'arguments complexes peut provoquer des effets pervers.

Par exemple, l'appel: a = abs(*x++)
provoquera deux incrémentations de x.

Séquence d'appel:

```
WORD val;
WORD ret;

ret = abs(val);
```

Argument:

val valeur d'entrée

Retour:

ret valeur absolue de val

access

La fonction 'access' recherche la possibilité d'accès à un fichier. Le fichier est considéré comme accessible s'il existe.

Séquence d'appel:

```
BYTE *name;
WORD mode;
WORD ret;

ret = access(name,mode);
```

Arguments:

name	pointe vers la chaîne du nom de fichier
mode	peut être l'une de ces quatre valeurs:
4	vérifie l'accès en lecture
2	vérifie l'accès en écriture
1	vérifie qu'il s'agit d'un fichier exécutable
0	vérifie le chemin d'accès au répertoire
	Ce dernier argument est ignoré sous CP/M-68K.

Retour:

ret	zéro si l'accès au fichier est possible, -1 dans le cas contraire.
-----	---

atoi, atof, atol

Les fonctions 'atoi', 'atof', 'atol' convertissent une chaîne de chiffres ASCII en, respectivement, un entier, un flottant, un long mot.

- 'atoi' et 'atol' réclament une chaîne de la forme [-][+] dddd.
- 'atof' réclame une chaîne de la forme [-][+]ddddddd.dd[e[-]dd].

Chaque "d" représente un chiffre ASCII. Le compilateur ignore les espaces mais prend en compte le signe de la chaîne à convertir. La conversion s'opère chiffre après chiffre, jusqu'à ce que la fin de chaîne soit rencontrée. La fonction retourne zéro si la chaîne est vide.

Séquence d'appel:

```
BYTE *string;
WORD ival, atoi();
LONG lval, atol();
FLOAT fval, atof();

ival = atoi(string);
lval = atol(string);
fval = atof(string);
```

Arguments:

string	un pointeur vers la chaîne (terminée par zéro) contenant le nombre à convertir.
--------	--

Retours:

ival	la chaîne convertie en un entier
lval	la chaîne convertie en un long mot
fval	la chaîne convertie en un flottant simple précision

Note:

Ces trois fonctions ne détectent pas un débordement de capacité.

brk, sbrk

Les fonctions 'brk' et 'sbrk' gèrent l'espace réservé au programme utilisateur. La fonction 'brk' fixe l'adresse haute, limite supérieure de la mémoire réservée (appelée "break" dans la terminologie UNIX). La fonction 'sbrk' étend cette zone réservée du nombre d'octets spécifié.

Séquence d'appel:

```
WORD brk();
BYTE *addr, *sbrk();
WORD ret, incr;
BYTE *start;
```

```
ret = brk(addr);
start = sbrk(incr);
```

Arguments:

addr	pointeur vers le haut de l'application
incr	nombre d'octets à ajouter à la zone réservée

Retours:

ret	zéro en cas de réussite de l'opération (brk) -1 si échec (brk)
start	adresse de départ de la nouvelle zone réservée zéro si échec de l'opération (sbrk)

calloc, malloc, realloc, free

Les fonctions 'calloc', 'malloc', 'realloc' et 'free' gèrent l'espace mémoire alloué à l'application.

La fonction 'malloc' alloue une zone mémoire du nombre d'octets spécifié et retourne l'adresse du début de cette zone. Si nécessaire, 'malloc' appellera la fonction 'sbrk' (voir page précédente) pour permettre l'allocation.

La fonction 'calloc' alloue une zone mémoire pour une structure d'éléments, zone représentant n fois le nombre d'octets demandé.

La fonction 'realloc' modifie la taille d'une zone allouée. La nouvelle adresse de la zone est retournée.

La fonction 'free' libère une zone préalablement réservée par 'malloc'.

Séquence d'appel:

```
WORD size, number;
BYTE *addr, *malloc(), *calloc(), *realloc();
```

```
addr = malloc(size);
addr = calloc(number, size);
addr = realloc(addr, size);
free(addr);
```

Arguments:

size	nombre d'octets désiré
number	nombre d'éléments désiré
addr	pointeur vers le début de la zone allouée

Retour:

addr	adresse de départ de la zone allouée, zéro en cas d'échec
------	--

Note:

La fonction 'free' avec, comme argument, une adresse inexacte peut avoir des effets désastreux.

N.d.T.:

La taille de la zone étant définie par un mot, l'allocation maximum permise est de 65535 octets (voir Malloc, fonction gemdos, pour une réservation plus importante).

ceil

La fonction 'ceil' fournit l'entier arrondi par excès de l'argument spécifié.

Par exemple, ceil(1.5) retournera 2.0. La valeur retournée est un flottant.

Séquence d'appel:

```

FLOAT ceil();
FLOAT arg;
FLOAT ret;

ret = ceil(arg);

```

Argument:

arg un nombre flottant

Retour:

ret un nombre flottant, résultat de la fonction

chmod, chown

Sous UNIX, ces fonctions permettent de modifier les statuts d'accès et l'identificateur de propriété d'un fichier existant. Sous TOS, ces fonctions testent uniquement l'existence d'un fichier spécifié.

Séquence d'appel:

```

BYTE *name;
WORD mode, owner, group, ret;

ret = chmod(name, mode);
ret = chown(name, owner, group);

```

Arguments:

name	le nom du fichier concerné (terminé par zéro)
mode	le nouveau mode du fichier (UNIX)
owner	le nouveau propriétaire (UNIX)
group	le nouveau numéro de groupe (UNIX)

Retour:

ret	zéro si le fichier existe -1 dans le cas contraire
-----	---

close

La fonction 'close' ferme l'accès à un fichier ou à un périphérique. Elle agit sur les fichiers ouverts avec les fonctions 'open' et 'creat'. L'argument d'appel doit être un identificateur de fichier et non un pointeur de structure FILE (on utilise dans ce cas la fonction 'fclose').

Séquence d'appel:

```
WORD fd, ret;

ret = close(fd);
```

Argument:

fd l'identificateur du fichier à fermer

Retour:

ret zéro si fermeture réussie
 -1, identificateur de fichier inconnu

cos, sin

La fonction 'cos' retourne le cosinus d'un nombre flottant, la fonction 'sin' retourne le sinus d'un nombre flottant. Les arguments des deux fonctions sont exprimés en radians.

Séquence d'appel:

```
FLOAT cos(), sin();
FLOAT val, ret;

ret = cos(val);
ret = sin(val);
```

Argument:

val un nombre flottant qui exprime un angle en radians

Retour:

ret le cosinus ou sinus exprimé en radians

Note:

La précision maximum est obtenue avec des nombres inférieurs à 2 PI.

creat, creata, creatb

La fonction 'creat' ajoute un nouveau fichier au catalogue du disque. Un fichier ainsi créé est référencé par un identificateur (à ne pas confondre avec 'fcreat' qui retourne un pointeur de structure FILE).

Les fonctions 'creat' et 'creata' sont identiques et créent un fichier ASCII.

La fonction 'creatb' crée un fichier binaire.

Séquence d'appel:

```
BYTE *name;
WORD mode, fd;

fd = creat(name,mode);
fd = creata(name,mode);
fd = creatb(name,mode);
```

Arguments:

name	chaîne du nom du fichier, terminée par zéro
mode	le mode pour fichier UNIX, ignoré sous TOS

Retour:

fd	l'identificateur du fichier créé
	-1 si erreur

Note:

Un fichier créé est considéré comme étant ouvert. Si un fichier portant le même nom existait déjà, son contenu est perdu.

les fonctions ctype

Le fichier 'ctype.h' définit un certain nombre de fonctions qui permettent de classer un caractère ASCII, en indiquant si un caractère envoyé en argument appartient à une classe donnée:

- * une valeur différente de zéro est retournée si le caractère appartient à cette classe,
- * la valeur zéro est retournée dans le cas contraire.

La liste qui suit décrit les fonctions de 'ctype.h':

isalpha(c)	Teste si 'c' est une lettre.
isupper(c)	Teste si 'c' est une lettre majuscule.
islower(c)	Teste si 'c' est une lettre minuscule.
isdigit(c)	Teste si 'c' est un chiffre.
isalnum(c)	Teste si 'c' est alphanumérique.
isspace(c)	Teste si 'c' = ' ', '\t', '\n', '\r', '\f'.
ispunct(c)	Teste si 'c' est une ponctuation.
isprint(c)	Teste si 'c' est imprimable.
iscntrl(c)	Teste si 'c' est un caractère de contrôle.
isascii(c)	Teste si 0 <= 'c' <= 127.

Trois macro-instructions sont également définies dans le fichier 'ctype.h'. Il s'agit de:

```
#define tolower(ch) (isupper(ch) ? (ch)+('a'-'A') : (ch))
/* si 'ch' est une majuscule, il est converti en minuscule */

#define toupper(ch) (islower(ch) ? (ch)+('A'-'a') : (ch))
/* si 'ch' est une minuscule, il est converti en majuscule */

#define toascii(ch) ((ch) & 0x7F)
/* le bit fort de 'ch' est mis à zéro */
```

N.d.T.:

Les programmes sources utilisant ces fonctions doivent obligatoirement être liés avec la bibliothèque 'GEMLIB'.

etoa, ftoa

Convertissent un nombre flottant en chaîne ASCII. Ces deux fonctions retournent l'adresse du tampon de stockage de la chaîne résultante. La chaîne retournée dans le tampon se présente, pour 'etoa' sous la forme [-]d.ddd E[-]dd, pour 'ftoa' sous la forme [-]ddd.dd (chaque "d" représente un chiffre).

Séquence d'appel:

```

FLOAT fval;
BYTE *ftoa(), *etoa(), *buf, *ret;
WORD prec;

```

```

ret = etoa(fval,buf,prec);
ret = ftoa(fval,buf,prec);

```

Arguments:

fval	le nombre flottant à convertir
buf	l'adresse de stockage de la chaîne résultante
prec	la précision demandée, soit le nombre de chiffres à droite du point décimal (valeur de 0 à 5)

Retour:

ret	l'adresse de stockage de la chaîne résultante (identique à buf)
-----	---

N.d.T.:

La longueur de la chaîne retournée est au maximum de 6 caractères, point décimal compris s'il existe. La fonction retourne donc 6 chiffres significatifs pour un nombre entier, mais 5 seulement pour un nombre réel.

D'autre part, la partie entière du nombre est prioritaire par rapport à la partie décimale. La précision demandée dépend donc du nombre à convertir. Ainsi, pour une précision de 3:

```

1 retournera 1.000
mais 123.123 retournera 123.12

```

exit, _exit

La fonction 'exit' provoque la sortie du programme avec fermeture des fichiers ouverts, et désallocation des zones de mémoire allouées. La fonction '_exit' provoque une sortie immédiate, sans fermeture ni désallocation.

Séquence d'appel:

```

WORD code;

exit(code);
_exit(code);

```

Argument:

code	(optionnel) valeur retournée au programme appelant et éventuellement gérée par ce dernier.
------	--

Retour:

aucun retour

exp

Retourne la constante 'e' (2.71828182845905) élevé à la puissance spécifiée en argument.

Séquence d'appel:

```
FLOAT exp();  
FLOAT fval, ret;  
  
ret = exp(fval);
```

Argument:

fval le nombre flottant dont on désire l'exponentielle

Retour:

ret le résultat

fabs

Retourne la valeur absolue d'un nombre flottant.

Séquence d'appel:

```
FLOAT fabs();  
FLOAT fval, retval;  
  
retval = fabs(fval);
```

Argument:

fval un nombre flottant

Retour:

retval la valeur absolue de fval

fclose, fflush

La fonction 'fclose' ferme un fichier identifié par un pointeur de structure FILE, après avoir vidé le tampon qui lui est associé.

La fonction 'fflush' vide un tampon associé à un fichier.

Séquence d'appel:

```
WORD ret;
FILE *stream;

ret = fclose(stream);
ret = fflush(stream);
```

Argument:

stream pointeur vers la structure FILE du fichier

Retour:

ret zéro si opération réussie
 -1 si pointeur inexact ou erreur d'écriture

Note:

Voir à la fonction 'fopen' le détail de la structure FILE.

feof, ferror, clearerr, fileno

Toutes ces fonctions nécessitent comme argument un pointeur de structure FILE associé à un fichier.

La fonction 'feof' teste la fin d'un fichier spécifié. Elle renvoie zéro si la fin de fichier n'est pas rencontrée, une valeur différente de zéro dans le cas contraire.

La fonction 'ferror' renvoie une valeur différente de zéro si une erreur s'est produite dans le fichier spécifié. La fonction 'clearerr' supprime cette erreur. (Toutes deux utiles pour les fonctions ne gérant pas le traitement d'erreur, 'putw' par exemple).

La fonction 'fileno' retourne l'identificateur de fichier associé à une structure FILE.

Séquence d'appel:

```
WORD ret;
FILE *stream;
WORD fd;

ret = feof(stream);
ret = ferror(stream);
clearerr(stream);
fd = fileno(stream);
```

Argument:

stream pointeur vers la structure FILE du fichier

Retours:

ret un drapeau prenant la valeur zéro ou non-zéro
fd l'identificateur du fichier

Note:

Voir à la fonction 'fopen' le détail de la structure FILE.

floor

Retourne l'entier arrondi par défaut de la valeur flottante spécifiée. Par exemple, 'floor(1.5)' retournera la valeur 1.0.

Séquence d'appel:

```
FLOAT floor();  
FLOAT fval, retval;  
  
retval = floor(fval);
```

Argument:

fval un nombre flottant

Retour:

retval la valeur entière retournée

fmod

La fonction 'fmod' retourne le reste de la division (le modulo) de deux arguments flottants. La première valeur est divisée par la seconde.

Séquence d'appel:

```
FLOAT fmod();  
FLOAT x, y;  
FLOAT ret;  
  
ret = fmod(x,y);
```

Arguments:

x le dividende
y le diviseur

Retour:

ret le reste de la division

fopen, freopen, fdopen

Les fonctions 'fopen', 'freopen' et 'fdopen' associent une structure de type FILE (définie dans <stdio.h>) à un nom de fichier. Elles agissent différemment selon l'accès défini en argument (accès en lecture, en écriture ou en ajout).

Les fonctions 'fopen' et 'fopena' créent un fichier ASCII si l'accès se fait en écriture ou en ajout. Un fichier ouvert en écriture, s'il existe déjà, est détruit. En lecture, ces fonctions ouvrent un fichier existant. La fonction 'fopenb' agit de même, mais avec un fichier binaire.

Les fonctions 'freopen' et 'freopa' substituent un nouveau fichier ASCII à une structure FILE déjà utilisée, 'freopb' agit de même avec un fichier binaire.

La fonction 'fdopen' associe une structure FILE à un fichier déjà ouvert (par les fonctions 'open' ou 'creat').

Séquence d'appel:

```
FILE *fopen(), fopena(), fopenb();
FILE *freopen(), freopa(), freopb();
FILE *fdopen();
FILE *stream;
BYTE *name, *access;
WORD fd;
```

```
stream = fopen(name,access);
stream = fopena(name,access);
stream = fopenb(name,access);
stream = freopen(name,access,stream);
stream = freopa(name,access,stream);
stream = freopb(name,access,stream);
stream = fdopen(fd,access);
```

Arguments:

name	la chaîne du nom de fichier, terminée par zéro
stream	pointeur vers la structure FILE
access	pointeur vers la chaîne contenant:
	"r" pour la lecture de fichier
	"w" pour l'écriture sur fichier
	"a" pour l'ajout à un fichier

Retour:

stream	pointeur vers la structure FILE associée au fichier
	Si erreur, retourne un pointeur nul

fopen, freopen, fdopen

Note:

Rappelons, pour mémoire, la déclaration de la structure de type FILE et la signification de ses différents membres.

```
typedef struct _iobuf {
    WORD _fd; /* identificateur système du fichier */
    WORD _flag; /* mode d'ouverture du fichier (0,1,2) */
    BYTE *_base; /* pointeur de début de tampon */
    BYTE *_ptr; /* pointeur sur l'octet courant */
    WORD _cnt; /* nbre d'octets qui ont été lus/écrits */
} FILE;
```

Le fichier <stdio.h> sur ATARI déclare un tableau de 16 structures de ce type, les trois premières de celles-ci étant déjà ouvertes par le système pour gérer le périphérique d'entrée (stdin), le périphérique de sortie (stdout), le périphérique d'erreur (stderr).

fread, fwrite

La fonction 'fread' transfère des données d'un fichier vers la mémoire centrale, la fonction 'fwrite' transfère des données de la mémoire centrale vers un fichier.

Séquence d'appel:

```
WORD nitems;
BYTE *buff;
WORD size;
FILE *stream;
```

```
nitems = fread(buff,size,nitems,stream);
nitems = fwrite(buff,size,nitems,stream);
```

Arguments:

buff	adresse du tampon de stockage
size	nombre d'octets dans chaque item
nitems	le nombre d'items à transférer
stream	pointeur de structure FILE du fichier concerné

Retour:

nitems	le nombre d'items lus ou écrits zéro si erreur (par exemple: EOF rencontré)
--------	--

Note:

Sur ATARI ST, la taille logique d'un item de transfert est de 512 octets.

N.d.T.:

La fonction 'fread' ne stocke pas dans son tampon de lecture les éventuels codes de Retour Chariot (0x0D) lus dans un fichier.

fseek, ftell, rewind

Ces trois fonctions agissent sur le pointeur de l'octet courant d'un fichier (voir 'fopen').

La fonction 'fseek' positionne le pointeur en décalant ce dernier du nombre d'octets spécifié, la fonction 'rewind' place le pointeur en début de fichier, la fonction 'ftell' demande la position courante du pointeur. Celle-ci est définie par rapport au début du fichier.

Ces fonctions n'ont aucun effet si le pointeur de structure FILE envoyé comme argument décrit un périphérique.

Séquence d'appel:

```
WORD ret;
FILE *stream;
LONG offset, ftell();
WORD ptrname;
```

```
ret = fseek(stream,offset,ptrname);
ret = rewind(stream);
offset = ftell(stream);
```

Arguments:

stream	pointeur de structure FILE du fichier concerné
offset	un long mot signé définissant un décalage relatif
ptrname	une valeur interprétant le décalage si 0 => à partir du début du fichier si 1 => à partir de la position courante si 2 => à partir de la fin de fichier

Retours:

ret	zéro si succès, -1 si erreur
offset	position du pointeur de l'octet courant

getc, getchar, fgetc, getw, getl

Ces fonctions lisent une valeur envoyée par un fichier ou un périphérique.

Les fonctions 'getc' et 'fgetc' lisent un caractère à partir du fichier spécifié. Dans <stdio.h>, 'getc' est défini comme 'fgetc'.

La fonction 'getchar' lit un caractère à partir du fichier d'entrée standard. Le fichier <stdio.h> définit cette fonction comme 'fgetc(stdin)'.

La fonction 'getw' lit un mot de 16 bits à partir du fichier spécifié, la fonction 'getl' lit un long mot de 32 bits.

Séquence d'appel:

```
WORD ichar;
FILE *stream;
WORD iword;
LONG ilong, getl();
```

```
ichar = getc(stream);
ichar = getchar();
ichar = fgetc(stream);
iword = getw(stream);
ilong = getl(stream);
```

Argument:

stream pointeur vers structure FILE du fichier concerné

Retours:

```
ichar      le caractère lu
iword      le mot (16 bits) lu
ilong      le long mot (32 bits) lu
-1          une erreur de lecture
```

Note:

La valeur -1 retournée peut ne pas être une erreur, particulièrement avec 'getw' et 'getl'. Utilisez 'feof' ou 'ferror' pour détecter la fin de fichier ou une erreur de lecture.

N.d.T.:

Le code de retour chariot (0x0D) ne provoque pas la sortie de la fonction 'fgetc', ni à fortiori de la fonction 'getchar'.

getpass

Cette fonction affiche une chaîne d'invite sur l'écran et lit un mot de passe entré au clavier sans en afficher l'écho. Elle retourne un pointeur vers la chaîne reçue, pouvant contenir jusqu'à 8 caractères et terminée par un caractère nul.

Séquence d'appel:

```
BYTE *prompt;
BYTE *getpass();
BYTE *pass;
```

```
pass = getpass(prompt);
```

Argument:

prompt un pointeur vers la chaîne d'invite
 (terminé par caractère nul)

Retour:

pass un pointeur vers la chaîne lue

Note:

Tout appel à cette fonction réécrit sur le "mot de passe" préalablement lu.

gets, fgets

Les fonctions 'gets' et 'fgets' lisent un chaîne de caractères à partir d'un fichier ou d'un périphérique.

La fonction 'gets' lit le périphérique d'entrée standard (stdin) jusqu'à recevoir un caractère d'interligne (0X0A). Elle renvoie alors le pointeur de la chaîne, terminée par le caractère nul mais n'incluant pas le code d'interligne.

La fonction 'fgets' lit, sur le fichier spécifié, un nombre déterminé de caractères. Un retour prématuré de la fonction sera cependant provoqué si le code d'interligne est lu. ce dernier sera, dans ce cas, intégré à la chaîne.

Séquence d'appel:

```
BYTE *addr;
BYTE *s;
BYTE *gets(), *fgets();
WORD n;
FILE *stream;

addr = gets(s);
addr = fgets(s,n,stream);
```

Arguments:

s	pointeur vers le tampon de stockage de chaîne
n	le nombre maximum de caractères à lire
stream	pointeur vers la structure FILE du fichier lu

Retour:

addr	pointeur vers le tampon de la chaîne lue
------	--

index, rindex

Ces fonctions cherchent et localisent un caractère dans une chaîne. La fonction 'index' retourne un pointeur vers la première occurrence du caractère recherché, 'rindex' retourne un pointeur vers la dernière occurrence.

Séquence d'appel:

```
BYTE *c;
BYTE *s, *ptr;
BYTE *index(), *rindex();

ptr = index(s,c);
ptr = rindex(s,c);
```

Arguments:

s	pointeur vers une chaîne terminée par zéro
c	le caractère recherché

Retour:

ptr	l'adresse du caractère recherché si zéro, caractère non présent dans la chaîne
-----	---

isatty

La fonction 'isatty' détermine si un identificateur de fichier est rattaché à la console (CON:).

Séquence d'appel:

```
WORD fd;  
WORD ret;  
  
ret = isatty(fd);
```

Argument:

fd un identificateur de fichier ouvert

Retour:

ret 1, fd rattaché à la console
 0, fd non rattaché à la console

log

Retourne le logarithme népérien d'une valeur flottante.

Séquence d'appel:

```
FLOAT log();  
FLOAT fval, ret;  
  
ret = log(fval);
```

Argument:

fval un nombre flottant

Retour:

ret le logarithme népérien de fval

lseek, tell

La fonction 'lseek' positionne le pointeur d'octet courant d'un fichier en décalant ce pointeur du nombre d'octets spécifié en argument.

La fonction 'tell' retourne la position courante de ce pointeur.

Pour interroger un fichier spécifique, ces deux fonctions envoient en paramètre l'identificateur du fichier (renvoyé par 'open') et non un pointeur vers une structure FILE (renvoyé par 'fopen'). Dans ce second cas, ce sont les fonctions 'fseek' et 'ftell' qu'il faut utiliser.

Séquence d'appel:

```
WORD fd;
WORD ptrname;
LONG offset, lseek(), tell(), ret;

ret = lseek(fd, offset, ptrname);
ret = tell(fd);
```

Arguments:

```
fd      l'identificateur du fichier
offset  un long mot signé définissant un décalage relatif
ptrname une valeur interprétant le décalage
        si 0 => à partir du début du fichier
        si 1 => à partir de la position courante
        si 2 => à partir de la fin de fichier
```

Retour:

```
ret      position du pointeur de l'octet courant
        -1 si erreur
```

mktemp

Crée un nom de fichier temporaire. L'argument d'appel est une chaîne de caractères composée de 6 fois la lettre "X", le nom du fichier est réécrit sur cette chaîne.

Séquence d'appel:

```
BYTE *string;
BYTE *mktemp();

string = mktemp(string);
```

Argument:

```
string      pointeur vers la chaîne de gabarit
```

Retour:

```
string      pointeur vers la chaîne résultante
```

open, opena, openb

Les fonctions 'open' et 'opena' ouvrent un fichier ASCII existant et retournent un identificateur de fichier.

La fonction 'openb' ouvre un fichier binaire existant et retourne un identificateur de fichier.

Un fichier qui n'existe pas doit auparavant être créé avec la fonction 'creat'.

Séquence d'appel:

```
BYTE *name;
WORD mode;
WORD fd;
```

```
fd = open(name,mode);
fd = opena(name,mode);
fd = openb(name,mode);
```

Arguments:

name	pointe la chaîne du nom de fichier, terminée par 0
mode	l'accès désiré:
	0 => Lecture uniquement
	1 => Ecriture uniquement
	2 => Lecture/Ecriture

Retour:

fd	l'identificateur du fichier ouvert
	si -1, erreur à l'ouverture

perror

Envoie sur le fichier d'erreur standard un message qui décrit la dernière erreur rencontrée par le système d'exploitation. La chaîne de caractères envoyée en argument est affichée avant le message du système.

L'annexe B décrit les différents messages que la librairie envoie en fonction de l'erreur détectée.

Séquence d'appel:

```
BYTE *s;
WORD err;
```

```
err = perror(s);
```

Argument:

s	pointe la chaîne à afficher
---	-----------------------------

Retour:

err	numéro d'erreur avant l'appel
-----	-------------------------------

pow

Retourne la valeur d'un nombre élevé à une puissance. Les valeurs d'appel et de retour sont des flottants.

Séquence d'appel:

```

FLOAT pow();
FLOAT x, y;
FLOAT ret;

```

Arguments:

x	la mantisse, nombre flottant
y	l'exposant, nombre flottant

Retour:

ret	le résultat flottant de x à la puissance y
------------	--

printf, fprintf, sprintf

Ces trois fonctions convertissent des données, les mettent en forme et les orientent vers une sortie.

Les données formatées sont orientées:

- pour 'printf', vers le périphérique standard de sortie (stdout);
- pour 'fprintf', vers le fichier défini en paramètre;
- pour 'sprintf', vers la mémoire.

Séquence d'appel:

```

WORD ret;
BYTE *fmt;
FILE *stream;
BYTE *string;
BYTE *sprintf(), rs;
/* Les autres arguments peuvent être de type divers */

```

```

ret = printf (fmt, arg1, arg2.....);
ret = fprintf(stream, fmt, arg1, arg2...);
rs = sprintf(string, fmt, arg1, arg2...);

```

Arguments:

fmt	chaîne spécifiant le formatage (cf page suivante)
arg1...	les arguments à convertir selon fmt
stream	pointeur vers la structure FILE définissant le fichier de sortie
string	adresse du tampon en mémoire

Retours:

ret	nombre de caractères sortis -1 si erreur
rs	pointeur vers la chaîne formatée zéro si erreur

printf, fprintf, sprintf

La chaîne de formatage:

La chaîne de formatage peut contenir du texte à sortir littéralement et des opérateurs de conversion s'appliquant aux arguments envoyés comme paramètres. Chaque opérateur est précédé du caractère %. La table qui suit définit les opérateurs valides:

- d 1'argument est converti en un décimal.
- o 1'argument est converti en un nombre en base 8 (octal).
- x 1'argument est converti en un nombre en base 16 (hexadécimal).
- c 1'argument est converti en un caractère ASCII.
- s 1'argument est une chaîne à afficher.
- f 1'argument est un flottant, il est converti en un décimal de la forme [-]dddd.dddd.

Un certain nombre de caractères optionnels peuvent être insérés entre le symbole % et l'opérateur de conversion:

- Le signe "-" (moins) spécifie une justification à gauche, la justification à droite étant choisie par défaut.
- Un nombre spécifie la largeur minimale du champ. Si ce nombre est précédé du chiffre 0, le remplissage des emplacements libres sera effectué par le caractère "0" au lieu des espaces.
- Le caractère "." (point) suivi d'un nombre donne la précision de la partie fractionnaire d'un flottant. Si l'argument à formater n'est pas une valeur flottante, le nombre suivant le point représente la taille maximale d'impression.
- Le caractère "l" ou "L" précise que la valeur à formater est un long mot.

Exemples: Soit la chaîne "bonjour" (7 caractères) et le flottant de valeur 123.456 (la largeur du champ est délimitée par des guillemets).

%10s	donne	" bonjour"
%-10s	donne	"bonjour "
%10.5s	donne	" bonjo"
%10.2f	donne	" 123.45"
%010.2f	donne	"0000123.45"

putc, putchar, fputc, fputw, putl

Envoi d'une valeur à un fichier ou à un périphérique.

Les fonctions 'putc' et 'fputc' écrivent un caractère sur le fichier spécifié. Dans <stdio.h>, 'putc' est défini comme 'fputc'.

La fonction 'putchar' écrit un caractère sur le fichier de sortie standard. Le fichier <stdio.h> définit cette fonction comme 'fputc(stdout)'.

La fonction 'putw' écrit un mot de 16 bits sur le fichier spécifié, la fonction 'putl' écrit un long mot de 32 bits.

Séquence d'appel:

```
BYTE c;
FILE *stream;
WORD w, ret;
LONG lret, l, putl();
```

```
ret = putc(c, stream);
ret = fputc(c, stream);
ret = putchar(c);
ret = putw(w, stream);
lret = putl(l, stream);
```

Arguments:

c	le caractère à envoyer
stream	un pointeur vers la structure FILE du fichier
w	le mot à envoyer
l	le long mot à envoyer

Retours:

ret	le caractère ou mot envoyé
lret	le long mot envoyé
-1	une erreur possible est détectée

Note:

La valeur -1 retournée peut ne pas être une erreur, particulièrement avec 'putw' et 'putl'. Utilisez 'ferror' pour détecter une erreur d'écriture.

puts, fputs

Les fonctions 'puts' et 'fputs' écrivent un chaîne de caractères, terminée par le caractère nul, sur un fichier ou vers un périphérique.

La fonction 'puts' envoie une chaîne vers le périphérique de sortie standard (stdout) et y ajoute un caractère d'interligne (0x0A). La fonction 'fputs' envoie une chaîne vers le fichier spécifié, sans cependant y ajouter de caractère d'interligne.

Aucune de ces deux fonctions n'envoie le caractère nul de fin de chaîne.

Séquence d'appel:

```
WORD ret;
BYTE *s;
FILE *stream;
```

```
ret = puts(s);
ret = fputs(s,stream);
```

Arguments:

s	pointeur de la chaîne à envoyer
stream	pointeur vers structure du fichier destinataire

Retour:

ret	le dernier caractère envoyé si -1, erreur
-----	--

qsort

La fonction 'qsort' est une routine de tri rapide à qui l'on fournit un tableau d'éléments à trier et l'adresse d'une routine de comparaison de deux éléments. Au retour de la fonction, le tableau est trié.

Séquence d'appel:

```
WORD ret;
BYTE *base;
WORD number;
WORD size;
WORD compare();
```

```
ret = qsort(base,number,size,compare);
```

Arguments:

base	adresse basse des éléments à trier
number	le nombre d'éléments à trier
size	la taille en octets de chaque élément
compare	l'adresse de la fonction de comparaison

Retour:

ret	toujours zéro
-----	---------------

Note:

'qsort' appelle la routine de comparaison comme suit:

```
ret = compare(a,b);
```

Le retour doit être:

inférieur à zéro	si a doit être classé avant b
égal à zéro	si a est équivalent à b
supérieur à zéro	si a doit être classé après b

rand, srand

La fonction 'rand' retourne un nombre aléatoire quelconque, la fonction 'srand' retourne un nombre aléatoire déduit par la "semence" fournie en argument. Le nombre renvoyé est un entier machine.

Séquence d'appel:

```
WORD seed;
WORD rnum;

rnum = srand(seed);
rnum = rand();
```

Argument:

seed semence du nombre aléatoire

Retour:

rnum le nombre aléatoire

N.d.T.:

On peut utiliser aussi la fonction 'Random()' (bios étendu) qui retourne un nombre aléatoire sur 24 bits.

read

Lit des données sur le fichier spécifié, défini par son identificateur (identificateur retourné par les fonctions 'open' ou 'creat'). La lecture débute à la position du pointeur d'octet courant, le nombre d'octets à lire est passé en argument.

Séquence d'appel:

```
WORD ret;
WORD fd;
BYTE *buffer;
WORD bytes;

ret = read(fd,buffer,bytes);
```

Arguments:

fd l'identificateur du fichier à lire
buffer pointeur du tampon de réception
bytes le nombre d'octets à lire

Retour:

ret le nombre d'octets effectivement lus
 si -1, erreur

scanf, fscanf, sscanf

A partir d'une entrée, ces trois fonctions attendent des données, les mettent en forme et les assignent aux arguments fournis en paramètres. Ces fonctions sont aux entrées de données ce que les fonctions 'printf', 'fprintf', et 'sprintf' sont aux sorties.

Les données à formater sont attendues:

- pour 'scanf', du périphérique standard d'entrée (stdin);
- pour 'fscanf', du fichier défini en paramètre;
- pour 'sscanf', d'une chaîne en mémoire.

ATTENTION: Les arguments de ces trois fonctions doivent être des adresses.

Séquence d'appel:

```
BYTE *format, *string;
WORD nitems;
FILE *stream;
/* les arguments peuvent être de type quelconque */
```

```
nitems = scanf (format,&arg1,&arg2...);
nitems = fscanf(stream,format,&arg1,&arg2...);
nitems = sscanf(string,format,&arg1,&arg2...);
```

Arguments:

format	chaîne spécifiant le formatage (cf page suivante)
arg1...	les arguments récepteurs des entrées
stream	pointeur vers la structure FILE définissant le fichier d'entrée
string	adresse de la chaîne d'entrée

Retour:

nitems	le nombre d'entrées correctement traitées
	si -1, erreur (EOF par exemple)

Note:

Le retour de ces fonctions s'opère lorsque l'entrée des arguments est complète ou qu'une erreur est détectée (par exemple une lettre pour un argument décimal attendu).

Lorsque les entrées sont multiples, les différents champs sont séparés par des espaces. (N.d.T.: Le code de retour chariot est inopérant. cf 'fgetc').

scanf, fscanf, sscanfLa chaîne de formatage:

Elle contient les spécifications de conversions permettant d'interpréter les séquences d'entrée. Elle peut se composer:

* de caractères <espace>, <tabulation>, <interligne> ou ASCII (non précédés de %) qui ne sont pas pris en compte mais qui doivent représenter le premier caractère lu en entrée.

* du caractère % suivi d'un spécificateur de conversion compris dans cette liste:

- d la donnée en entrée est un entier décimal.
- o la donnée en entrée est un entier en base 8.
- x la donnée en entrée est un entier en base 16.
- s La donnée en entrée est une chaîne terminée par le caractère <espace> ou par les limites du tableau récepteur.
- c La donnée en entrée est un caractère ASCII unique.
- f La donnée en entrée est un flottant: [+] [-] ddd.ddd [e [-] dd]

Un certain nombre de caractères optionnels peuvent être insérés entre le symbole % et l'opérateur de conversion:

- Un nombre spécifiant la largeur maximum du champ d'entrée.
- Le caractère "*" qui supprime l'affectation.

Exemples:

```
scanf ("%2d %*s %2d %*s", &heure, &minute);
```

Avec en entrée: 15 H 28 Mn ,
la valeur 15 est affectée à 'heure', la valeur 28 à 'minute'.

Même chose d'ailleurs si l'entrée est : 153 H 289 Mn

```
scanf (" +%s", chaîne);
```

oblige l'utilisateur à taper le signe '+' avant la chaîne mais ne l'intègre pas à l'argument.

setjmp, longjmp

Ces fonctions permettent d'exécuter un GOTO externe à une fonction. L'appel se fait en deux temps:

- 1) Juste avant l'instruction où l'on souhaite que soit effectué un branchement absolu, un appel à 'setjmp' doit être fait.
- 2) Dans une autre fonction, l'appel à 'longjmp' permet un branchement absolu à l'instruction 'setjmp' en fournissant un paramètre à cette dernière.

ATTENTION: l'instruction 'longjmp' n'est permise que si la fonction contenant 'setjmp' n'a pas rencontré l'accolade fermante de fin de fonction.

Séquence d'appel:

```

#include <setjmp.h>
WORD xret, ret;
jmp_buf env;      /* défini comme tableau de longs */
                  /* dans <setjmp.h> */

{
    .
    .
    .
    xret = setjmp(env);
    .
    .
    .
    longjmp(env, ret);
    .
}

```

Arguments:

env	tableau défini dans <setjmp.h> et préservant l'environnement.
ret	la valeur renvoyée de 'longjmp' vers 'setjmp'

Retour:

xret	zéro au premier appel de 'setjmp' la valeur de 'ret' lors d'un retour par 'longjmp'
------	--

signal

La fonction 'signal' permet d'assurer un déroutage, lors d'un TRAP exception du 68000, vers une fonction C. Le numéro de l'exception concernée est fourni à l'appel de la fonction.

Voici les numéros d'exceptions valides:

- * 4 "Instruction illégale", ainsi que "violation de privilège", "ligne A", "ligne F".
- * 5 Mode TRACE.
- * 6 Tout TRAP autre que les TRAP £2, £3, £8.
- * 8 TRAP arithmétiques, c'est-à-dire "division par zéro", "CHK" (donnée hors d'un intervalle), "TRAPV" (dépassement de capacité).
- * 10 "Erreur BUS" et "erreur adresse".

Tout autre valeur est ignorée afin d'assurer la compatibilité avec UNIX.

Au retour d'une fonction activée par 'signal', le processus normal du programme est repris: la routine s'occupe en effet de préserver puis de restituer l'état des registres.

Séquence d'appel:

```

WORD ret, sig;
WORD func();

ret = signal(sig, func);

```

Arguments:

sig	numéro d'exception (voir plus haut)
func	adresse de la fonction C de déroutage

Retour:

ret	zéro si aucune erreur, -1 si numéro non valide
-----	---

sinh, tanh

Retournent respectivement le sinus hyperbolique et la tangente hyperbolique d'un nombre flottant. L'argument est exprimé en radians.

Séquence d'appel:

```
FLOAT sinh(), tanh();  
FLOAT fval, ret;
```

```
ret = sinh(fval);  
ret = tanh(fval);
```

Argument:

fval un flottant exprimant un angle en radians

Retour:

ret sinus ou tangente hyperbolique de l'argument

sqrt

Retourne la racine carrée d'un nombre flottant.

Séquence d'appel:

```
FLOAT sqrt();  
FLOAT fval, ret;  
  
ret = sqrt(fval);
```

Argument:

fval un nombre flottant

Retour:

ret la racine carrée de l'argument

strcat, strncat

Fonctions de concaténation de chaînes de caractères.

La fonction 'strcat' ajoute à la chaîne "s1" la chaîne "s2".

La fonction 'strncat' ajoute à la chaîne "s1" n caractères de la chaîne "s2". Les chaînes en argument doivent être terminées par le caractère nul.

Séquence d'appel:

```
BYTE *s1, *s2, *ret;
BYTE *strcat(), *strncat();
```

```
ret = strcat(s1,s2);
ret = strncat(s1,s2,n);
```

Arguments:

s1	la première chaîne
s2	la seconde chaîne, ajoutée à s1
n	maximum de caractères de s2 à ajouter à s1

Retour:

ret pointeur vers "s1" contenant la chaîne concaténée

Note:

La fonction 'strcat (s1,s1)' a toutes les chances de "planter" le système.

strcmp, strncmp

Fonctions de comparaison de chaînes.

La fonction 'strcmp' compare deux chaînes, la fonction 'strncmp' limite la comparaison aux n premiers caractères. Les chaînes en arguments doivent être terminées par le caractère nul.

La valeur retournée par ces fonctions est égale à zéro si les chaînes (ou les n premiers caractères des chaînes) sont identiques. Sinon, est renvoyé le résultat de la soustraction entre les premiers caractères ASCII différents.

Séquence d'appel:

```
BYTE *s1, *s2;
WORD val, n;
```

```
val = strcmp(s1,s2);
val = strncmp(s1,s2,n);
```

Arguments:

s1	pointeur vers la première chaîne
s2	pointeur vers la seconde chaîne
n	le nombre maxi. de caractères à comparer

Retour:

val	le résultat de la comparaison
inférieur à zéro	=> s1 < s2
égal à zéro	=> s1 = s2
supérieur à zéro	=> s1 > s2

strcpy, strncpy

Fonctions de copie de chaînes.

La fonction 'strcpy' copie une chaîne source vers une chaîne destination. La fonction 'strncpy' copie n caractères de la chaîne source vers la chaîne destination. Les chaînes passées en arguments doivent être terminées par le caractère nul.

Séquence d'appel:

```
BYTE *s1, *s2, *ret;
BYTE *strcpy(), *strncpy();
WORD n;
```

```
ret = strcpy(s1,s2);
ret = strncpy(s1,s2,n);
```

Arguments:

```
s1    pointeur de la chaîne destination
s2    pointeur de la chaîne source
n      le nombre maximum de caractères à copier
```

Retour:

```
ret    pointeur vers 's1' contenant le résultat
```

Note:

Dans 'strncpy', si la valeur 'n' excède la longueur de la chaîne 's2', de graves problèmes peuvent survenir.

strlen

Retourne la longueur d'une chaîne terminée par zéro.

Séquence d'appel:

```
BYTE *s;
WORD len;

len = strlen(s);
```

Argument:

```
s      pointeur de la chaîne
```

Retour:

```
len     longueur de la chaîne
```

swab

La fonction 'swab' copie une zone de mémoire dans une autre avec inversion des octets faibles et forts. Cette fonction peut être utilisée pour copier des données binaires depuis un VAX ou PDP-11 vers un ordinateur à base de 68000. Le nombre d'octets à transférer doit être pair.

Séquence d'appel:

```
WORD ret;
BYTE *from, *to;
WORD nbytes;

ret = swab(from, to, nbytes);
```

Arguments:

from	adresse de la source
to	adresse de la destination
nbytes	le nombre d'octets à copier

Retour:

ret	toujours zéro
-----	---------------

tan, atan

La fonction 'tan' retourne la tangente d'un nombre flottant, la fonction 'atan' retourne l'arc tangente d'un nombre flottant. Les angles doivent être exprimés en radians.

Séquence d'appel:

```
FLOAT tan(), atan();
FLOAT val, rval, ret;

ret = tan(rval);
ret = atan(val);
```

Arguments:

rval	un flottant exprimant un angle en radians
val	un flottant

Retour:

ret	la tangente ou arc tangente exprimée en radians
-----	---

Note:

La précision maximum est obtenue avec des nombres inférieurs à 2 PI.

N.d.T.:

La fonction 'tan' n'est pas reconnue dans l'actuelle version de la bibliothèque 'GEMLIB'.

ttyname

La fonction 'ttyname' retourne un pointeur vers la chaîne de caractères "CON:" si l'identificateur de fichier envoyé comme paramètre correspond à la console.

Séquence d'appel:

```
BYTE *name, *ttyname();
WORD fd;
```

```
name = ttyname(fd);
```

Argument:

fd l'identificateur d'un fichier ouvert

Retour:

name un pointeur vers la chaîne "CON:"
zéro si l'identificateur n'est pas associé à la console

ungetc

La fonction 'ungetc' renvoie un caractère dans le tampon de lecture d'un fichier. Le prochain appel des fonctions 'getc', 'getw' ou 'getchar' restituera le caractère.

Cette fonction n'a d'effet que si une lecture (et donc une création de tampon) s'est déjà produite dans le fichier considéré.

Séquence d'appel:

```
BYTE c;
FILE *stream;
WORD ret;
```

```
ret = ungetc(c, stream);
```

Arguments:

c le caractère à renvoyer dans le tampon
stream pointeur de structure FILE du fichier concerné

Retour:

ret le caractère correctement renvoyé
 -1, si erreur

Note:

La fonction 'fseek' efface tout caractère renvoyé.
Il est interdit de renvoyer le caractère de fin de fichier.

unlink

La fonction 'unlink' supprime un nom de fichier dans le catalogue du disque. Cette fonction renvoie une erreur si le fichier à supprimer est soit ouvert, soit inexistant.

Séquence d'appel:

```
WORD ret;
BYTE *name;

ret = unlink(name);
```

Argument:

name le nom du fichier (chaîne terminée par zéro)

Retour:

ret si zéro, opération réussie
 si -1, échec

write

La fonction 'write' écrit des données sur un fichier reconnu par son identificateur (cf 'open'). Le transfert s'effectue à la position du pointeur d'octet courant, position définie par une écriture antérieure ou par la fonction 'lseek'.

A l'appel de la fonction, le nombre d'octets à écrire est spécifié; au retour le nombre d'octets effectivement écrits est renvoyé. S'il existe une différence entre ces deux nombres, une erreur s'est produite.

Séquence d'appel:

```
WORD fd;
BYTE *buffer;
WORD bytes, ret;

ret = write(fd,buffer,bytes);
```

Arguments:

fd l'identificateur du fichier ouvert
buffer pointeur vers les données à écrire
bytes le nombre d'octets à écrire

Retour:

ret le nombre d'octets effectivement écrits
 -1 si erreur

Note:

L'écriture dans le fichier spécifié n'est certaine qu'une fois celui-ci fermé.

2. La bibliothèque des fonctions VDI: VDIBIND

La bibliothèque 'VDIBIND' contient la totalité des fonctions VDI. Une application tournant sous l'environnement GEM utilise obligatoirement cette bibliothèque, ne serait-ce que pour ouvrir une station virtuelle de travail (fonction 'v_opnvwk()').

La ligne de commande de l'édition de liens doit débiter avec le lanceur 'APSTART' si le programme est une application n'utilisant pas la bibliothèque 'GEMLIB'; avec le lanceur 'GEMSTART' dans le cas contraire.

Si le programme à "lier" est un accessoire, 'ACCSTART' devra être le premier fichier de la ligne de commande de l'éditeur de liens.

NOTE: Le fichier 'VDIBIND.H' contient toutes les déclarations de la bibliothèque VDI.

N.D.T.: L'ouvrage 'Au coeur de l'ATARI ST' fournit une description détaillée de toutes les fonctions VDI.

3. La bibliothèque des fonctions AES: AESBIND

La bibliothèque 'AESBIND' contient la totalité des fonctions AES. Sous environnement GEM, celles-ci sont chargées de la gestion des applications (initialisation, etc...), des événements (souris, clavier, etc...), des menus déroulants, des objets GEM, de certaines fonctions graphiques, des fenêtres, des ressources, des communications entre applications, du sélecteur de fichier ainsi que des fonctions "presse-papiers".

L'utilisation de cette bibliothèque est intimement liée à la bibliothèque 'VDIBIND' décrite ci-avant. La librairie des fonctions VDI doit donc également être présente lorsqu'une édition de liens fait appel à 'AESBIND'.

La ligne de commande de l'édition de liens doit débiter avec le lanceur 'APSTART' si le programme est une application n'utilisant pas la bibliothèque 'GEMLIB'; avec le lanceur 'GEMSTART' dans le cas contraire.

Si le programme à "lier" est un accessoire, 'ACCSTART' devra être le premier fichier de la ligne de commande de l'éditeur de liens.

N.D.T.: L'ouvrage 'Au coeur de l'ATARI ST' fournit une description détaillée de toutes les fonctions AES.

4. La bibliothèque système: OSBIND

La bibliothèque 'OSBIND' résout les appels aux fonctions GEMDOS, BIOS et XBIOS, c'est à dire les appels au TRAP 1, TRAP 13 et TRAP 14.

Le fichier 'OSBIND.H' est généralement inclus dans le fichier source afin de faire appel aux fonctions système par leurs noms génériques plutôt que par leurs numéros de fonction: dans un listing, 'Cconin()' est plus compréhensible que 'gemdos(0x1)'.

Les manuels de référence du GEMDOS et du BIOS fournissent la description détaillée de ces fonctions système.

5. Conseils de programmation en langage C

Pour rendre votre programme C portable, lisible, et facilement corrigible, essayez de suivre dans la mesure du possible les principes édictés dans cette section. Malgré tout, aucune règle ne peut évaluer chaque situation et c'est en définitive votre jugement personnel qui doit trancher lorsque des cas particuliers se présentent à vous.

5.1. La "modularité"

Découper une application en plusieurs modules sources distincts réduit les coûts de maintenance et de portabilité. Chaque module doit regrouper les fonctions travaillant à une tâche spécifique. Cette pratique a deux intérêts:

- * le programmeur de maintenance peut traiter, pour ses modifications, la plupart des modules comme des "boîtes noires";
- * la nature des structures de données est transparente au reste du programme. Dans une application modulaire, la majorité des structures de données peuvent être modifiées en corrigeant un unique module.

5.1.1. LA TAILLE DES MODULES

La taille maximale conseillée pour un module est de 500 lignes de source. Ne créez pas un module plus grand que n'en nécessite l'exécution d'une tâche particulière.

5.1.2. COMMUNICATIONS ENTRE MODULES

Dans la mesure du possible, les modules communiquent entre eux par l'appel de fonction avec passage d'arguments. Evitez l'emploi de variables globales. Si plusieurs modules emploient les mêmes structures de données, utilisez des fichiers d'en-tête (fichiers .H).

5.1.3. LES FICHIERS D'EN-TETE

Dans une programmation modulaire, utilisez les fichiers d'en-tête pour définir les types, les constantes symboliques et les structures, communs à tous les modules. La liste qui suit donne les règles à appliquer pour l'utilisation des fichiers '.H':

- * Utilisez le format '#include "fichier.h"' pour les fichiers d'en-tête spécifiques à l'application.
- * Incluez les fichiers '.H' du système en utilisant les signes '<' et '>': '#include <stdio.h>'.
- * N'emboîtez pas des fichiers d'en-tête.
- * Seules les variables globales doivent être définies dans un fichier '.H'. Ces variables ne doivent en aucun cas y être initialisées.

- * A l'écriture de macro-instructions, utilisez les parenthèses pour indiquer la précedence des opérateurs.

5.2. Conventions de code

Pour assurer la portabilité de vos programmes, vous devez suivre strictement les conventions présentées dans ces paragraphes. Sinon, peuvent survenir les problèmes suivants:

- * La longueur différente des variables de type "int" d'une machine à l'autre peut engendrer des incohérences dans la portabilité.
- * L'ordre des octets (la place de l'octet faible et de l'octet fort) variable d'une machine à l'autre peut provoquer des problèmes à la lecture des fichiers binaires.
- * Le nombre de caractères significatifs d'un identificateur peut varier d'un compilateur à l'autre. Certains compilateurs ne font même pas la distinction entre minuscules et majuscules.
- * Certains compilateurs transforment les variables de type CHAR ou SHORT en entiers signés lors d'opérations arithmétiques; certains autres ne le font pas.
- * Certains compilateurs considèrent une constante hexadécimale ou octale comme un entier non signé; d'autres n'agissent pas de même. Ainsi, l'instruction suivante peut donner des résultats différents:

```
LONG data;
```

```
.
```

```
printf("%ld\n", (data & 0xffff));
```

Au lieu de masquer le mot fort de la variable "data", des compilateurs peuvent étendre la constante et donc afficher le résultat non masqué.

- * Une grande prudence est requise lors de tests complexes, particulièrement ceux donnant un résultat booléen. Une erreur dans les parenthèses peut conduire à un résultat inattendu.

5.2.1. NOMS DE VARIABLES ET DE CONSTANTES

Les sept premiers caractères des variables locales sont significatifs sur le compilateur ALCYON (huit caractères sur certains compilateurs) et doivent donc être uniques. Sept caractères significatifs également pour les variables globales et les noms de fonctions (six caractères sur certains compilateurs). Les noms de variables et de fonctions sont habituellement en lettres minuscules.

La convention veut également que les constantes définies par '#define' ou les types définis par 'typedef' soient en lettres majuscules. Les macros par contre (ex: getchar, max, min...) sont en minuscules.

Pour éviter toute confusion, il n'est pas conseillé de définir une variable locale en utilisant le même nom qu'une variable globale.

5.2.2. TYPES DE VARIABLES

Compte tenu des différences entre compilateurs, utiliser les types et classes standards de variables peut s'avérer néfaste pour la portabilité des programmes. Mieux vaut définir les types et les classes d'allocation avec les directives "typedef" ou "#define". Les tableaux suivants donnent les types et les classes des variables en langage C.

Définition des types

TYPE	TYPE DE BASE EN C
LONG	long mot signé (32 bits)
WORD	un short signé (16 bits)
UWORD	un short non-signé (16 bits)
BOOLEAN	un short (16 bits)
BYTE	un char signé (8 bits)
UBYTE	un char non-signé (8 bits)
VOID	void (retour de fonction)
DEFAULT	int (16/32 bits)

Définition des classes

CLASSE	CLASSE DE BASE EN C
REG	variable registre
LOCAL	variable automatique
MLOCAL	variable statique
GLOBAL	variable globale
EXTERN	variable externe

Vous devez déclarer les variables globales au début du module source et en dehors de toute fonction. Les variables locales (ou automatiques) se définissent dans le corps de la fonction qui les utilise. Il est conseillé de toujours spécifier le type et la classe de chaque variable, même dans les cas où un compilateur ne l'exige pas.

5.2.3. EXPRESSIONS ET CONSTANTES

Ecrivez les expressions et les constantes de façon à être indépendant du compilateur utilisé. N'hésitez pas à utiliser les parenthèses pour lever les ambiguïtés. Ainsi l'expression:

```
if(c = getchar() == '\n')
```

ne transmet pas à la variable 'c' la valeur retournée par 'getchar'. Cette dernière au contraire est comparée à la constante '\n' et la variable 'c' reçoit le résultat de la comparaison (0 ou 1). La valeur retournée par 'getchar' est perdue. Placer des parenthèses pour spécifier les priorités résoud le problème:

```
if((c = getchar()) == '\n')
```

Ecrivez les constantes à utilisation de masque de façon à éviter la taille entière par défaut. Dans l'exemple précédemment évoqué,

```
LONG data;
```

```
printf("%ld\n", (data & 0xffffL));
```

la spécification longue de la constante résoud le problème sur tous les compilateurs.

Pour la portabilité, les constantes-caractère doivent être constituées d'un unique caractère. Sinon l'utilisation d'une constante-chaine s'impose.

Les virgules qui séparent les arguments dans une fonction ne sont pas des opérateurs. En conséquence, l'ordre d'évaluation n'est pas garanti. Par exemple, l'appel de la fonction suivante

```
printf("%d %d\n", i++, i++);
```

peut donner des résultats différents selon les compilateurs.

5.2.4. ARITHMETIQUE DES POINTEURS

Ne manipulez pas les pointeurs comme des entiers ou autres variables arithmétiques. Si le langage C permet l'addition ou la soustraction d'un pointeur et d'un entier, ne tentez pas les multiplications, divisions ou opérations logiques (AND, OR,...) sur des pointeurs.

Un pointeur vers un certain type d'objet peut être converti en un pointeur d'objets de tailles plus restreintes (par exemple, un pointeur d'objets "LONG" peut pointer des objets "BYTE"). L'inverse peut, par contre, poser des problèmes d'alignement.

Il est possible de tester l'égalité d'un pointeur avec un autre pointeur ou avec une constante (particulièrement la constante NULL). Cependant, les comparaisons arithmétiques du type ">=", non seulement ne sont pas admises par tous les compilateurs, mais peuvent encore conduire à des résultats aberrants.

N'évaluez pas mentalement la taille d'une structure de données: le compilateur a pu se ménager des emplacements libres afin de permettre des alignements. Utilisez plutôt l'opérateur "sizeof".

5.2.5. LES CONSTANTES CHAINES

Assurez vous que les chaînes de caractères de votre application puissent facilement être traduites en une autre langue. Une méthode judicieuse consiste à constituer une zone de pointeurs vers les constantes chaîne, zone initialisée dans un fichier séparé. De cette manière, chaque chaîne est référencée par rapport à son pointeur.

Ne modifiez jamais une chaîne de caractères de la manière suivante:

```
BYTE string[ ] = "BDOS ERROR ON x:";
```

```
string[14] = 'A';
```

Après traduction, les chaînes ont toutes les chances d'avoir une taille différente, ce qui vous obligera à modifier l'instruction.

Ne vous servez jamais du bit de poids fort d'un caractère ASCII (comme drapeau par exemple). Le jeu étendu de caractères utilise les valeurs supérieures à 0x7F.

5.2.6. ZONE 'DATA' ET ZONE 'BSS'

Habituellement, un programme écrit en C se divise en trois sections:

- * la zone "TEXTE" (contenant les instructions du programme)
- * la zone "DATA" (contenant les données initialisées)
- * la zone "BSS" (contenant les données non-initialisées)

Evitez de modifier les données initialisées d'un programme. En agissant ainsi, vous améliorez les performances de transfert et diminuez l'obligation des lectures du disque lorsque votre programme tourne sur un système multi-utilisateurs.

Vous vous laissez également la possibilité de placer votre application en mémoire morte sans modification.

5.2.7 PRESENTATION D'UN FICHIER SOURCE

La liste suivante définit la structure que devrait adopter chaque module d'un programme afin d'en assurer une parfaite lisibilité:

- * au début du fichier, un commentaire décrivant:
 - la tâche du module
 - les points d'entrée dans le module
 - les variables externes utilisées
 - la dépendance à la machine ou au compilateur
- * les fichiers à inclure
- * les "#define" propres au module
- * les variables globales et leurs commentaires
- * la liste des fonctions avec, pour chacune d'elle:
 - un commentaire décrivant la tâche de la fonction, les paramètres reçus et retournés ainsi que d'éventuelles précisions aidant à la compréhension.
 - le titre de la fonction avec, explicitement, le type du paramètre retourné. S'il est inexistant, précisez le par le type 'VOID'.
 - la définition des arguments avec, explicitement, leurs types et classes d'allocation.
 - la définition des variables locales avec, là encore, leurs types et classes d'allocation.
 - les instructions de la fonction.

CHAPITRE III

LES OUTILS DE COMPILATION

1. INTRODUCTION /
2. LE PRE-PROCESSEUR CP68 /
3. L'ANALYSEUR SYNTAXIQUE C068 /
4. LE GENERATEUR DE CODE C168 /
5. CONSEILS DE COMPILATION /

1. Introduction

La compilation d'un programme C requiert trois passes, à savoir:

- * la passe du pré-processeur produisant, à partir d'un fichier source C, un fichier à suffixe '.I',
- * la passe d'analyse syntaxique produisant, à partir d'un fichier '.I', deux fichiers à suffixe '.1' et '.2',
- * la passe de génération de codes produisant, à partir des deux fichiers '.1' et '.2', un fichier source assembleur au suffixe '.S'.

Ces trois phases achevées, il convient encore de lancer la phase d'assemblage (Cf. AS68 au chapitre 4), la phase d'édition de "liens" (Cf. LINK68 et L068 au chapitre 5) et enfin la phase de conversion au format GEMDOS (Cf. l'utilitaire RELMOD au chapitre 5, à 4).

Si aucune erreur n'a été décelée durant ces différentes étapes et si aucune "bogue" ne s'est glissée dans votre source, vous obtenez un fichier exécutable à partir du Bureau GEM ou du programme de command 'COMMAND.TOS'.

Ce chapitre décrit les trois premières étapes de la compilation et fournit des méthodes pour résoudre les problèmes d'espace disque et accélérer le processus de compilation.

2. Le pré-processeur (CP68)

Le pré-processeur produit, à partir du fichier source, un fichier intermédiaire dans lequel tous les noms et constantes symboliques ont été convertis. Sa tâche consiste donc en la lecture des fichiers inclus (`#include`) et en la résolution des `#define`. Une ligne de commande invoquant le pré-processeur obéit au format suivant:

```
'a:cp68 [-I d:] fichier.C fichier.I [>chemin et nom de sortie]
```

-I

Indique que le prochain argument désigne le disque et le répertoire où seront lus les fichiers inclus, c'est à dire les fichiers à suffixe '.H'.

NOTE: Le chemin peut également être signifié derrière l'instruction `#include` (exemple: `#include "b:fichier.h"`), mais cette méthode figeant le chemin d'accès n'est pas recommandable.

fichier.C

Correspond au fichier source à compiler.

fichier.I

Représente le fichier produit par CP68.

>chemin et nom du fichier de sortie

Par défaut, l'affichage des messages d'erreur se fait sur l'écran. Si cette option est spécifiée, ceux-ci sont orientés vers le fichier disque indiqué.

N.D.T.: Il est également possible d'ajouter les sorties à un fichier déjà existant en utilisant l'option '>>nom du fichier'. Rappelons d'autre part qu'en lançant le programme à partir de 'COMMAND.TOS', le périphérique de sortie peut être modifié. Ainsi: 'a:' ligne de commande >PRN: oriente les sorties vers l'imprimante.

Les messages d'erreur générés par CP68 sont décrits en annexe A.

3. L'analyseur syntaxique (C068)

L'analyseur syntaxique effectue la phase principale de la compilation en effectuant un pré-codage aboutissant à la création d'un arbre syntaxique. C'est lui qui détecte la plus grande partie des erreurs de syntaxe ou de sens du fichier source. L'appel de C068 a le format suivant:

```
'a:c068 fichier.I fichier.1 fichier.2 fichier.3 [-e|-f] [-w][-T]
```

fichier.I

Le fichier d'entrée, produit par CP68.

fichier.1 fichier.2

Les deux fichiers créés par C068.

fichier.3

Un fichier de travail collectant les constantes. Il est ensuite inclus dans 'fichier.2' et effacé par C068.

-e ou -f, -w, -T

N.D.T.(1): Aucune documentation n'est fournie sur la signification de ses différentes options. L'usage veut que la ligne de commande de C068 se présente ainsi:

```
c068 fichier.I fichier.1 fichier.2 fichier.3 -f
```

N.D.T.(2): Les messages d'erreur de C068 sont dirigés vers l'écran. Si l'appel se fait de 'COMMAND.TOS', il est cependant possible de les orienter vers un fichier ou un périphérique.

Les messages d'erreur générés par C068 sont décrits en annexe A.

4. Le générateur de code (C168)

Le générateur de code, à partir des fichiers intermédiaires générés par C068, produit un fichier source en langage assembleur.

```
*a:c168 fichier.1 fichier.2 fichier.S [-L][-D]
```

```
fichier.1 fichier.2
```

Les deux fichiers produits par C068 et qui représentent les fichiers d'entrées de C168.

```
fichier.S
```

Source assembleur créé par C168, produit final de la compilation proprement dite.

-L

Si ce drapeau est spécifié, indique à C168 que les adresses doivent être codées sur 32 bits. Par défaut, elles sont codées sur 16 bits.

-D

Si ce drapeau est spécifié, oblige C168 à inclure les numéros de ligne du source C (comme des commentaires) dans le fichier '.S'. Cette option est très utile pour le débogage puisque l'assembleur AS68 annonce ses messages d'erreur en précisant le numéro de ligne par rapport au fichier.S.

N.D.T.: Les messages d'erreur de C168 sont dirigés vers l'écran. Si l'appel se fait de 'COMMAND.TOS', il est cependant possible de les orienter vers un fichier ou un périphérique.

Les messages d'erreur générés par C168 sont décrits en annexe A.

5. Conseils de compilation

Une compilation est une phase suffisamment longue pour que les dispositions nécessaires soient prises afin d'éviter de la renouveler trop souvent et tenter de l'accélérer.

5.1. Avant la compilation

Évitez les fichiers sources trop importants. Si la taille excède 500 lignes, travaillez sur plusieurs modules.

Relisez attentivement votre source en axant la lecture sur les erreurs de syntaxe. Vérifiez particulièrement l'absence d'un point-virgule derrière les titres de fonctions, leur présence derrière les instructions. Vérifiez que le nombre d'accolades ouvrantes est égal au nombre d'accolades fermantes, que le nombre

de parenthèses ouvrantes est égal au nombre de parenthèses fermantes.

5.2. Pendant la compilation

Tout dépend bien entendu de la configuration dont vous disposez.

Avec un DISQUE DUR, vous disposez de suffisamment d'espace pour placer tous les outils de développement sur une même partition. Les fichiers intermédiaires seront stockés également sur ce support.

Avec UN LECTEUR, un disque virtuel est recommandé afin d'accélérer la phase de compilation. Placez sur votre disque physique les fichiers '.H', le pré-processeur CP68, l'analyseur C068, le générateur C168, l'assembleur AS68 et son fichier d'initialisation AS68SYMB.DAT, l'utilitaire BATCH.TTP et/ou COMMAND.TOS, l'utilitaire WAIT.PRG. Créez un disque virtuel (C: par exemple) en laissant au moins 200 Ko pour les applications et placez-y votre programme source (ESSAI.C par exemple). Le fichier 'CRAM.BAT' placé sur votre disque physique pourrait être le suivant:

```
cp68 %1.c %1.i
c068 %1.i %1.1 %1.2 %1.3 -f
rm %1.i
c168 %1.1 %1.2 %1.s
rm %1.1
rm %1.2
as68 -f c: -l -u %1.s
rm %1.s
wait.prg
```

L'appel par BATCH.TTP ou par COMMAND.TOS (Cf. Chapitre 7) est identique et se fait sous la forme:
cram c:essai

Tous les fichiers créés sont ainsi stockés sur le disque RAM et ceci entraîne un gain de temps appréciable. Le fichier ESSAI.O produit doit être immédiatement sauvegardé sur disque physique.

Avec DEUX LECTEURS, la méthode précédente est encore recommandable. Mais vous pouvez également utiliser le disque B comme disque de travail. Pour ce faire, placez-y votre source et modifiez la ligne de commande en conséquence. La commande d'assemblage doit également être modifiée: as68 -f b: -l -u %1.s

NOTE: Avec un tel fichier '.BAT', les messages d'erreur sont affichés à l'écran (Cf. les modes d'emploi des différents utilitaires pour une redirection sur fichier). Après l'exécution de CP68, nous vous conseillons fortement de "geler" l'écran avec CONTROL-S afin de ne pas perdre le premier message d'erreurs éventuelles de C068 (Cf. Annexe A, les messages d'erreur de C068).

CHAPITRE IVL'OUTIL D'ASSEMBLAGE: AS68

1. OPERATION D'ASSEMBLAGE
2. INITIALISATION D'AS68
3. LANCEMENT DE L'ASSEMBLAGE
4. EXEMPLES DE COMMANDES
5. DIRECTIVES DU LANGAGE ASSEMBLEUR
6. DIFFERENCES ENTRE LANGAGES ASSEMBLEUR
7. EXTENSIONS AU LANGAGE
8. MESSAGES D'ERREUR (Cf. ANNEXE A)
9. SOMMAIRE DU JEU D'INSTRUCTIONS

1. Opération d'assemblage

L'assembleur GEMDOS AS68 transforme un code source écrit en langage assembleur en un programme exécutable sur microprocesseur 68000. Il produit un fichier objet relogeable et, sur option, un listing. Vous trouverez le sommaire du jeu d'instructions reconnues par AS68 à la fin de ce chapitre. Les exceptions et les ajouts au jeu d'instructions standard de MOTOROLA sont explicités dans les sections 1.6 et 1.7.

2. Initialisation de AS68

Si le fichier AS68SYMB.DAT n'est pas présent sur votre disque, il vous faut créer ce fichier d'initialisation avant toute utilisation de l'assembleur. Pour ce faire, lancez du COMMAND.PRGM la commande suivante:

```
{a} AS68 -I AS68INIT
```

AS68 crée ainsi le fichier AS68SYMB.DAT nécessaire à toute phase d'assemblage. Après création de ce fichier, il ne sera plus utile de le créer de nouveau sauf si vous désirez modifier l'emplacement de la zone TPA (zone des programmes).

3. Lancement de l'assemblage

Une commande d'appel à l'assembleur a le format suivant (les arguments entre crochets sont facultatifs):

```
{a}as68 [-F chemin] [-P] [-S chemin] [-U] [-L ou -A] [-N] [-I]
chemin et nom du fichier source [>chemin et nom du listing]
```

Ces différentes options ont les significations suivantes:

-F chemin

Spécifie le disque et le répertoire dans lequel seront stockés les fichiers intermédiaires. Si cette option n'est pas spécifiée, ce sont les disques et répertoires courants qui sont utilisés.

-I

Initialisation de l'assembleur (voir à 1.2.)

-P

Si présent, AS68 produit et imprime un listing sur le périphérique de sortie standard (par défaut, l'écran). Il est possible d'orienter la sortie du listing (messages d'erreur compris) vers le disque en utilisant l'option [>chemin et nom du

listing]. Notez que même si l'option -P n'est pas spécifiée, les messages d'erreur sont toujours affichés.

-S chemin

Indique à l'assembleur le nom du disque et du répertoire contenant le fichier d'initialisation AS68SYMB.DAT (voir section 1.2.). Si cette option n'est pas spécifiée, AS68 lit ce fichier sur le disque et répertoire courants.

-U

Cette option a pour effet de considérer tout symbole indéfini dans le programme source courant comme une référence externe. L'éditeur de liens se charge alors de les "reconnaître" dans d'autres fichiers à suffixe '.o' ou dans une bibliothèque (voir également la directive '.globl').

-L ou -A

L'option -L (option par défaut) force AS68 à considérer les adresses de constantes comme des longs mots.

L'option -A, travaillant au contraire sur des mots, ne peut s'utiliser que pour les programmes s'exécutant sur les premiers 64 Ko de mémoire. (Option sans intérêt sur le ST).

-N

Désactive l'optimisation par branchements relatifs. Normalement, AS68 essaie d'utiliser, chaque fois que possible, les instructions BRA (branchement) et BSR (branchement à sous-routine) plutôt que JSR. Cela permet d'accélérer la vitesse d'exécution du programme et de réduire le code produit.

-T

Autorise AS68 à accepter et à traiter les instructions supplémentaires du microprocesseur 68010.

chemin et nom du fichier source.S

Spécifie le fichier à assembler: ce paramètre, nécessairement terminé par le suffixe '.S', est obligatoire.

>chemin et nom du fichier listing

Écrit sur le fichier disque spécifié:

- * le listing ainsi que les messages d'erreur si l'option -P est précisée;
- * uniquement les messages d'erreur dans le cas contraire.

N.D.T.: Il est également possible d'ajouter les sorties à un fichier déjà existant en utilisant l'option '>>nom du fichier'. Rappelons d'autre part qu'en lançant le programme à partir de

'COMMAND.TOS', le périphérique de sortie peut être modifié. Ainsi:
{a}'ligne de commande' >PRN:
oriente les sorties vers l'imprimante.

4. Exemples de commandes

{a} as68 -u -l test.s

Cette commande assemble le fichier source 'test.s' et produit le fichier objet 'test.o'. Les messages d'erreur seront affichés sur l'écran et tout symbole indéfini sera traité comme une référence globale.

{a} as68 -f c: -p -s b: -l test.s > test.l

Cette commande assemble le fichier source 'test.s' placé sur le disque courant pour produire le fichier objet 'test.o'. Listing et messages d'erreur seront dirigés vers le fichier 'test.l'. Les fichiers intermédiaires de l'assemblage seront écrits sur le disque C tandis que le fichier d'initialisation AS68SYMB.DAT sera lu sur le disque B.

5. Directives du langage assembleur

Voici la liste exhaustive de toutes les directives d'assemblage reconnues par AS68:

.comm label,expression

La directive 'comm' spécifie un label et la taille d'une zone commune que des programmes assemblés séparément peuvent se partager. L'éditeur de liens donne la même adresse à chaque zone ayant le même label. Si la taille de la zone commune diffère d'un programme source à l'autre, c'est le plus grand espace réservé qui sera retenu.

.data

Cette directive indique à l'assembleur que les données qui suivent doivent être stockées dans la zone des données initialisées (data segment).

.bss

Cette directive spécifie que les symboles qui suivent ont une place réservée dans la zone BSS (Block Storage Segment), c'est à dire la zone des données non-initialisées. Il est impossible, après '.bss', d'entrer des instructions ou des données initialisées. Seule la directive '.ds' (voir plus loin) est utilisable.

.dc opérande [,opérande,...]

La directive '.dc' (define constant) définit en mémoire une ou plusieurs constantes. Les opérands peuvent être des symboles, des expressions numériques, des valeurs numériques explicites (en décimal ou hexadécimal) ou des chaînes de caractères ASCII. Les différents opérands sont séparés par une virgule.

Une constante chaîne est délimitée par des apostrophes ou des guillemets, chaque caractère occupant un octet.

Vous pouvez spécifier la longueur de chaque constante en ajoutant derrière le séparateur '.' une lettre indiquant une constante octet (.b), mot (.w) ou long mot (.l).

.dc.b opérande [,opérande,...]

Le ou les opérande(s) sont des constantes ayant la taille d'un octet. Si un nombre impair d'opérands suit cette directive, AS68 complète l'octet impair de droite par la valeur nulle, à moins que la directive suivante soit également 'dc.b'. Si tel est le cas, l'octet impair est occupé par la constante suivante.

.dc.w opérande [,opérande,...]

Le ou les opérands sont des constantes ayant la taille d'un mot. Si vous spécifiez un nombre impair d'octets (par exemple avec une constante chaîne), l'octet le plus à droite sera forcé à zéro afin de retrouver une adresse paire. AS68 aligne toujours les constantes mots à des adresses paires.

.dc.l opérande [,opérande,...]

Le ou les opérands sont des constantes ayant la taille d'un long mot. Si la ou les constantes occupent un espace qui n'est pas un multiple de 4 octets, les emplacements vides du dernier long mot seront forcés à zéro. AS68 aligne toujours les constantes longs mots à des adresses paires.

.ds opérande

La directive '.ds' (define storage) réserve une allocation mémoire, sans initialiser un contenu. L'opérande précise le nombre d'octets, de mots ou de longs mots qu'il y a lieu de réserver. La spécification de taille est donnée par la lettre qui suit '.ds':

.ds.b	X	réserve X octets
.ds.w	X	réserve X mots
.ds.l	X	réserve X longs mots

.end

Obligatoire à la fin de code source, termine l'assemblage en cours. Derrière cette directive, un unique code de Retour Chariot est autorisé (ni <espace>, ni commentaire).

.endc

Fin d'assemblage conditionnel. Voir les directives 'if condition'.

.equ (ou =) expression

Assigne la valeur de l'expression au symbole qui précède la directive. La syntaxe est la suivante:

```
label .equ expression    ou
label = expression
```

Le label doit être unique et ne peut être défini ailleurs que dans le programme source. L'expression ne peut contenir un symbole indéfini ou non encore défini.

.even

Incrémente le compteur d'allocation si nécessaire pour forcer une adresse paire. L'instruction ou la donnée qui suit cette directive sera alignée sur une adresse paire.

```
.globl label [,label,...]
.xdef label [,label,...]
.xref label [,label,...]
```

Ces directives sont équivalentes et indiquent à l'AS68 que les labels désignés sont des références externes. Si ces labels sont définis à l'intérieur du programme source courant, ils deviennent disponibles pour d'autres sources au moment de l'édition de liens. S'ils ne sont pas définis dans le programme source courant, ils seront recherchés, lors de l'édition de liens, dans d'autres sources.

Rappelons ici que l'option -U lors de l'appel d'AS68 transforme tout label indéfini en une référence externe.

```
.ifeq expression    (si égal à zéro)
.ifne expression    (si différent de zéro)
.ifle expression    (si plus petit ou égal à zéro)
.iflt expression    (si plus petit que zéro)
.ifge expression    (si plus grand ou égal à zéro)
.ifgt expression    (si plus grand que zéro)
```

Directives d'assemblage conditionnel qui testent l'expression en fonction de la directive choisie. Si le résultat est vrai, les instructions qui suivent sont assemblées. Sinon AS68 recherche la directive de fin de condition '.endc' en ignorant tout ce qui la précède.

```
.ifc 'chaîne1','chaîne2'
.ifnc 'chaîne1','chaîne2'
```

Directives d'assemblage conditionnel.

Avec '.ifc', la condition est vraie si les deux chaînes sont identiques.

Avec '.ifnc', la condition est vraie si les deux chaînes sont différentes.

.offset expression

Permet de créer une zone de stockage propre au programme source en cours. Cette zone débute à l'adresse spécifiée par 'expression'. Seules les directives '.ds', '.equ' et '.reg' sont autorisées dans cette zone. Celle-ci est terminée par l'une des directives suivantes: '.bss', '.data', '.end', ou '.text'

.org expression

Fixe l'adresse absolue contenue dans 'expression' à partir de laquelle le programme objet sera logé. L'emploi de cette directive n'autorise plus l'usage des références externes.

.page

Force un saut de page lors de la sortie du listing sur imprimante.

.reg liste des registres

Définie par un label, cette directive crée un masque de registres qui peut être utilisé par l'instruction 'movem'. Un ou plusieurs registres sont placés dans une liste par ordre ascendant en respectant le format suivant:

```
R?[-R?[/R?[-R?...]]...]
```

Les mnémoniques de registres permis sont: A0-A7, D0-D7, R0-R15. Par exemple:

```
truc .reg    D0-D7/A0/A2
```

```
.text
```

```
....      movem.l  truc, sauve  * Sauve les registres *
....                        * D0 à D7, A0 et A2 *
```

```
.bss
sauve .ds.l    10
.end
```

.section f

Directive non disponible sur AS68. Voir section 1.6.

.text

Indique à AS68 que le texte qui suit cette directive correspond à des instructions assembleur à placer dans la zone des programmes (zone TPA). Voir aussi les directives '.data' et '.bss'. En l'absence de cette directive, l'assembleur

suppose que la zone programme commence dès la première ligne du code source.

6. Différences entre langages assembleur

Les différences de syntaxe entre l'assembleur AS68 et l'assembleur MOTOROLA sont décrites ci-dessous.

- * Dans AS68, le point qui précède une directive est optionnel.
 equ est identique à .equ
 ds est identique à .ds

- * AS68 ignore les directives suivantes:
 comline
 mask2
 idnt
 opt

- * La directive '.set' de MOTOROLA est considérée par AS68 de la même façon que la directive '.equ'.

- * AS68 accepte aussi bien les caractères minuscules que majuscules. Les instructions et directives sont reconnues dans les deux cas. Cependant pour les variables et les labels, il n'y a pas d'équivalence. Ainsi 'DEBUT' et 'Debut' ne sont pas identiques.

- * Pour AS68, les labels doivent être terminés par deux points (:), sauf s'il débute en première colonne.

- * Avec AS68, il est possible de délimiter les chaînes de caractères aussi bien par des guillemets que par des apostrophes. Par exemple: "ABCD" ou 'ABCD'

- * Avec AS68, les registres peuvent être référencés avec les mnémoniques suivantes:

r0-r15	ou	R0-R15
d0-d7	ou	D0-D7
a0-a7	ou	A0-A7

Majuscules et minuscules sont équivalentes. Les registres R0-R7 correspondent aux registres D0-D7, les registres R8-R15 correspondent aux registres A0-A7.

- * Avec AS68, les lignes de commentaires peuvent commencer soit par l'astérisque (*), soit par un point-virgule (;). Pensez à laisser un espace après l'un ou l'autre signe pour éviter toute confusion.

- * L'assembleur MOTOROLA permet de loger un programme dans 16 segments distincts. AS68 n'autorise que trois zones: la zone texte (text), la zone des données initialisées (data) et la zone des données non-initialisées (bss).

7. Extensions au langage

Afin de faciliter la programmation en assembleur, des perfectionnements ont été apportés à AS68. En voici la liste:

- * Lorsque les instructions 'add', 'sub', et 'cmp' sont utilisées avec un registre d'adresse en source ou destination, elles sont transformées respectivement en 'adda', 'suba', et 'cmpa'. L'instruction 'clr' avec un registre d'adresse est transformée en 'sub Ax, Ax'.

- * 'add', 'and', 'cmp', 'eor', 'or', 'sub' sont autorisées avec comme premier opérande une donnée immédiate. Elles seront respectivement traduites comme 'addi', 'andi', 'cmpi', 'eori', 'ori', 'subi', si toutefois le mode d'adressage choisi le permet.

- * Toute instruction de branchement génère, si possible, un branchement court.

- * Dans toute instruction de décalage où n'est pas spécifié le nombre de décalages, la valeur 11 est prise par défaut. Ainsi, 'asl R1' est identique à 'asl 11, R1'.

- * L'instruction 'jsr' est transformée en instruction 'bsr' chaque fois que cela provoque un gain de place mémoire.

- * Les instructions 'move', 'add', et 'sub' utilisées avec une donnée immédiate comme premier opérande et un registre de donnée comme destination sont transformées, lorsque c'est possible, en 'moveq', 'addq', et 'subq'.

8. Les messages d'erreur

L'annexe A fournit liste de tous les messages d'erreur générés par AS68.

9. Sommaire du jeu d'instructions

La liste qui suit présente le jeu d'instructions complet du microprocesseur 68000.

add	abcd	addition décimale avec bit d'extension
	add	addition binaire
	adda	addition avec registre d'adresse
	addi	addition immédiate
	addq	addition immédiate rapide
	addx	addition binaire avec bit d'extension
and	and	ET logique
	andi	ET logique immédiat
	andi to CCR	ET logique immédiat avec codes condition

	andi to SR	ET logique immédiat avec registre d'état
	asl	décalage arithmétique à gauche
	asr	décalage arithmétique à droite
	bcc	branchement conditionnel
	bchg	test d'un bit et modification
	bclr	test d'un bit et remise à zéro
	bra	branchement
	bset	test d'un bit et mise à un
	bsr	branchement à un sous-programme
	btst	test d'un bit
cmp	chk	test des limites d'un registre
	clr	remise à zéro d'un opérande
	cmp	comparaison
	cmpa	comparaison avec un registre d'adresse
	cmpi	comparaison immédiate
	cmpm	comparaison mémoire
	dbcc	test condition, décrément, branchement
	divs	division signée
	divu	division non signée
eor	eor	OU exclusif
	eori	OU exclusif immédiat
	eori to CCR	OU exclusif immédiat avec codes condition
	eori to SR	OU exclusif immédiat avec registre d'état
	exg	échange de registres
	ext	extension du signe
	illegal	instruction illégale
	jmp	saut
	jsr	saut vers un sous-programme
	lea	chargement adresse effective
	link	réalisation de "liens"
	lsl	décalage logique à gauche
	lsr	décalage logique à droite
move	move	transfert de donnée
	movea	transfert vers registre d'adresse
	movem	transfert de registres multiples
	movep	transfert de données vers des périphériques
	moveq	transfert immédiat rapide
	move to CCR	transfert vers registre codes condition
	move to SR	transfert vers registre d'état
	move USP	transfert de ou vers pointeur de pile
	move from SR	transfert issu du registre d'état
	muls	multiplication signée
	mulu	multiplication non signée

	nbcd	complémentation décimale avec extension
	neg	négation
	negx	négation avec bit d'extension
	nop	pas d'opération
	not	complément à un
or	or	OU logique inclusif
	ori	OU logique inclusif immédiat
	ori to CCR	OU logique inclusif avec codes condition
	ori to SR	OU logique inclusif avec registre d'état
	pea	empilement d'adresse effective
	reset	remise à zéro des circuits externes
	rol	rotation à gauche sans bit d'extension
	ror	rotation à droite sans bit d'extension
	roxl	rotation à gauche avec bit d'extension
	roxr	rotation à droite avec bit d'extension
	rte	retour d'exception
	rtr	retour et restitution des codes condition
	rts	retour de sous-programme
	sbcd	soustraction décimale avec bit d'extension
	scc	positionnement selon conditions
	stop	chargement du SR et arrêt
sub	sub	soustraction binaire
	suba	soustraction à un registre d'adresse
	subi	soustraction immédiate
	subq	soustraction immédiate rapide
	subx	soustraction avec bit d'extension
	swap	échange interne
	tas	test d'un octet et mise à 1 du bit 7
	trap	exception
	trapv	exception si débordement
	tst	test d'un opérande
	unlk	déconnexion

CHAPITRE VOUTILS D'EDITION DE LIENSCONVERSION DE FORMAT

1. INTRODUCTION
2. L'EDITEUR DE LIENS: LINK68
 - 2.1. APPEL DE LINK68
 - 2.2. OPTIONS DE LA LIGNE DE COMMANDE
 - 2.3. EXEMPLES DE COMMANDES
 - 2.4. REDIRECTION DES MESSAGES D'ERREUR
3. L'EDITEUR DE LIENS: LO68
 - 3.1. APPEL DE LO68
 - 3.2. OPTIONS DE LA LIGNE DE COMMANDE
 - 3.3. EXEMPLES DE COMMANDE
4. L'UTILITAIRE RELMOD

1. Introduction

Deux éditeurs de liens ("linker" en anglais) sont fournis avec le pack de développement. Il s'agit de LINK68 et LO68. Tous deux sont chargés de transformer des fichiers objet en programmes exécutables par le système d'exploitation.

LINK68 et LO68 produisent un fichier exécutable et relogeable au format CPM-68K. Il faut ensuite utiliser l'utilitaire de conversion RELMOD pour transformer le fichier du format CPM-68K (avec suffixe .68K) au format GEMDOS (avec suffixe .PRG). L'utilitaire RELMOD, qui effectue cette tâche, est décrit au paragraphe 4.

2. L'éditeur de liens: LINK68

LINK68 est un éditeur de liens qui combine les différents fichiers objets spécifiés dans la ligne de commande. LINK68 accepte aussi bien les fichiers créés à partir d'un compilateur C ou directement de l'assembleur AS68. Cet éditeur de liens produit un fichier exécutable qui peut être au format "segments contigus" ou "segments non-contigus". Notez que les fichiers aux "segments non-contigus" ne sont pas exécutables dans l'environnement GEM. (Cf. AR68 pour une description du format "segments contigus").

LINK68 résoud toutes les références aux symboles externes et chaîne les différents fichiers objets dans l'ordre spécifié. Le point d'entrée du fichier exécutable, résultat de l'édition de liens, est la première instruction du premier fichier contenu dans la ligne de commande.

NOTE: Même un fichier objet unique, ne contenant aucune référence externe, doit passer "l'épreuve" de l'édition de liens. Un tel fichier est "lié" avec lui-même.

Si, dans AS68, vous avez utilisé la directive '.common' spécifiant ainsi une zone commune partagée par différents modules, LINK68 donne la même adresse (dans la zone BSS) à toute zone définie par le même label.

Si une même variable globale est utilisée dans plusieurs modules, LINK68 utilise la plus grande taille nécessaire pour l'allocation à effectuer.

2.1. Appel de LINK68

Pour lancer LINK68, entrez une commande au format suivant:

```
*a:link68 [ fichier = ] fichier-objet-1 [, fichier-objet-2,....
          ....., fichier-objet-n]
```

Si, lors d'un appel de LINK68, la commande n'obéissait pas à

la règle de syntaxe minimale, l'éditeur de liens afficherait la liste des options possibles et retournerait au programme appelant.

La première spécification de fichier (avant le signe '=') correspond au nom du fichier exécutable à produire. Les noms 'fichier-objet-1' jusqu'à 'fichier-objet-n' correspondent aux différents fichiers objet à "lier". Par exemple, la commande suivante crée le fichier de sortie MATH:

```
*a: link68 math = sin, cos, tan
```

Si le nom du fichier de sortie (qui précède le signe "=") est omis, LINK68 crée un nom à partir du premier fichier objet de la liste en ajoutant le suffixe .68K. La commande suivante crée le fichier SIN.68K:

```
*a: link68 sin, cos, tan
```

2.2. Options de la ligne de commande

A l'appel de LINK68, il est possible de contrôler le processus de l'édition de liens par différentes options indiquées dans la ligne de commande. Il existe deux types d'options:

- * les options globales qui s'appliquent au processus dans sa totalité,
- * les options locales qui ne s'appliquent qu'au fichier auquel elles sont rattachées.

Les options globales sont délimitées par des crochets précédant immédiatement le nom du fichier de sortie; les options locales sont délimitées par des crochets et suivent immédiatement le fichier objet auquel elles sont rattachées.

Les caractères <espace> sont autorisés entre les noms de fichiers afin de faciliter la lisibilité, et plus d'une option est permise entre les crochets si chaque argument est séparé par une virgule. LINK68 vous permet également d'utiliser les options dans des formes abrégées si ces dernières n'ont pas un caractère ambigu. Vous trouverez ci-dessous la liste des options disponibles avec, pour chacune d'elles, sa forme abrégée entre parenthèses.

NOTE: Un programme exécutable sous GEM nécessite le nombre magique \$601A. LINK68 ne fournit ce nombre qu'avec des fichiers relogeables possédant une zone de programme (text), une zone de données initialisées (data) et une zone de données non-initialisées (bss), toutes trois contiguës. De ce fait, pour créer un fichier exécutable sous GEM, n'utilisez pas les options ABSOLUTE, DATABASE[n], BSSBASE[n] ou TEXTBASE[n] dans une commande adressée à LINK68.

LISTE DES OPTIONS DE LINK68

ABSOLUTE (AB)

Génère un programme avec adresses absolues, sans table de relocation. Par défaut, LINK68 génère un programme relogeable. (N'oubliez pas que, pour tourner sous GEMDOS, un fichier DOIT être relogeable).

ALLMODS (AL)

Demande à LINK68 de charger tous les modules de bibliothèque même si ceux-ci ne sont pas nécessaires. Par défaut, seuls les modules utiles sont chargés.

BSSBASE[n] (B[n])

Pour des programmes à segments non-contigus, positionne la zone des données non-initialisées (BSS) à la valeur "n" (en hexadécimal) spécifiée. Par défaut, cette zone est placée à partir du premier mot pair qui suit la zone des données initialisées (data).

COMMAND [chemin et nom de fichier] (C[....])

Indique à LINK68 que le nom de fichier placé derrière cette option contient le reste de la ligne de commande. Tout ce qui pourrait suivre est ignoré. L'imbrication de fichiers de commandes n'est pas autorisé. Cette option est particulièrement utilisée lorsque la ligne de commande dépasse 128 caractères, taille maximale autorisée pour un passage d'arguments.

DATABASE[n] (D[n])

Pour des programmes à segments non-contigus, positionne la zone des données initialisées (data) à la valeur 'n' (en hexadécimal) spécifiée. Par défaut, cette zone est placée à partir du premier mot pair qui suit la zone des programmes (text).

IGNORE (IG)

Indique à LINK68 d'ignorer les débordements des adresses 16 bits.

INCLUDE[nom du module] (IN[....])

Oblige la librairie précédant cette option à charger le module spécifié.

LOCALS (L)

Indique à LINK68 de placer les symboles locaux dans la table des symboles (parfois utile pour le SID!). LINK68 intègre les symboles locaux à la table des symboles à partir de l'endroit où la commande apparaît.

L'option NOLOCALS stoppe cette intégration. Combinez ces deux options pour intégrer les symboles locaux de certains fichiers uniquement. LINK68 ignore toujours les symboles locaux commençant par la lettre "L".

NOLOCALS (NO)

Option par défaut. Voir LOCALS.

SYMBOL (S)

Demande à LINK68 de placer la table des symboles dans le fichier produit. (Très utile pour un débogage avec SID!). Par défaut, LINK68 ne crée pas cette table.

TEMPFILES[d:] (TEM[d:])

Indique à LINK68 d'utiliser le disque 'd:' pour stocker ses fichiers intermédiaires. Par défaut, ces derniers sont stockés sur le disque courant. Cette option, lorsqu'elle est spécifiée, doit être placée dans la ligne de commande avant tout nom de fichier en entrée.

N.d.T.: Placer les fichiers temporaires sur un disque virtuel accélère l'édition de liens d'un facteur 3 par rapport à un disque physique.

TEXTBASE[n] (TEX[n])

Spécifie l'adresse de base de la zone des programmes (text). La valeur 'n' est en hexadécimal. Par défaut, la zone 'text' est placée à l'adresse \$0.

UNDEFINED (U)

Demande à LINK68 d'ignorer les symboles non-résolus: LINK68 affiche sur le périphérique de destination (l'écran par défaut) la liste des symboles indéfinis mais continue le processus d'édition de liens. Par défaut, LINK68 affiche les symboles indéfinis et stoppe le processus.

2.3. Exemples de commande

```
*a:link68 [sym,tem[b:]] foobaz = foomain, foolib
```

Cette commande "lie" les fichiers objets 'foomain' et 'foolib' pour les transformer en un fichier exécutable 'foobaz'. La table des symboles est incluse dans le programme, les fichiers temporaires sont écrits sur le disque "B:"

```
*a:link68 [com[linkit.inp
```

Cette commande indique à LINK68 que le reste de la ligne de commande est à lire dans le fichier "linkit.inp". Notez que les

crochets fermants ne sont pas nécessaires.

Le fichier 'linkit.inp' pourrait contenir la commande suivante:

```
link68 [ab, tex[500], d[2A00], b[3000]]
        screen = scrnsl[L], iolib[AL]
```

Dans ce cas, LINK68 créerait le programme 'screen' à partir des fichiers objet 'scrnsl' et 'iolib'. Le programme serait placé à une adresse absolue, avec la zone de programme débutant à l'adresse \$500, la zone des données initialisées à l'adresse \$2A00, et la zone des données non-initialisées à l'adresse \$3000. Les symboles de 'scrnsl' seraient inclus dans la table des symboles et tous les modules de la librairie 'iolib' seraient intégrés au programme.

2.4. Redirection des messages d'erreur

Habituellement, LINK68 envoie tous les messages d'erreur vers l'écran. Il est cependant possible de rediriger ces messages vers un fichier en utilisant le caractère ">".

Ainsi, la commande:

```
*a:link68 monfic.68K = fich1, fich2 >b:erreur.1
```

stockera tout message sur le disque B:, dans le fichier 'erreur.1'.

N.D.T.: Il est également possible d'ajouter les sorties à un fichier déjà existant en utilisant l'option '>>nom du fichier'. Rappelons d'autre part qu'en lançant le programme à partir de 'COMMAND.TOS', le périphérique de sortie peut être modifié. Ainsi:
**a:'ligne de commande' >PRN:*
oriente les sorties vers l'imprimante.

3. L'éditeur de liens: L068

L'éditeur de liens L068 réunit les fichiers objets assemblés par AS68 pour produire un programme exécutable. Toutes les références aux symboles externes sont résolues. Les différents fichiers objets sont chaînés dans l'ordre spécifié; le point d'entrée du fichier exécutable, résultat de l'édition de liens, est la première instruction du premier fichier contenu dans la ligne de commande.

NOTE 1: Même un fichier objet unique, ne contenant aucune référence externe, doit passer "l'épreuve" de l'édition de liens. Un tel fichier est alors "lié" avec lui-même.

NOTE 2: Un programme exécutable sous GEM nécessite le nombre magique \$601A. L068 ne fournit ce nombre qu'avec des fichiers relogeables possédant une zone de programme (text), une zone de

données initialisées (data) et une zone de données non-initialisées (bss), toutes trois contiguës. De ce fait, pour créer un fichier exécutable sous GEM, n'utilisez pas les options -T, -Z, -B et -S. Par contre, faites appel à L068 avec l'option -R.

3.1. Appel de L068

L'appel consiste en une commande se présentant sous le format suivant:

```
*a:l068 [-F chemin et répertoire] [-R] [-S] [-I] [-Unom]
        [-O nom de fichier] [-X] [-Z adresse][-B adresse] [-B adresse]
        nom du fichier objet 'l' [nom du fichier objet 'n']
        [>nom du fichier de sortie des messages]
```

3.2. Options de la ligne de commande

Les significations des différentes options sont les suivantes:

-F chemin et répertoire

Indique le disque et le répertoire où seront créés les fichiers temporaires.

-R

Préserve la table de relocation afin de rendre relogeable le programme créé par L068.

-S

Dépouille le programme produit de la table des symboles et de la table de relocation.

-I

Indique à L068 de ne pas générer de message en cas de débordement d'adresse: un programme source assemblé par AS68 sans l'option -L, peut, au moment de l'édition de liens, provoquer des messages de débordement d'adresse si ce programme contient des adresses codées sur un long mot.

-Unom

Force l'édition de liens avec un module spécifié de la librairie, même si ce dernier n'est pas nécessaire à la résolution des autres modules.

Normalement, les modules de librairie ne sont intégrés au programme que lorsqu'ils sont nécessaires. Cette option permet d'intégrer "de force" un module, qui pourra lui-même faire appel à des symboles externes contenus dans les autres modules.

-O nom de fichier

Donne un nom au fichier produit par L068 (un espace est obligatoire entre "-O" et le nom du fichier). Si cette option n'est pas spécifiée, le fichier créé sera nommé "C.OUT".

-X

Force l'inclusion de tous les symboles locaux dans la table des symboles, hormis ceux commençant par la lettre "L". Si cette option n'est pas spécifiée, L068 place dans cette table uniquement les symboles globaux.

-Tadresse ou Zadresse

Ces deux options sont équivalentes et permettent de fournir à L068 l'adresse de base de la zone des programmes (text). Par défaut à zéro, l'adresse fournie (en hexadécimal, minuscules ou majuscules) peut avoir une valeur comprise entre \$0 et \$FFFFFFF. Cette option est spécialement utilisée pour placer des programmes en ROM.

-Dadresse

Définit l'adresse de base de la zone des données initialisées (zone "data"). Par défaut placée juste au dessus de la zone des programmes, l'adresse de cette zone peut être positionnée à la valeur absolue fournie en argument (une adresse hexadécimale comprise entre \$0 et \$FFFFFFF).

-Badresse

Définit l'adresse de base de la zone des données non-initialisées (bss). Par défaut placée au dessus de la zone "data", l'adresse de la 'BSS' peut être positionnée à la valeur absolue fournie en argument (une adresse hexadécimale comprise entre \$0 et \$FFFFFFF).

nom de fichier objet 'l' [.....]

Le ou les noms de fichiers objet créés par AS68. Ils correspondent aux fichiers que L068 doit "lier".

>nom du fichier de sortie des messages

Habituellement, les messages d'erreur sont affichés à l'écran. Si cette option est spécifiée, ceux-ci sont écrits dans le fichier spécifique dont le chemin et le nom sont fournis en arguments.

N.D.T.: Il est également possible d'ajouter les sorties à un fichier déjà existant en utilisant l'option '>>nom du fichier'. Rappelons d'autre part qu'en lançant le programme à partir de 'COMMAND.TOS', le périphérique de sortie peut être modifié. Ainsi: 'a: 'ligne de commande' >PRN: oriente les sorties vers l'imprimante.

3.3. Exemples de commande

*a: lo68 -S -O test.68K test.o

Cette commande "lie" le fichier objet "test.o" et produit un fichier exécutable "test.68K" dont les tables de symboles et de relocation ont été supprimées.

*a: lo68 -T4000 -D8000 -BC000 a.o b.o c.o

Cette commande "lie" les fichiers 'a.o', 'b.o' et 'c.o' et produit le fichier exécutable 'C.OUT'. La zone des programmes débute à l'adresse \$4000, la zone des données initialisées à l'adresse \$8000, la zone des données non-initialisées à l'adresse \$C000.

*a: lo68 -I -O test.68K test.o testl.o >b:error

Cette commande "lie" les fichiers objet 'test.o' et 'testl.o' pour produire le fichier exécutable 'test.68K'. Les débordements d'adresse sont ignorés, les messages d'erreur sont orientés vers le fichier 'error' créé sur le disque B.

4. Utilitaire de conversion: RELMOD

LINK68 et L068 produisent des fichiers relogeables au format CPM-68K. L'utilisation de RELMOD permet de transformer les fichiers produits en programmes compatibles avec le système GEMDOS.

La commande d'appel de cet utilitaire se présente sous la forme:

*a: relmod nom-de-fichier.68K nom-de-fichier.PRg

où le premier fichier correspond au programme produit par l'éditeur de liens, le second fichier correspondant au nom du fichier transformé au format GEM.

CHAPITRE VI

L'UTILITAIRE DE MISE AU POINT: SID68

1. LANCEMENT DE SID68

- 1.1. CONVENTIONS RELATIVES AUX COMMANDES
- 1.2. FORMULATION DES ADRESSES
- 1.3. REFERENCES SYMBOLIQUES
- 1.4. SORTIE DE SID68
- 1.5. GESTION DES INTERRUPTIONS ET SID68

2. COMMANDES DE SID68

- 2.1. COMMANDE C
- 2.2. COMMANDE D
- 2.3. COMMANDE E
- 2.4. COMMANDE F
- 2.5. COMMANDE G
- 2.6. COMMANDE H
- 2.7. COMMANDE I
- 2.8. COMMANDE K
- 2.9. COMMANDE L
- 2.10. COMMANDE M
- 2.11. COMMANDE P
- 2.12. COMMANDE R
- 2.13. COMMANDE S
- 2.14. COMMANDE T
- 2.15. COMMANDE U
- 2.16. COMMANDE V
- 2.17. COMMANDE W
- 2.18. COMMANDE X

SID68 est un outil interactif de test et d'aide à la mise au point de vos programmes dans un environnement GEM DOS. La lecture de ce chapitre suppose que vous n'ignorez rien du microprocesseur 68000, de l'assembleur (AS68) et du système d'exploitation GEMDOS.

Note des traducteurs: SID68 est un merveilleux outil, bien qu'un peu fruste et mal adapté à l'environnement GEM. Il constitue le recours ultime du programmeur à la recherche d'une 'bogue' et, à ce titre, nous en avons une longue expérience. Aussi nous sommes-nous permis de compléter largement la documentation originale par des exemples réels. Quelques commandes du SID68 n'étaient pas documentées, nous les expliquons ici. L'ampleur des ajouts effectués était telle que nous nous sommes dispensés de la mention "N.D.T." précédant chacun de ces ajouts. Le lecteur désireux de consulter les informations originales se référera à la version américaine de la documentation système.

1. Lancement de SID68

Le lancement de SID68 peut s'effectuer par l'une des commandes suivantes:

```
SID
SID nom_de_fichier
```

La première commande provoque le chargement et l'exécution de SID68. Une fois chargé, SID68 affiche un message d'introduction et le signe d'invite (-), lequel vous avertit que vous pouvez entrer une commande.

La deuxième commande provoque le chargement et l'exécution de SID68 puis le chargement du fichier désigné par 'nom_de_fichier' (chemin puis nom de fichier puis l'appendice). En fait cette seconde commande est équivalente à la première puis, après affichage du signe d'invite, à l'entrée de la commande E (voir 5.2.2.) destinée à charger un programme exécutable:

```
-E nom_de_fichier
```

REMARQUES:

- 1) SID68 a été renommé SID.TOS dans sa version destinée au ST.
- 2) SID.TOS peut être lancé à partir du Bureau GEM comme tout programme. Il peut également être lancé à partir du COMMAND.TOS par la commande 'SID.TOS', avec éventuellement un nom de fichier exécutable à la suite (mais séparé par un espace de la commande 'SID.TOS').

Exemple:

```
{a)SID.TOS COMMAND.TOS
```

```
*****
SID-68K for GEMDOS 3/22/85          Version 0.1
Serial fXXXX-0000-654321          All Rights Reserved
Copyright 1982,1983,1984,1985 Digital Research Inc.
*****
```

```
text base = 00022A04  data base = 0002622A  bss base = 00026E0E
text length = 00003826  data length = 00000BE4  bss length = 00002FD0
base page address = 00022904  initial stack pointer = 000F7FF8
-
```

3) SID.TOS peut être renommé en 'SID.TTP' et déclaré comme étant du type 'TOS avec paramètres', auquel cas vous entrez le nom du fichier à charger après lancement de SID dans la fenêtre destinée à cet effet.

1.1. Conventions relatives aux commandes

Lorsque SID68 est prêt à recevoir une commande, il affiche un tiret d'invite (-). Vous pouvez alors entrer une ligne de commande ou taper un Control-C (^C) pour terminer la phase de mise au point. Une ligne de commande peut comporter jusqu'à 82 caractères et doit se terminer par un retour de chariot (Return). On notera que le retour automatique de ligne n'étant pas actif sous SID, les 4 derniers caractères se superposeront en bout de ligne mais seront pris en compte. L'entrée de la ligne de commande s'effectue selon le standard GEMDOS en ce qui concerne la correction des erreurs (cf. le manuel du GEMDOS; signalons à ce propos la commande intéressante et peu connue 'Control-X' (^X) qui permet l'effacement de la ligne et le retour en début de ligne). SID68 ne prend en compte votre commande qu'après entrée de Return. Les commandes peuvent être entrées indifféremment en majuscules ou en minuscules.

Chaque code de commande est formé d'un seul caractère qui peut être suivi d'un ou plusieurs arguments, lesquels peuvent être des valeurs hexadécimales, des noms de fichier ou des informations complémentaires. Certaines commandes peuvent opérer sur un octet, un mot ou un long mot. La lettre W pour mot (word) et la lettre L pour long mot doit être ajoutée au caractère de commande si l'on désire que celle-ci opère sur des mots ou des longs mots. Le mode octet est le mode par défaut. Les arguments doivent être séparés les uns des autres par des virgules ou des espaces. En principe, le caractère de commande peut être séparé des arguments par un ou plusieurs espaces. Cette règle souffre toutefois des exceptions, qui seront détaillées lors de l'étude précise de chacune des commandes. La section 5.2 est consacrée à cette étude détaillée des commandes.

La table ci-dessous résume les commandes de SID68. Chacune de ces commandes est étudiée en détail par la suite.

RESUME DES COMMANDES DE SID68

Commande	Fonction
C	Exécute une routine avec envoi de paramètres
D	Affiche le contenu de mémoires en hexa. et ASCII
E	Charge un programme pour exécution
F	Remplit des mémoires avec une valeur donnée
G	Lance l'exécution avec d'éventuels points d'arrêt
H	Calcul hexadécimal et énoncé des symboles
I	Fixe ligne de commande d'un programme à exécuter
K	Fournit adresse de SID68 et travail sur symboles
L	Désassemble le contenu de mémoires
M	Déplacement de bloc mémoire
P	Fixe et supprime des points d'arrêt
R	Lecture de fichier sur disque
S	Fixe le contenu de mémoires
T	Exécution de programme pas à pas (Trace)
U	Mode pas à pas simplifié (voir commande T)
V	Fournit les adresses de début et fin d'un fichier
W	Ecrit le contenu de mémoires sur disque
X	Examine et modifie le contenu de registres 68000

1.2. Formulation des Adresses

De nombreuses commandes de SID68 nécessitent une ou plusieurs adresse(s) comme argument(s). Toutes les adresses doivent être entrées en hexadécimal et peuvent comporter jusqu'à huit chiffres hexadécimaux (32 bits) bien que le 68000 ne prenne en compte que les six chiffres inférieurs (24 bits, comme le bus d'adresses du 68000 l'autorise).

1.3. Références symboliques

Le SID68 est un outil d'aide symbolique, c'est-à-dire qu'il vous autorise à faire référence à des symboles au lieu d'adresses. Vous pouvez faire afficher la liste complète des symboles d'un programme par la commande H (voir 5.2.) et ils apparaîtront dans votre listing désassemblé à l'aide de la commande L. Lorsque vous entrez une commande comportant un symbole, faites précéder le nom du symbole d'un point, exemple:

commande.symbole

Ainsi, pour demander l'exécution de la commande G de la position actuelle du compteur de programme (PC) jusqu'à la routine QUIT du fichier exécutable chargé, entrez:

G,.QUIT

Le compilateur C d'Alcyon rajoute un trait de soulignement (_) au début de tous les symboles externes. Aussi, pour demander l'exécution de la commande G jusqu'à la fonction PUTCHAR, entrez:

G,._PUTCHAR

1.4. Sortie de SID68

Pour sortir de SID68, tapez les touches 'Control' et 'C' simultanément (^C) en réponse au signe d'invite (-) ou même en cours d'entrée de ligne de commande. Vous retournerez au programme d'appel (le Bureau GEM ou le COMMAND.TOS), après affichage du message "Bye".

1.5. Gestion des interruptions et SID68

SID68 opère avec les interruptions activées ou désactivées, et préserve l'état d'interruption du programme exécuté sous SID. Lorsque SID68 a le contrôle de l'unité centrale, que ce soit lors de son initialisation ou lorsqu'il redonne la main après l'exécution d'un fragment de programme, le masque des interruptions est identique à celui de l'initialisation, sauf pour quelques routines critiques durant lesquelles les interruptions sont inhibées. Lorsque le programme à tester a le contrôle de l'unité centrale, l'état du registre d'état du 68000, qui peut être affiché par la commande X, définit le masque des interruptions.

On notera que SID68 utilise les exceptions Trace et Illegal afin de gérer l'exécution du programme à tester. Pour cette raison, les programmes à tester ne doivent pas utiliser ces exceptions.

2. COMMANDES de SID68

Ce sous-chapitre fournit la liste des commandes de SID68 et leurs arguments. Les commandes de SID68 vous autorisent principalement à contrôler l'exécution d'un programme et à afficher et modifier le contenu de mémoires et l'état du processeur.

Parmi les exemples d'utilisation de commandes fournis ci-dessous, beaucoup nécessitent le chargement préalable d'un programme. Nous avons choisi COMMAND.TOS pour ces exemples. Si vous désirez les reproduire, il faut donc cliquer SID.TOS au Bureau puis entrer la commande:

E COMMAND.TOS

qui constitue une demande de chargement de COMMAND.TOS pour test par SID68.

2.1. Commande C ('Call', appel de routine)

La commande C permet d'exécuter une routine (partie de programme se terminant par l'instruction "rts") en lui envoyant de 0 à 10 paramètres LONGS. En retour, est affiché le message "Return value --> " suivi du

contenu long hexadécimal du registre D0 du 68000 (valeur de retour utilisée par le compilateur C d'Alcyon).

SYNTAXE: C<adr_rout>[,param1][,...][,param10]

où <adr_rout>, argument INDISPENSABLE, désigne l'adresse de la routine à exécuter (ou le label de cette routine).
param1,...,param10, arguments optionnels, correspondent aux paramètres LONGS envoyés à la routine.

EXEMPLE : -C_bios,000B0000
Return value --> 0h

La routine "bios" de COMMAND.TOS (chargé auparavant par E) est exécutée. Cette routine correspond à un appel du Trap 13. Elle est appelée ici avec un paramètre entier \$B (correspondant à la fonction Getshift(a)) et un deuxième paramètre entier nul (correspondant à une remise à 0 des indicateurs de touches spéciales). La valeur retournée correspond à l'état des touches spéciales lors de l'appel (ce sera par exemple 10h si la touche CapsLock était active lors de l'appel).

REMARQUES:

- 1) Cette fonction n'est pas décrite dans la documentation d'origine.
- 2) L'adresse (ou le label) de la routine doit suivre IMMEDIATEMENT le code de commande C. Autrement dit, pas d'espaces entre C et l'adresse.
- 3) Les paramètres à envoyer sont des LONGS. Si la routine nécessite un ou des paramètre(s) entier(s), il faut le(s) placer dans un ou des longs comme dans l'exemple ci-dessus.
- 4) L'état des registres du 68000 est sauvegardé par SID lors de l'appel de la commande C. Pour l'utilisateur, cela offre l'inconvénient de rendre invisibles les modifications des registres consécutives à l'appel, sauf pour D0 dont le contenu est affiché en retour. Mais cela présente l'avantage de permettre un appel de routine alors qu'on se trouve dans une autre routine sans modification du contexte!

2.2. Commande D ('Display', lecture de mémoires)

La commande D permet de faire afficher à l'écran le contenu de mémoires sous forme d'octets, de mots ou de longs mots et en ASCII.

SYNTAXE: D[adr_debut][,adr_fin]
ou DW[adr_debut][,adr_fin]
ou DL[adr_debut][,adr_fin]
ou D@<num_reg_ad> (ou DW@<num_reg_ad>, ou DL@<num_reg_ad>)

où: adr_debut désigne l'adresse OPTIONNELLE de début de lecture
adr_fin désigne l'adresse OPTIONNELLE de fin de lecture
num_reg_ad OBLIGATOIRE désigne le numéro d'un registre d'adresse

La commande D formate les contenus de mémoires sur des octets, la commande DW les formate sur des mots et DL sur des longs mots.

EXEMPLES:

```
-D 00000100,0000010F
00000100 40 B0 3B 04 35 58 2F AC 00 FC 27 AE 1E A8 18 FC @.:.5X/...'.....
-DW 00000110,0000011F
00000110 1350 0DA4 00FC 313C 00FC 29CE F148 EB9C .P....1<...).H.
-DL 00000120,0000012F
00000120 E5F0E044 00FC281C 00FC2794 00FC27FA ...D...('...'...
```

REMARQUES:

- 1) Employée seule, la commande D (ou DW, ou DL) fait débiter la lecture à la position courante du compteur de programme (PC), soit 0 si aucun programme n'a été chargé par la commande E.
- 2) Employée sans adresse de fin, la commande D (ou DW, ou DL) affiche onze lignes de contenus de mémoire, soit 176 octets. Une nouvelle commande D affichera les 176 octets suivants, etc.
- 3) Si les adresses de début et de fin sont éloignées, l'affichage s'effectuera sur de nombreuses lignes avec scrolling. Les combinaisons de touches Control-S et Control-Q vous permettent respectivement de figer l'affichage puis de le faire reprendre. L'appui de toute autre touche provoquera l'arrêt de l'affichage en fin de ligne courante.
- 4) Il est interdit de placer une adresse de fin même précédée d'une virgule sans adresse de départ.
- 5) La commande D (ou DW ou DL) peut être séparée du ou des adresses optionnelles par un ou plusieurs espaces (cf. exemple ci-dessus).
- 6) Seuls les codes ASCII compris entre 0x20 et 0x7F sont affichés, les autres étant remplacés par un point.
- 7) Une bogue fait qu'avec la commande DW seuls 15 octets de la dernière ligne affichée sont convertis en ASCII. Avec DL, seuls 13 octets de la dernière ligne affichée sont convertis!
- 8) L'affichage des mémoires 00000000 à 003FFFFF est autorisé par SID, qui gère le passage en superviseur nécessaire pour accéder aux mémoires 0 à 7FF. Une demande d'affichage des mémoires à partir de 00400000 aura par contre toutes les chances de 'planter' le système.
- 9) Si l'on utilise la commande DW ou DL avec une adresse de départ impaire, celle-ci est convertie dans l'adresse paire supérieure.
- 10) La commande D@ (et DW@, et DL@) doit TOUJOURS être suivie d'un numéro de registre d'adresse. Sont alors affichés onze lignes, soit 176 octets, correspondant aux contenus des mémoires à partir de l'adresse contenue dans le registre d'adresse dont le numéro a été envoyé en appel.
Exemple: D@7 affichera les 176 octets contenus dans la pile.

2.3. Commande E ('Execute', exécuter)

La commande E charge un fichier exécutable en mémoire de sorte qu'une commande C, G, T ou U permette ensuite l'exécution de tout ou partie de ce programme.

SYNTAXE: E <nom_de_fichier>

où <nom_de_fichier> désigne le nom d'un programme (.PRG, .TOS ou .TTP)

EXEMPLE:

```
e command.tos
text base = 00022A04 data base = 0002622A bss base = 00026E0E
text length = 00003826 data length = 00000BE4 bss length = 00002FD0
base page address = 00022904 initial stack pointer = 000F7FF8
```


REMARQUES:

- 1) La commande E et le nom de fichier peuvent être séparés par des espaces.
- 2) Le nom de fichier peut être un chemin (ex: A:\MASTER2.DEV\COMMAND.TOS).
- 3) Le fichier DOIT être un programme (voir commande R).
- 4) Après chargement, les "coordonnées" du programme sont affichées (voir l'exemple ci-dessus). Le programme est prêt à être exécuté.
- 5) Après cette commande, le compteur de programme (PC) pointe le début du programme chargé. Les registres sont remis à 0, sauf la pile et le PC.
- 6) Il n'est possible d'exécuter qu'UNE SEULE commande E par session. Si vous désirez tester un autre programme, il faut quitter SID puis le relancer.

2.4. Commande F ('Fill', remplissage de mémoires)

La commande F permet de placer une même valeur dans un bloc mémoire, défini par deux adresses. La valeur peut être un octet (commande F), un mot (commande FW) ou un long mot (commande FL).

SYNTAXE: F<adr_debut><adr_fin><octet>
 FW<adr_debut><adr_fin><mot>
 FL<adr_debut><adr_fin><long_mot>

où: adr_debut et adr_fin sont des arguments INDISPENSABLES, désignant l'adresse de début et de fin du remplissage sur huit octets hexadécimaux au maximum (ou des symboles)
 octet, mot, long_mot, désigne la valeur sur 8, 16 ou 32 bits qui doit être placée dans les mémoires.

EXEMPLE: F 200,2FF,0
 place des octets nuls dans les mémoires 0x200 à 0x2FF.

REMARQUES:

- 1) La commande F et les arguments peuvent être séparés par des espaces.
- 2) Si l'adresse de fin est plus petite que l'adresse de départ, s'affiche un point d'interrogation (?), signe de non-exécution de commande.
- 3) Le même signe d'erreur résulte de l'emploi d'une adresse impaire avec la commande FW ou FL, ainsi que d'une valeur à placer trop importante (>0xFF pour F, >0xFFFF pour FW et >0xFFFFFFFF pour FL).
- 4) Si l'on tente de placer une valeur dans des mémoires inexistantes (par exemple supérieures à 0xFFFF sur un 1040), un message d'erreur est affiché pour chaque mémoire à remplir (bogue!). Mais, attention, si l'adresse est supérieure à 0x400000, cela "plantera" le système.
- 5) La commande F est peu utile sinon pour remettre un bloc à 0. Par contre, elle peut être très dangereuse si elle est mal employée...

2.5 Commande G ('Go', exécuter jusqu'à)

La commande G transfère le contrôle du processeur au programme à tester, et fixe sur option de un à dix points d'arrêt.

SYNTAXE: G[adr_debut][,arret_1][...][,arret_10]

où adr_debut est un argument OPTIONNEL désignant l'adresse de début d'exécution.
 arret_1 à arret_10 sont des arguments OPTIONNELS définissant des ADRESSES où l'exécution du programme doit être stoppée.

EXEMPLE:

```
-G, in term
PC=00022A9A USP=00027478 SSP=00027474 ST=2300->SUP IM=3
D 00FC07EE 00022B08 00000000 00000000 00000000 00000000 00000000 00000000
A 00000404 0000093A 00000000 00000000 00000000 00022904 00000000 00027474
in term:
move.l f$22AEA,-(a7)
```

Cet exemple suppose que COMMAND.TOS a été chargé précédemment. La commande 'G, in term' demande l'exécution de COMMAND.TOS à partir de la position courante du compteur de programme PC (c'est-à-dire à partir du début du programme si aucune commande G,T ou U n'a encore été exécutée) jusqu'à la routine 'in_term' appartenant à ce programme.

REMARQUES:

- 1) La commande G et ses arguments peuvent être séparés par des espaces.
- 2) L'utilisation de la commande G n'est possible que si un programme a été chargée précédemment par la commande E. Par contre il n'est pas indispensable que l'adresse de départ ou les points d'arrêt appartiennent à ce programme!
- 3) Si l'adresse de départ n'est pas mentionnée, l'exécution débute à partir de l'adresse pointée par le compteur de programme (PC).
- 4) Si aucun point d'arrêt n'est indiqué, l'exécution se poursuit jusqu'à la sortie du programme appelé, laquelle produira habituellement la sortie de SID également (bogue?).
- 5) Lorsque le contrôle a été passé au programme par G, celui-ci est exécuté en temps réel jusqu'à ce qu'un point d'arrêt soit rencontré. A ce moment, le contenu des registres est affiché, de même que l'instruction correspondant au point d'arrêt (voir exemple ci-dessus). Cette instruction N'A PAS ENCORE ETE EXECUTEE. Tous les points d'arrêt sont alors effacés (mais non les points de passage, voir commande F).
- 6) Lorsqu'une erreur système se produit durant l'exécution, celle-ci est interrompue et le type de l'erreur ainsi que le contenu des registres lors de cette erreur sont affichés. Dans certains cas (heureusement rares), cette erreur provoquera le 'plantage' irrémédiable du ST (reset nécessaire) et parfois aussi le blocage de SID...

2.6. Commande H ('Hexadecimal Math', calcul en hexadécimal)

Cette commande calcule la somme et la différence de deux nombres hexadécimaux considérés comme des longs mots.

SYNTAXE: H<long_mot_1><long_mot_2>

où long_mot_1 et long_mot_2 sont deux nombres hexadécimaux.

EXEMPLE:

```
-h 27000,12345
00039345 00014CBB
```

La somme des nombres 0x27000 et 0x12345 est égale à 0x39345. Leur différence est égale à 0x14CBB.

REMARQUES:

- 1) La commande H et les arguments peuvent être séparés par des espaces.
- 2) Les arguments peuvent être des symboles (labels).
- 3) La commande H employée SANS ARGUMENT permet d'obtenir un affichage de tous les symboles du programme chargé. Malheureusement, une bogue fait que l'affichage de ce listing ne s'effectue pas correctement. Seuls Control-S et Control-Q permettent de figer et de reprendre l'affichage.
- 4) La commande H suivie d'UNE SEULE ADRESSE permet d'obtenir l'affichage du symbole éventuellement affecté à cette adresse.

2.7. La commande I ('Input', entrée de ligne de commande)

La commande I permet de placer une commande dans le tampon de commande de la page de base du programme à tester (chargé par la commande E).

SYNTAXE: I<commande>

où commande correspond à l'ensemble de la ligne de commande.

EXEMPLE:

-Icat

Suppose qu'un programme (en l'occurrence COMMAND.TOS) a été chargé auparavant par la commande E. Place 'cat' dans la ligne de commande de ce programme, c'est-à-dire qu'ici COMMAND.TOS sera exécuté ensuite par G avec la commande 'cat' qui sera donc elle-même exécutée par ce programme (voir le manuel du GEMDOS; cette 'ligne de commande' correspond aux paramètres d'un programme 'TOS avec paramètres').

REMARQUES:

- 1) Ne pas séparer la commande I et son argument par des espaces, faute de quoi ces espaces seront intégrés à l'argument.
- 2) La commande I n'est utilisable que si un programme a été chargé par E.
- 3) La 'ligne de commande' correspond généralement à un ou plusieurs noms de fichier. La longueur de cette ligne de commande est limitée par SID à 81 caractères.
- 4) Le nombre de caractères de la commande argument est calculé par SID. Il ne doit donc pas être précisé.

2.8. Commande K ('Kernel', informations sur SID)

La commande K permet d'obtenir des informations sur l'implantation en mémoire du programme SID.TOS lui-même. La commande KA permet d'ajouter un symbole à la table des symboles. La commande KH permet de désactiver un symbole. La commande KR permet de réactiver un ou tous les symboles.

SYNTAXE: K
KA<nom_symbole><,<adresse>[,type]
KH<adresse>
KR<adresse>

où: nom_symbole est un symbole de 8 lettres OBLIGATOIREMENT.
adresse est l'adresse correspondant au symbole
type est un MOT (OPTIONNEL) correspondant au type du symbole.

EXEMPLE:

-k
SID begins at 0001A5B0H with base page at 0001A4B0H
Symbol table at: 00218B8
-kA deuxinst,22A0E
-kH 22A0E
-KR
-

La commande K fournit des informations sur le SID. La commande KA affecte le symbole 'deuxinst' à l'adresse correspondant ici à la deuxième instruction du programme COMMAND.TOS chargé par E. La commande KH désactive ce nouveau symbole et la commande KR réactive tous les symboles, donc naturellement le symbole 'deuxinst' désactivé.

REMARQUES:

- 1) Cette fonction non documentée du SID est un fourre-tout.
- 2) Les commandes KA, KH et KR ne peuvent être employées que si un programme a été chargé auparavant par la commande E.
- 3) La fonction KA ne fonctionne malheureusement pas dans la version du SID actuellement fournie. Si votre version fait 28800 octets, voici un 'truc' pour la faire fonctionner. Ce truc consiste à modifier une adresse relative dans le programme. Vous ne devez l'utiliser que si vous avez bien saisi toutes les ficelles du SID et, dans tous les cas, sauvez une version du SID fournie au départ sur une disquette de sauvegarde. Lancez SID.TOS puis entrez les commandes suivantes:

```
-r sid.tos
Start = 00021894 End = 00028913
-h 21894,1720
00022FB4 00020174
-sl 22FB4
00022FB4 0000710C 0000710A
00022FB8 B0816D10
```

-w sid.tos
La commande 'r sid.tos' lit le fichier 'sid.tos' (alors qu'on est dans SID!). Les adresses de début et de fin s'affichent. ATTENTION, ces adresses seront certainement différentes sur votre système. Calculez alors la somme adresse_départ+1720 par la commande H. Vous obtenez ainsi l'adresse à modifier. Pour la modifier utilisez la commande SL, la valeur qui s'affiche DOIT ÊTRE ÉGALE à 0000710C, faute de quoi vous avez fait une erreur ou bien votre version du SID est différente. Si la valeur est égale à 0000710C, remplacez-la par 0000710A. Puis remplacez le fichier modifié sur votre disque par la commande 'w sid.tos'. Le tour est joué. Vous pouvez utiliser la commande KA dès votre prochain appel de SID.TOS.

- 4) Le 'type' du symbole correspond aux caractéristiques définies dans NM68 (Cf. chapitre 7, à7). Le type par défaut est 0xA700.

5) Le nombre maximal de symboles pris en compte par SID est de 255.

2.9. Commande L ('List', désassemblage de contenus mémoire)

La commande L fournit le désassemblage du contenu des mémoires.

SYNTAXE: L<adr_debut><adr_fin>

où: adr_debut et adr_fin, arguments OPTIONNELS, correspondent aux adresses de début et de fin du désassemblage.

EXEMPLE:

```
-L FC0020,FC002F
00FC0020      move.w  $S2700,SR
00FC0024      reset
00FC0026      cmpi.l  $FA52235F,$FA0000
```

désassemble les mémoires comprises entre FC0020 et FC002F (ROM).

REMARQUES:

- 1) La commande L et ses arguments peuvent être séparés par des espaces.
- 2) L'adresse de départ, si elle est fournie, doit être PAIRE.
- 3) Si l'adresse de départ n'est pas précisée, c'est l'adresse contenue dans le compteur de programme (PC) qui est considérée comme adresse de début.
- 4) Si l'adresse de fin n'est pas précisée, SID désassemble douze lignes d'affilée. Par une nouvelle commande L sans arguments, on désassemble les douze lignes suivantes, etc.
- 5) Si l'adresse de fin est plus grande que l'adresse de début, un point d'interrogation est affiché et aucun désassemblage n'est effectué.
- 6) Le désassemblage peut être stoppé à tout moment par l'appui d'une touche, Control-S et Control-Q permettant de le figer et de le reprendre.
- 7) Lors du désassemblage, les symboles, s'il y en a, sont affichés tant pour les labels que pour les variables.
- 8) La syntaxe du désassemblage est conforme à celle de l'assembleur AS68 (voir le chapitre sur ce sujet). Toutefois l'ensemble des valeurs fournies sont en hexadécimal.

2.10. Commande M ('Move', déplacement de bloc mémoire)

Cette commande copie le contenu d'un bloc mémoire d'une adresse vers une autre. Le bloc destination peut recouvrir en tout ou partie le bloc source.

SYNTAXE: M<adr_debut><,adr_fin><,nouv_adr>

où: adr_debut et adr_fin correspondent à l'adresse de base et à l'adresse de fin du bloc source, nouv_adr à l'adresse de base du bloc destination. Arguments INDISPENSABLES.

EXEMPLE:

```
M F8000,F9F40,F9F90
```

recopie les mémoires 0xF8000 à 0xF9F40, à partir de 0xF9F90. Sur un

1040, cela provoque une copie du haut de l'écran sur le deuxième quart de cet écran.

REMARQUES:

- 1) La commande M et les arguments peuvent être séparés par des espaces.
- 2) Tous les arguments sont INDISPENSABLES.
- 3) Le bloc source et le bloc destination peuvent se recouper sans aucun problème.
- 4) Les adresses doivent être comprises entre 0 et 3FFFFFF.
- 5) Attention, si vous tentez de déplacer le programme chargé en mémoire par la commande E, les appels à des adresses fixes ne seront plus corrects.

2.11. Commande P ('Pass Points', points de passage)

La commande P permet de fixer, de supprimer et d'afficher les points de passage. Un point de passage est un point d'arrêt fixé jusqu'à sa suppression éventuelle, contrairement aux points d'arrêt placés par la commande G, lesquels sont supprimés dès le premier point d'arrêt rencontré. A chaque point de passage est associé un compteur pouvant varier entre 1 et 0xFFFF. Ce compteur indique le nombre de fois que l'instruction correspondant à ce point de passage peut être exécutée avant que le contrôle du programme soit rendu au SID. On peut définir jusqu'à 32 points de passage.

Attention, l'instruction correspondant à un point de passage est exécutée avant l'arrêt, contrairement à ce qui se passe avec un point d'arrêt. L'arrêt après un point de passage s'effectue lorsque le compteur est à 1.

SYNTAXE: -P efface tous les points de passage
 -P<adr_point> efface le point de passage d'adresse adr_point
 P affiche tous les points de passage
 P<adr_point> place un point de passage en adr_point (compteur=1)
 P<adr_point><,nombre> place un point de passage en adr_point et place le compteur de ce point à nombre.

EXEMPLE:

```
-p22A1A,6      fixe un point de passage en 22A1A avec 6 comme compteur
               demande la liste des points de passage
-p            supprime le point de passage en 22A1A
```

REMARQUES:

- 1) La commande P et son ou ses argument(s) ne doivent jamais être séparés par un ou des espaces.
- 2) L'adresse d'un point de passage doit toujours être paire.
- 3) L'adresse peut être remplacée par le symbole lui correspondant.
- 4) Le nombre de points de passage possibles est de 32 et non de 16 comme indiqué par la documentation originale.
- 5) La liste des points de passage est fournie en deux fois si elle dépasse 16 points. Malheureusement une bogue fait qu'il n'y a pas d'arrêt entre les deux séries d'affichage bien que le message "Continue (y/n)?" soit affiché. Utiliser Control-S et Control-X pour figer et reprendre l'affichage.

2.12. Commande R ('Read', Lecture de fichier)

La commande R permet de lire un fichier sur disque et de le stocker en mémoire. Les adresses de départ et de fin du bloc mémoire dans lequel est chargé le fichier sont affichées après chargement. L'adresse de début par défaut de la commande D devient l'adresse de base du fichier chargé.

SYNTAXE: R<nom_de_fichier>

où nom_de_fichier est le nom d'un fichier.

EXEMPLE:

Rlarge.s

Start = 00021894 End = 00022FFE

REMARQUES:

- 1) La commande R et le nom de fichier peuvent être séparés par des espaces.
- 2) La commande R n'est pas antinomique de la commande E. Toutefois, il n'est possible de charger qu'un fichier par R, pas plusieurs.

2.13. Commande S ('Set', modifie le contenu de mémoires)

La commande S permet de modifier le contenu d'octets, de mots ou de longs mots en mémoire.

SYNTAXE: S<adresse>
SW<adresse>
SL<adresse>

où: adresse correspond à l'adresse de l'octet, du mot ou du long mot à modifier.

EXEMPLE:

s300

00000300 FF 00

00000301 FF

La commande 's300' provoque l'affichage en ligne suivante de l'adresse 0x300 et de son contenu, ici 0xFF, et un curseur clignotant afin que l'on puisse entrer une nouvelle valeur, ici 0x00. Puis l'adresse de la mémoire suivante et son contenu sont affichés et ainsi de suite. On stoppe cette série en entrant une lettre de code supérieur à la lettre 'F' ou un point. Si l'on entre un retour de chariot, le contenu de l'adresse affichée est inchangé et la série se poursuit.

REMARQUES:

- 1) La commande S et l'argument peuvent être séparés par des espaces.
- 2) L'adresse fournie en argument peut être un symbole.
- 3) Si la valeur fournie est trop élevée, un point d'interrogation est affiché et l'entrée de valeurs est stoppée.
- 4) Si l'on utilise la commande SW ou SL, l'adresse fournie en argument doit être paire.

2.14. Commande T ('Trace', pas à pas)

La commande T permet d'exécuter pas à pas le programme chargé par la commande E. Après chaque exécution d'instruction, SID affiche le contenu courant des registres du 68000 et l'instruction désassemblée qui va être exécutée ensuite. L'instruction exécutée correspond à l'adresse pointée par le compteur de programme (PC).

La commande T suivie d'un nombre n (voir syntaxe ci-dessous) provoque l'exécution de n instructions à la suite, SID affichant l'état des registres du 68000 entre chacune de ces instructions. Il est possible d'interrompre l'exécution en pressant n'importe quelle touche.

La commande Tw provoque une exécution d'instruction sans appels à des sous-programmes (en fait un sous-programme appelé EST exécuté mais de façon transparente à l'utilisateur). Cette commande très performante rend le mode pas à pas réellement opérationnel, la recherche d'erreur pouvant s'effectuer par tests successifs de plus en plus fins (les routines testées et "sûres" pouvant être passées par Tw).

L'utilisation de la commande T, remettant à jour le compteur de programme PC, modifie par le fait même l'adresse de départ par défaut des commandes G et L, qui n'est autre que ce compteur de programme.

SYNTAXE: T[nombre]

Tw

où: nombre correspond au nombres d'instructions à exécuter pas à pas avant de rendre la main à l'utilisateur.

EXEMPLE:

-t

PC=00022A0A USP=000F7FF8 SSP=0000755A ST=0300=>IM=3

D 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

A 00000000 00000000 00000000 00000000 00000000 00027486 00000000 000F7FF8

move.l \$4(a7),-(a5)

-t2

PC=00022A0E USP=000F7FF8 SSP=0000755A ST=0300=>IM=3

D 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

A 00000000 00000000 00000000 00000000 00000000 00027482 00000000 000F7FF8

clr.l -(a5)

0001 PASS 22A10

PC=00022A10 USP=000F7FF8 SSP=0000755A ST=0300=>IM=3

D 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

A 00000000 00000000 00000000 00000000 00000000 0002747E 00000000 000F7FF8

movea.l a5,a7

-

Dans cet exemple, la commande T exécutée juste après le chargement de COMMAND.TOS par E provoque l'exécution de la première instruction du programme puis la commande T2 provoque l'exécution de deux instructions de suite. La ligne '0001 PASS 22A10' signale qu'un point de passage avait

été placé en 22A10 (voir commande P) et que la nouvelle valeur du compteur pour ce point de passage est 1.

REMARQUES:

- 1) La commande T et l'argument peuvent être séparés par des espaces.
- 2) Cette commande n'est utilisable que si un programme a été chargé par la commande E. Par contre, il est possible de 'tracer' hors du programme chargé (pas en ROM cependant).
- 3) Comme l'exemple ci-dessus le montre, les points de passage définis par la commande P sont pris en compte et lorsque le compteur d'un de ces points est à 1 (donc passe à 0), la commande T<nombre> stoppe à ce point de passage.
- 4) Si l'on fait précéder la commande T, avec ou sans arguments, du signe - les symboles éventuels ne sont pas affichés.
- 5) Les instructions 'trap' sont exécutées de façon invisible en pas à pas, ce qui semble logique puisqu'elles se traduisent généralement par des appels à des routines en ROM. Il en va de même des instructions de la ligne "A" et de la ligne "F".

2.15 Commande U ('Untrace', mode pas à pas simplifié)

La commande U est identique à la commande T si ce n'est que l'état des registres du 68000 n'est affiché qu'après la dernière instruction exécutée lorsqu'on a appelé la commande U<nombre>. La commande Uw est identique à la commande Tw.

SYNTAXE: U ou U<nombre> ou Uw (voir T ci-dessus)

EXEMPLE:

```
-u
PC=00022A0A USP=000F7FF8 SSP=0000755A ST=0300=>IM=3
D 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
A 00000000 00000000 00000000 00000000 00000000 00027486 00000000 000F7FF8
move.l $4(a7),-(a5)
-u2
PC=00022A10 USP=000F7FF8 SSP=0000755A ST=0300=>IM=3
D 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
A 00000000 00000000 00000000 00000000 00000000 0002747E 00000000 000F7FF8
movea.l a5,a7
```

L'exemple précédent a été repris dans des conditions identiques mais sans définition de point de passage.

REMARQUES: (voir remarques de la commande T, qui s'appliquent toutes ici).

2.16. Commande V ('Value', informations sur le fichier chargé)

La commande V permet d'obtenir des informations sur le fichier chargé par la commande E (fichier exécutable) ou la commande R. Si le fichier a été chargé par E, la commande V affiche l'adresse de début et la longueur de chacune des zones du programme ainsi que le pointeur de la page de base et le pointeur de pile initial. Si un fichier a été chargé par R, seules l'adresse de début et de fin sont fournies.

SYNTAXE: V

EXEMPLE:

-V

Start = 00021894 End = 00022FFE

Voir la commande R par laquelle ce fichier a été chargé.

REMARQUE:

- 1) Les informations fournies correspondent à la dernière commande E ou R entrée.

2.17. Commande W ('Write', Ecriture de fichier)

Cette commande écrit le contenu d'un bloc mémoire sur disque. L'adresse de départ et de fin du bloc mémoire copié sont ceux résultant d'une commande R antérieure. Si aucun nom de fichier n'est fourni, le SID reprend le nom du fichier chargé.

SYNTAXE: W[nom_de_fichier]

où: nom de fichier, OPTIONNEL, désigne un nom au format GEMDOS
(ex: A\MASTER2.DEV\LARGE2.S)

EXEMPLE:

-W LARGE2.S

Recopiera le contenu du fichier LARGE.S (chargé précédemment par R et éventuellement modifié) dans un nouveau fichier 'LARGE2.S'

REMARQUES:

- 1) Des espaces peuvent figurer entre W et le nom de fichier.
- 2) Si le fichier que l'on désire écrire existe déjà, il est effacé SANS ALERTE PREALABLE et un nouveau fichier est créé.
- 3) L'écriture n'est pas vérifiée, c'est-à-dire qu'il peut arriver qu'une partie seulement du fichier soit écrite, par exemple si votre disque n'a plus suffisamment d'espace disponible.
- 4) Selon la documentation originale, une adresse de départ et une adresse de fin pourraient être ajoutées à W<nom_de_fichier>. Malheureusement, cette option n'EXISTE PAS dans cette version de SID68. Il est toutefois possible de leurrer SID en modifiant dans la zone des variables de SID l'adresse et la longueur du programme à écrire. Pour cela, demandez l'adresse d'implantation en mémoire de SID par la commande K. Puis, en ajoutant 00068C6 à cette adresse par la commande H, vous obtiendrez l'adresse du pointeur de début de fichier et en ajoutant 000068CA, vous obtiendrez la longueur du fichier. Il suffit alors de placer dans la première adresse l'adresse du bloc que vous voulez sauvegarder et dans la seconde sa longueur et vous pouvez sauvegarder sur disque n'importe quel bloc de mémoire.

2.18. Commande X ('Examine', affichage des registres du 68000)

La commande X affiche l'état complet des registres du 68000, à savoir le compteur de programme (PC), le pointeur de pile Utilisateur (USP), le pointeur de pile Superviseur (SSP), le registre d'état (ST), les huit

registres de données et les huit registres d'adresses. Est également affichée l'instruction désassemblée pointée par le compteur de programme.

SYNTAXE: X
X<nom_de_registre>

où: nom de registre désigne un registre (D0,D1,D2,D3,D4,D5,D6,D7,A0,A1,A2,A3,A4,A5,A6,A7,PC,USP,SSP,ST)

EXEMPLE:

```
-X
PC=00022A10 USP=000F7FF8 SSP=0000755A ST=0300=>IM=3
D 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
A 00000000 00000000 00000000 00000000 00000000 0002747E 00000000 000F7FF8
movea.l a5,a7
-XD1
D1=00000000 0000002D
```

La commande X provoque l'affichage de tous les registres dans l'état actuel du programme (ici l'état succédant à l'exemple de la commande U). La commande XD1 provoque l'affichage du contenu du registre de donnée D1 puis un curseur clignote comme pour la commande S. Ici la valeur 2D a été entrée au clavier. Un retour de chariot sans entrée de données aurait laissé le registre inchangé.

REMARQUES:

- 1) Aucun espace ne doit exister entre X et un nom de registre éventuel.
- 2) Les informations fournies à la suite du contenu de ST (première ligne affichée après la commande X) ont la signification suivante:
 - TR bit de trace (indique que le mode 68000 trace est actif)
 - SUP bit de mode superviseur
 - IM=x fournit le masque d'interruptions(trois bits)
 - EXT bit d'extension à 1
 - NEG bit de résultat négatif à 1
 - ZER bit de résultat nul à 1
 - OFL bit de débordement arithmétique à 1
 - CRY bit de retenue à 1
- 3) La commande X peut être utilisée même si aucun fichier n'a été chargé.
- 4) Contrairement aux informations de la documentation originale, il est possible de modifier le registre d'état par la commande XST. Cela permet même de passer le plus simplement du monde de mode utilisateur en mode superviseur et vice-versa. Attention cependant à ne pas positionner le masque des interruptions à 7, le système serait alors bloqué puisque les interruptions clavier ou souris ne seraient plus prises en compte. Reset obligatoire dans ce cas!
- 5) Une grande prudence est requise lors de la modification de registres par cette commande. Le contrôle d'erreur par le SID est restreint, ainsi est-il possible de fixer le compteur de programme à une adresse impaire... inutile de décrire le résultat lors de la première commande G ou T ou U qui suivra!

CHAPITRE VII

LES AUTRES UTILITAIRES

1. L'UTILITAIRE D'ARCHIVAGE AR68
 - 1.1. INTRODUCTION
 - 1.2. LA SYNTAXE D'AR68
 - 1.3. SPECIFICATIONS
 - 1.4. COMMANDES ET OPTIONS
 - 1.5. MESSAGES D'ERREUR
2. L'EDITEUR DE COMMANDES: COMMAND.TOS
 - 2.1. RAPPELS SUR LE SYSTEME CPM
 - 2.2. PRINCIPES DE FONCTIONNEMENT
 - 2.3. COMMANDES
3. L'UTILITAIRE DE TRAITEMENT PAR LOTS: BATCH
 - 3.1. CONVENTIONS D'APPEL
 - 3.2. APPEL SANS ARGUMENT
 - 3.3. APPEL AVEC ARGUMENT(S)
 - 3.4. INTERROMPRE UNE COMMANDE
4. L'UTILITAIRE D'AFFICHAGE: DUMP
 - 4.1. CONVENTIONS D'APPEL
 - 4.2. FORMAT DE SORTIE
5. L'UTILITAIRE D'INFORMATIONS SUR FICHIERS: SIZE68
 - 5.1. CONVENTIONS D'APPEL
 - 5.2. EXEMPLES DE COMMANDES
6. L'UTILITAIRE DE FORMAT "S-RECORD": SEND
 - 6.1. CONVENTIONS D'APPEL
 - 6.2. LE FORMAT "S-RECORD"
 - 6.3. EXEMPLES DE COMMANDES
7. L'UTILITAIRE D'AFFICHAGE DE SYMBOLES: NM68
 - 7.1. CONVENTIONS D'APPEL
 - 7.2. EXEMPLE DE COMMANDE
8. L'UTILITAIRE DE REFERENCES: MMXREF
9. L'UTILITAIRE D'EFFACEMENT DE FICHIER: RM
10. L'UTILITAIRE DE TEMPORISATION: WAIT

Ce chapitre décrit les utilitaires AR68, COMMAND.TOS, BATCH, DUMP, SIZE68, SEND, NM68 et MMXREF, RM et WAIT.

- * AR68 permet l'archivage de bibliothèques,
- * COMMAND.TOS est un éditeur de commandes proche de CPM-68K,
- * BATCH est un utilitaire d'exécution de commande par lots,
- * DUMP affiche le contenu d'un fichier en hexadécimal et en ASCII,
- * SIZE68 affiche la taille totale d'un fichier et les tailles détaillées de ses différents segments,
- * SEND crée, à partir d'un programme, un fichier au format "S-record" de MOTOROLA,
- * NM68 affiche la table des symboles d'un programme,
- * MMXREF est un analyseur de références,
- * RM est utilisé pour supprimer des fichiers,
- * WAIT est un utilitaire d'attente d'appui de touche.

1. L'utilitaire d'archivage: AR68

1.1. Introduction

L'utilitaire d'archivage AR68 permet de créer une bibliothèque de toutes pièces ou bien, à partir d'une bibliothèque déjà existante, d'y remplacer, d'y ajouter, d'y lister ou d'en extraire des modules objets.

1.2. La syntaxe d'AR68

Pour appeler AR68, utilisez le format de commande qui suit (les arguments optionnels sont placés entre crochets):

```
^a:ar68 DRTWX [AV] [F chemin] [OPMOD] ARCHIVE
      OBMOD1 [OBMOD2.....] [> chemin et nom du fichier de sortie]
```

Voici les différentes significations des arguments de la ligne de commande:

DRTWX

Une de ses lettres au moins doit être spécifiée dans la commande. Chacune de ses lettres indique une option particulière dont la description est fournie au paragraphe 1.4.

F chemin

Spécifie le disque et le répertoire où seront stockés les fichiers intermédiaires d'AR68. Si cette option est omise, les disques et répertoires courants sont utilisés.

OPMOD

Indique le module objet interne à la bibliothèque spécifiée. Ce paramètre sert à localiser une position à partir de laquelle devront être incorporés d'autres modules objet (si l'option A est choisi, Cf. 1.4.).

ARCHIVE

Le nom de la bibliothèque sur laquelle opère la commande.

OBMOD1 [OBMOD2.....]

Indique un ou plusieurs modules objet qu'AR68 doit ajouter, supprimer, remplacer, extraire.

>chemin et nom du fichier de sortie

Envoyées par défaut sur l'écran, les sorties d'AR68 peuvent être écrites dans le fichier spécifié.

N.D.T.: Il est également possible d'ajouter les sorties à un fichier déjà existant en utilisant l'option '>>nom du fichier'. Rappelons d'autre part qu'en lançant le programme à partir de 'COMMAND.TOS', le périphérique de sortie peut être modifié. Ainsi: 'a: ligne de commande' >PRN: oriente les sorties vers l'imprimante.

1.3. AR68: Spécifications

AR68 analyse la ligne de commande comme un seul bloc, et non pas séquentiellement.

Lorsqu'AR68 exécute une commande, il crée un fichier temporaire appelé AR68.TEMP, utilisé comme zone de travail. A la fin du processus, ce fichier temporaire est effacé. Si une erreur survient, il se peut cependant qu'AR68 "oublie" d'effectuer cet effacement. Il vous appartient alors de supprimer ce fichier temporaire afin d'éviter toute ambiguïté lors d'un appel ultérieur.

Les messages d'erreur générés par AR68 sont explicités dans l'annexe A.

1.4. AR68: Commandes et options

Cette section décrit les commandes d'AR68 avec leurs différentes options. Les exemples fournis illustrent les effets et les interactions entre chaque commande et les options que chacune d'elle reconnaît.

1.4.1. COMMANDE D

Cette commande supprime un ou plusieurs modules objet dans la librairie spécifiée. Elle accepte l'option suivante:

V Affiche la liste des différents modules de la librairie en indiquant ceux qui sont conservés (noms précédés de la lettre 'c') et ceux qui sont supprimés (noms précédés de la lettre 'd'). La commande qui suit:

```
*a:ar68 dv fichier.lib orc.o
```

pourrait afficher:

```
c red.o
c blue.o
d orc.o
c white.o
```

La commande 'D' détruit le module 'orc.o' appartenant à la librairie 'fichier.lib'. L'option 'V' permet, quant à elle, de lister les différents modules ainsi que leur état.

1.4.2. LA COMMANDE R

Cette commande crée une librairie si le nom du fichier placé dans les arguments de la commande n'existe pas. Si la librairie existe, des modules peuvent y être remplacés ou ajoutés. Un ou plusieurs noms de modules doivent être inclus dans la ligne de commande.

Il est possible de remplacer plusieurs modules en un même appel, avec cependant quelques limitations:

* Si une librairie contient déjà deux ou plusieurs modules possédant le même nom, AR68 ne remplace que le premier module trouvé.

* AR68 remplace les modules déjà existants seulement si vous spécifiez leurs noms AVANT les noms des modules qui doivent être ajoutés. Par exemple, si vous placez dans la commande le nom d'un module que vous désirez remplacer APRES le nom d'un module que vous voulez ajouter, AR68 ajoutera les deux modules à la fin de la librairie.

Par défaut, la commande 'R' ajoute des nouveaux modules à la librairie si l'une des conditions suivantes est remplie:

* Le module objet n'existe pas encore dans la librairie.

* L'option 'A' est spécifiée dans la ligne de commande.

* Le nom du module suit un nom de module qui n'existe pas dans la librairie.

L'option 'A' indique à AR68 que de nouveaux modules sont à ajouter dans la librairie. La position relative à laquelle doit débiter l'ajout est définie par le paramètre OPMOD.

La commande 'R' accepte également l'option 'V' qui affiche les noms des différents modules et la nature des opérations effectuées.

A Ajoute un ou plusieurs modules objet à partir du module spécifié existant déjà dans la librairie.

```
*a:ar68 rav sdav.o fichier.lib work.o mail.o
c much.o
c sdav.o
a work.o
a mail.o
c less.o
```

La commande RAV de l'exemple ci-dessus a ajouté les modules 'work.o' et 'mail.o' derrière le module 'sdav.o' de la librairie 'fichier.lib'. L'option 'V', décrite ci-dessous, a listée les différents modules en précédant de la lettre 'a' les modules nouvellement ajoutés.

V Liste les modules que la commande 'R' remplace ou ajoute.

```
*a:ar68 rv fichier.lib nail.o wrench.o
c saw.o
c ham.o
r nail.o
c screw.o
a wrench.o
```

La commande 'R' a remplacé le module 'nail.o' et ajouté le module 'wrench.o' dans la librairie 'fichier.lib'. L'option 'V' en rend compte en affichant les noms des différents modules précédés de la lettre 'r' s'ils sont nouvellement remplacés, précédés de la lettre 'a' s'ils sont nouvellement ajoutés.

1.4.3. LA COMMANDE T

Cette commande demande à AR68 d'afficher une table décrivant les caractéristiques de tous (ou certains) modules de la librairie spécifiée. La totalité des modules est décrite si la commande ne contient pas un ou plusieurs noms de modules. Cette commande accepte l'option 'V'.

V Affiche la table de description comme dans l'exemple qui suit:

```
*a:ar68 tv fichier.lib
rw-rw-rw- 0/0 6818 rose.o
rw-rw-rw- 0/0 2348 white.o
rw-rw-rw- 0/0 396 red.o
```

La commande 'T' affiche la table des modules de la librairie 'fichier.lib', l'option 'V' a, dans ce cas précis, pour but

d'afficher la taille de chaque module.

La chaîne "rw-rw-rw- 0/0" qui précède le nom de chaque module n'est pas significative dans le système GEMDOS. Son utilité n'est sensible que sous UNIX, puisque cette chaîne indique alors le degré de protection du fichier et son propriétaire.

1.4.4. LA COMMANDE W

Cette commande duplique un module objet contenu dans une librairie et place la copie dans le fichier spécifié. L'argument '>nom de fichier' devient obligatoire.

```
*a:ar68 w fichier.lib now.o >b:\racine\fichier.o
```

Cette commande lit le module 'now.o' de la librairie 'fichier.lib' et le recopie dans le fichier 'fichier.o', dans le répertoire 'racine' du disque B.

NOTE: Les messages d'erreur, s'ils existent, seront eux aussi orientés vers le fichier 'fichier.o'.

N.D.T.: Cette commande ne fonctionne que sur les librairies que vous avez créées. Pour les librairies système, utilisez la commande 'X' décrite ci-dessous.

1.4.5. LA COMMANDE X

Cette commande extrait d'une librairie un ou plusieurs modules objet et les réécrit sur le disque par défaut. Si aucun nom de module n'est spécifié dans la commande, ce sont tous les modules de la librairie qui seront réécrits. La commande 'X' accepte l'option suivante:

V L'option 'V' affiche sur le périphérique de sortie la liste des modules que la commande extrait, et le signale en plaçant la lettre 'x' devant chaque nom.

```
*a:ar68 xv fichier.lib mod1.o mod2.o mod3.o
x mod1.o
x mod2.o
x mod3.o
```

Après exécution, les trois fichiers '.o' de la librairie 'fichier.lib' seront écrits sur le disque par défaut.

1.5. Messages d'erreur d'AR68

Lorsqu'AR68 détecte une erreur pendant une opération, aucun processus n'est encore engagé. Cela signifie que la librairie originale n'est pas modifiée, même si elle devait l'être: aucun module n'est supprimé, ajouté ou remplacé.

Rappelez-vous que les messages d'erreur sont toujours orientés vers le fichier ou périphérique de sortie. Dans le cas

précis de la commande 'W', il est souhaitable d'effectuer une lecture du fichier produit (à l'aide de DUMP ou SID) pour s'assurer qu'aucun message d'erreur n'y est inclus.

2. L'éditeur de commandes: COMMAND.TOS

COMMAND.TOS est un programme vous permettant d'entrer au clavier des commandes du CPM68K. Ces commandes couvrent la quasi totalité des opérations fichiers. Vous disposez donc d'une interface GEMDOS proche de celle du Bureau GEM mais fondée sur des commandes clavier et non sur des manipulations de la souris.

En dehors de la gestion des fichiers, COMMAND.TOS autorise le lancement de programmes et de fichiers ".BAT", c'est-à-dire de fichiers de commandes traitées successivement.

NOTE: Ce chapitre a été entièrement rédigé par les traducteurs, aucune documentation originale n'étant fournie par ATARI Corp. Les informations fournies correspondent à la version 0.35 du 26/6/1985 de ce programme (24041 octets), fournie sur la disquette MASTER2. DEV à la date de la traduction. Ces informations n'engagent que la responsabilité des traducteurs.

2.1. Rappels sur le système CPM

CPM est un système d'exploitation mono-utilisateur conçu par Digital Research pour le microprocesseur Z80. Sa version adaptée au processeur Motorola M68000 est dénommée CPM68k. Ce système ne comprend au sens strict que les programmes destinés au dialogue avec l'utilisateur et à la gestion des fichiers. Des utilitaires standard tels que PIP et ED ont toutefois été ajoutés au système originel et sont généralement perçus comme partie intégrante du CPM.

Dans sa genèse, le ST d'ATARI devait fonctionner à partir du CPM68K. Ce système étant supplanté par l'interface graphique du DESKTOP et la gestion de fichiers GEMDOS, la seule implémentation réelle du CPM68k sur le ST est constituée par COMMAND.TOS. On notera d'ailleurs que les fonctions CPM classiques accessibles en principe à partir du Trap 2 ne sont pas mises en oeuvre dans les versions actuelles du système d'exploitation du ST (voir à ce propos les commandes 'GET' et 'PUT').

2.2. Principes de fonctionnement

2.2.1. MISE EN ROUTE

Pour lancer l'exécution de COMMAND.TOS, cliquez l'icône correspondant à ce nom au bureau GEM. Si vous désirez lancer COMMAND.TOS dès le démarrage du système, renommez-le COMMAND.PRg et mettez-le dans un dossier AUTO sur la disquette placée dans le lecteur A au lancement du système.

Après chargement du programme apparaît le message de copyright puis un signe d'invite ('a:') où la lettre désigne le code du disque par défaut (celui d'où vous avez lancé le programme, voir 3.1.).

NOTE: Lors de la procédure interne d'initialisation de COMMAND.TOS le fichier AUTOEXEC.BAT est chargé et exécuté S'IL EXISTE. Il vous est ainsi possible de lancer un programme ou une suite de commandes automatiquement au lancement de COMMAND.TOS.

2.2.2. SORTIE COMMAND.TOS

Pour sortir de COMMAND.TOS et retourner au programme d'appel, habituellement le bureau GEM, tapez la commande EXIT, laquelle provoque la fermeture de tous les fichiers et termine le programme.

2.2.3. COMMANDES

Ce programme supporte un certain nombre de commandes destinées principalement à la gestion de fichiers. Le détail de chacune de ces commandes, avec des exemples précis, est fourni au paragraphe 2.3 de ce chapitre. Toutes les commandes peuvent être entrées au clavier indifféremment en majuscules ou en minuscules.

2.2.4. EXECUTION DE PROGRAMMES

Il vous est possible, à partir de COMMAND.TOS, de lancer un ou plusieurs programmes successivement. Ainsi la commande SID.TOS provoquera le chargement et l'exécution du programme SID. Lorsque vous sortirez de SID, le retour s'effectuera dans COMMAND.TOS, vous permettant d'appeler un nouveau programme, etc.

Vous pouvez également lancer l'exécution de fichiers de commandes, dits fichiers "batch" (traitement par lots) et pourvus de l'appendice ".BAT". Ces fichiers peuvent comporter des noms de commandes du COMMAND.TOS ou/et des noms de programmes à exécuter (voir BATCH.TTP et les fichiers ".BAT" fournis avec ce système).

NOTE:

Si vous désirez faire exécuter un programme ou un fichier ".BAT", vous pouvez vous dispenser de frapper au clavier les lettres de l'appendice (".PRG" ou ".BAT"), mais il faut savoir que pour un nom donné, par exemple "XYZ", COMMAND.TOS tentera d'abord d'exécuter la COMMANDE "XYZ" (inexistante dans la liste des commandes, voir à 2.3), puis tentera de charger et d'exécuter le FICHIER "XYZ.BAT" et, si ce fichier n'existe pas, tentera de charger et d'exécuter le PROGRAMME "XYZ.PRG". Les appendices ".TOS", ".TTP", ".EXE" et ".APP" ne sont pas reconnus par COMMAND.TOS et devront donc être spécifiés dans leur totalité. Ainsi l'appel de SID se fera par l'entrée au clavier de "SID.TOS".

2.2.5. CARACTERES GENERIQUES

Sont appelés "caractères génériques" les symboles '*' et '?', lesquels permettent de remplacer des caractères dans un nom de fichier. Ainsi "COMPIL???.C" permettra de désigner aussi bien le fichier COMPIL.C que les fichiers COMPIL_1.C ou COMPILER.C, le caractère générique '?' pouvant remplacer un code quelconque.

Plus puissant, le caractère '*' permet de remplacer une chaîne de caractères. Ainsi "*.S" représentera l'ensemble des fichiers d'une disquette, le premier symbole "*" recouvrant l'ensemble des noms de fichiers possibles et le second recouvrant l'ensemble des appendices possibles. Par exemple, "*.S" désignera l'ensemble des fichiers assembleur de la disquette.

Plusieurs fonctions de COMMAND.TOS autorisent l'utilisation de ces symboles génériques. Chaque cas est traité en section 2.3.

2.2.6. REDIRECTION

Les versions classiques du CPM autorisaient la redirection des entrées clavier et des sorties écran vers d'autres périphériques ou des fichiers par le biais d'une redirection par "CON:", "LST:", "PUN:", etc. Sous CPM68k, la même possibilité vous est offerte mais selon un protocole légèrement différent.

Le CPM68k permet de rediriger les fichiers standard 'stdin' et 'stdout'. Le premier correspond au canal d'entrée des données, c'est-à-dire le clavier. Le second correspond au canal de sortie des données, c'est-à-dire l'écran.

'stdin' est symbolisé par '<', 'stdout' est symbolisé par '>'. La redirection de ces fichiers standard s'effectue par:

```
<nom_de_fichier  pour le canal d'entrée de données
>nom_de_fichier  pour le canal de sortie de données
```

où 'nom_de_fichier' peut être:

CON: pour désigner le clavier ou l'écran (sans intérêt ici),
 PRN: pour désigner l'imprimante (sortie parallèle),
 AUX: pour désigner la sortie série.
 'nom_fichier' où 'nom_fichier' désigne un fichier sur disque, par exemple 'a:sortie.doc' ou 'commande.inf', etc.

Cette demande de redirection n'est valide que pour la ligne de commande à laquelle elle appartient. Si l'on désire rediriger systématiquement toutes les sorties vers un périphérique ou un fichier, il faut ruser! Ainsi pour rediriger toutes les sorties du COMMAND.TOS vers l'imprimante, on entrera la commande:

```
>prn: command.tos
```

qui lancera COMMAND.TOS à partir de lui-même et redirigera toutes les sorties vers l'imprimante!

2.2.7. PASSAGE DE PARAMETRES

Quelques commandes CPM68K nécessitent des paramètres. Les appels de programmes nécessitent aussi souvent le passage de paramètres. D'une façon générale, le nom de la commande et chacun des paramètres doivent être séparés par un espace. Cet espace est interprété par COMMAND.TOS comme un séparateur de paramètres. Si vous aviez besoin dans un programme quelconque d'entrer un paramètre contenant un ou plusieurs espaces, vous pouvez placer le paramètre entier entre guillemets, auquel cas le ou les espaces ne seront pas traités comme des séparateurs, mais les guillemets seront passés avec le paramètre à votre programme.

Il est également possible de passer des paramètres par l'intermédiaire d'un fichier, ce fichier contenant la liste de vos paramètres. Cet appel se fait par:

@nom_de_fichier

où 'nom_de_fichier' est le nom du fichier de paramètres.

Exemple: Soit le fichier 'disques.par' dont le contenu est:

A:\MASTER2.DEV\

A:\MASTER2.DEV\SHELL.UNX\

La commande 'dir @disque.par' (voir commande dir en section 2.3) provoquera l'affichage des composantes des fichiers du catalogue 'MASTER2.DEV' puis du catalogue 'SHELL.UNX' appartenant au catalogue précédent.

Le passage de paramètres par fichier est naturellement compatible avec la redirection. Ainsi la commande:

>prn: dir @disque.par

provoquera l'impression du catalogue MASTER2.DEV puis du sous-catalogue SHELL.UNX.

2.3. Commandes

Conventions: Les noms de paramètres indispensables seront entourés par les signes '<' et '>'. Les noms de paramètres optionnels seront encadrés par des crochets. Les noms de commandes seront toujours fournis en majuscules bien qu'ils puissent être entrés en minuscules.

2.3.1. CHANGEMENT DU DISQUE PAR DEFAULT

Le disque par défaut correspond lors du lancement* de COMMAND.TOS au disque d'où a été lancé le programme. Vous pouvez changer ce disque simplement en entrant au clavier la lettre du nouveau disque puis le signe ':' et un retour chariot.

Exemple: Supposons que le disque par défaut soit le disque A, le signe d'invite est donc 'a:'. Entrez 'b:' puis pressez Return. Le signe d'invite devient 'b:' et le disque B est devenu disque par défaut.

2.3.2. ENV : DEMANDE L'ENVIRONNEMENT

Cette commande fournit l'environnement de lancement du programme COMMAND.TOS. Si vous avez lancé ce programme à partir du bureau GEM et quelque soit le disque de lancement, la chaîne d'environnement est: A:\

Si un chemin a été fixé par la commande PATH (cf.3.3), ce chemin est affiché sous la forme: PATH=<chemin>

2.3.3. PATH : FIXE UN CHEMIN

Cette commande permet de demander ou de fixer un chemin d'accès par défaut, valide pour les appels de PROGRAMMES ou de fichiers batch (.BAT) ultérieurs. Ne confondez pas cette commande avec CD (voir 3.4).

Cette commande peut prendre trois formes:

PATH	demande le chemin actuel,
PATH ;	fixe le chemin actuel au catalogue racine,
PATH <chemin>	fixe <chemin> comme chemin d'accès courant.

Le chemin fixé par la commande PATH constituera la chaîne d'environnement lors de l'appel de programmes à partir de COMMAND.TOS.

2.3.4. CD : FIXE LE CHEMIN COURANT

CD demande ou fixe le chemin d'accès courant de COMMAND.TOS, c'est-à-dire le chemin à partir duquel il listera les fichiers (dir) ou à partir duquel il recherchera un fichier.

Cette commande peut prendre deux formes:

CD	demande le chemin actif courant,
CD <chemin>	fixe <chemin> comme chemin actif courant.

Pour fixer un chemin correspondant à la racine du catalogue (le niveau 0 du disque), on utilisera la commande: CD \

Exemples:

*a:cd	demande le chemin courant.
\MASTER2.DEV	affiche le chemin courant.
*a:cd SHELL.UNX	fixe MASTER2.DEV\SHELL.UNX comme chemin
*a:cd	demande le nouveau chemin courant
\MASTER2.DEV\SHELL.UNX	

2.3.5. LS ou DIR : AFFICHAGE DE CATALOGUE

Les commandes LS et DIR, strictement équivalentes, permettent d'obtenir le catalogue des fichiers du chemin courant (défini par

CD). Ces commandes peuvent prendre quatre formes:

```
DIR [num_disque:]    fournit les fichiers et les dossiers,
DIR [num_disque:] -F fournit les fichiers, pas les dossiers,
DIR [num_disque:] -D fournit les dossiers, pas les fichiers,
DIR [num_disque:] -T fournit uniquement les noms de fichiers.
```

Le paramètre [num_disque:] est optionnel, il permet de demander le catalogue d'un autre disque. On notera que si l'on se trouve dans un dossier, il ne sert à rien de placer [num_disque:] pour obtenir le catalogue de tout le disque, il faut revenir au niveau racine par la commande CD.

Les trois premières formulations de DIR fournissent nombre d'informations sur chacun des fichiers ou dossiers: son nom et son appendice, sa date de création, son heure de création, le nombre d'octets qu'il contient et son type. Par contre la dernière option ne provoque l'affichage que des noms.

Exemple: >PRN: DIR a:
imprimera la liste complète des fichiers et dossiers du disque A, si l'on se trouve au niveau racine du disque (ce cas de figure fournit un exemple intéressant de redirection).

2.3.6. CAT ou TYPE : AFFICHAGE D'UN FICHIER

Les commandes CAT et TYPE sont identiques. Elles provoquent la sortie sur 'stdout' (l'écran s'il n'y a pas de redirection) du fichier désigné après la commande (ces commandes n'ont d'intérêt que pour les fichiers ASCII):

```
CAT <nom_de_fichier> affiche le contenu du fichier désigné
                        par <nom_de_fichier>
```

Cette commande peut être utilisée avec les caractères génériques, ce qui permet l'affichage de fichiers CONCATENES. Ceci est particulièrement utile pour concaténer deux ou plusieurs fichiers en un seul avec une redirection.

Exemple: >groupe.s CAT *.S
provoquera une concaténation de tous les fichiers à appendice '.S' et sauvera le produit de cette concaténation dans le fichier GROUPE.S.

Note: Cette commande ne peut être employée avec un nom de dossier.

2.3.7. CHMOD : MODIFICATION DU TYPE D'UN FICHIER

La commande CHMOD permet de changer le type d'un fichier. Sa syntaxe d'appel est la suivante:

```
CHMOD <nom_de_fichier> <type>
    où <nom_de_fichier> désigne le fichier,
    et <type> le nouveau type de ce fichier.
```

Seuls les trois bits faibles du type peuvent être modifiés par cette commande. Ces bits ont la signification suivante:

```
bit 0: à 1 si fichier protégé en écriture (lecture seule),
bit 1: à 1 si fichier "caché",
bit 2: à 1 si fichier système.
```

NOTE: Cette commande ne peut être utilisée avec un nom de dossier.

2.3.8. RM ou DEL ou ERA : EFFACEMENT DE FICHIER

Les commandes RM, DEL et ERA sont équivalentes et permettent d'effacer un fichier (la commande RM n'a rien à voir avec RM.PRg).

```
RM <nom_de_fichier>
    où <nom_de_fichier> est le nom de fichier à effacer.
```

Cette commande autorise les caractères génériques. Elle n'est utilisable qu'avec les fichiers (voir RD pour les dossiers).

Exemples: RM COMPIL.C
efface le fichier compile.c
RM COMPIL.?
efface le fichier compile.c mais aussi compile.s s'il existe et tous les fichiers 'compile' ayant une lettre en appendice.
RM *.BAK
efface tous les fichiers du niveau courant ayant l'appendice '.BAK'

2.3.9. REN : RENOMME UN FICHIER

```
REN <nom1_fichier> <nom2_fichier>
    renomme le fichier <nom1_fichier> en <nom2_fichier>.
```

2.3.10. COPY : COPIE UN FICHIER

```
COPY <nom1_fichier> <nom2_fichier>
```

recopie le fichier <nom1_fichier> en un fichier <nom2_fichier>. Le fichier original reste INTACT.

Cette commande autorise les caractères génériques dans le nom source mais, dans ce cas, le nom de la destination doit être '*. *' autrement dit doit être sur un autre disque.

Exemples: COPY FICHE.DOC FICHE.BAK
crée un fichier 'fiche.bak' équivalent à 'fiche.doc'.
COPY A:*.S B:*. *
copie tous les fichiers à appendice '.S' du disque A vers B.
COPY A:*. * B:*. *
copie tous les fichiers du disque A sur le disque B.

2.3.11. MOVE : COPIE UN FICHIER ET EFFACE L'ORIGINAL

MOVE <nom1_fichier> <nom2_fichier>

recopie le fichier <nom1_fichier> en un fichier <nom2_fichier>. Le fichier original est DETRUIT.

Cette commande autorise les caractères génériques dans le nom source mais, dans ce cas, le nom de la destination doit être '*. *' autrement dit doit être sur un autre disque. D'une certaine façon, cette commande est semblable à REN lorsque les disques de départ et d'arrivée sont identiques.

Exemples: MOVE A:FICHE.DOC B:FICHE.BAK
crée un fichier 'b:fiche.bak' équivalent à 'a:fiche.doc' et efface le fichier 'fiche.doc'.

MOVE A:*.S B:*. *
copie tous les fichiers d'appendice '.S' du disque A vers B.

2.3.12. MD : CREE UN DOSSIER

MD <nom_de_dossier> crée le dossier <nom_de_dossier>.

2.3.13. RD : SUPPRIME UN DOSSIER

RD <nom_de_dossier> supprime le dossier <nom_de_dossier>.

Comme avec Ddelete (GEMDOS), la suppression d'un dossier comportant des fichiers est impossible. Il faut d'abord effacer les fichiers du dossier puis supprimer le dossier.

2.3.14. SHOW : INFORMATIONS SUR UN DISQUE

SHOW [num_disque:]

fournit des informations sur le disque par défaut ou le disque [num_disque:] si ce paramètre optionnel est entré.

Exemple:

*a:show b:

Allocation Information: Drive B:

Total Units on Drive	711
Free Units on Drive	388
Sectors per Unit	2
Bytes per Sector	512

Les informations fournies correspondent à:

Nombre de blocs du disque
Nombre de blocs disponibles du disque
Nombre de secteurs par bloc
Nombre d'octets par secteur

2.3.15. INIT : REINITIALISATION DES FATs ET DES CATALOGUES

INIT [num_disque:]

remet à 0 tous les secteurs des tables d'allocation et du catalogue racine du disque par défaut ou de [num_disque:] si ce paramètre optionnel est fourni. Pratiquement, cela équivaut à FORMATER à nouveau la disquette, le contenu des fichiers restant sur les secteurs mais n'étant plus accessible. ATTENTION donc à l'emploi de cette commande!

2.3.16. PUTBOOT : IMPLANTATION D'UN SECTEUR BOOT

PUTBOOT [num_disque:] <nom_de_fichier>

charge le fichier <nom_de_fichier> (paramètre nécessaire) et place les octets 28 à 539 de ce fichier sur le secteur boot du disque par défaut ou de [num_disque:] si ce paramètre optionnel est fourni. ATTENTION, si le fichier ne contient pas un secteur boot, le disque sur lequel il est copié ne sera plus accessible!

2.3.17. GET : LECTURE DIRECTE DE FICHIER CPM

GET <nom_fich_cpm> <nom_fichier>

charge le contenu du fichier cpm <nom_fich_cpm> et le place dans le fichier désigné par <nom_fichier>.

NOTE: Cette commande fait appel à une routine cpm de lecture de fichier via le trap 2. Malheureusement cette routine n'est pas implantée en ROM. Aussi cette fonction écrit-elle 0 octet sur le fichier <nom_fich_cpm>!

2.3.18. PUT : ECRITURE DIRECTE DE FICHIER CPM

PUT <nom_fichier> <nom_fich_cpm>

charge le contenu du fichier <nom_fichier> et le place dans le fichier cpm <nom_fich_cpm>.

NOTE: Commande inutilisable pour la raison explicitée ci-dessus (voir 3.17).

2.3.19. EXIT : FIN DE COMMAND.TOS

La commande EXIT permet de quitter COMMAND.TOS.

2.3.20. BREAK : APPEL DE L'INSTRUCTION 'ILLEGAL'

La commande BREAK correspond à un appel de l'instruction 68000 ILLEGAL. Etant donné qu'il s'agit d'une exception, cela provoque habituellement l'apparition de 4 petites bombes dans le milieu gauche de l'écran!

Cette commande peut être utile lorsqu'on a lancé `COMMAND.TOS` à partir du `SID` (voir chapitre 6), l'instruction `ILLEGAL` provoquant un retour au `SID`.

2.3.21. REM ou ECHO : ENTREE DE COMMENTAIRES

Les commandes `REM` et `ECHO` sont identiques. Elles permettent d'entrer des commentaires sur une ligne de commande. Ces commentaires peuvent être placés entre guillemets.

```
REM ["commentaires"]
```

Exemple:

```
REM "commande REM"
```

2.3.22. PAUSE : ATTENTE D'ENTREE CLAVIER

La commande `PAUSE` provoque une attente d'appui de la touche `Return` ou de l'entrée de 126 caractères. Elle se traduit par l'affichage du message "CR to continue".

2.3.23. HELP : RESUME DES COMMANDES

La commande `HELP` provoque l'affichage d'un résumé du mode d'emploi des commandes, en anglais. Utiliser `Control-S (^S)` et `Control-Q (^Q)` pour figer et reprendre l'affichage.

2.3.24. ERR : CODE ERREUR

La commande `ERR` provoque l'affichage du code d'erreur correspondant à la dernière commande entrée. Ce code est nul si la dernière commande n'a pas engendrée d'erreur.

2.3.25. VERSION : NUMERO DE VERSION DU GEMDOS

La commande `VERSION` fournit le numéro de version du `GEMDOS` (gestion de fichiers en ROM). Ce numéro est égal à '0.19' pour les ROMs de la date du 24/4/86 au 27/4/87. Il est égal à '0.20' si vous avez implanté l'accélérateur de gestion de fichiers 'TURBODOS' sur votre système.

2.3.26. CLS : EFFACEMENT DE L'ECRAN

La commande `CLS` provoque un effacement de l'écran et replace le curseur en première rangée et première colonne.

2.3.27. WRAP : ACTIVE LE RETOUR DE LIGNE AUTOMATIQUE

La commande `WRAP` active le retour de ligne automatique. Ainsi après l'entrée du dernier caractère d'une ligne, le curseur passe en ligne suivante.

2.3.28. NOWRAP : DESACTIVE LE RETOUR DE LIGNE AUTOMATIQUE

Voir la commande `WRAP` (3.27). En bout de ligne, tous les caractères entrés se superposent sur le dernier caractère. Ils sont toutefois pris en compte dans la commande. Mode par `DEFAULT`.

3. L'utilitaire de traitement par lots: BATCH

`BATCH` exécute une séquence de commandes définie dans un fichier au suffixe `'.bat'`. Se lançant du Bureau `GEM`, il évite d'entrer au préalable dans le programme `'COMMAND.TOS'` pour effectuer un processus de commandes multiples.

3.1. Convention d'appel

`BATCH` doit être installé comme un programme "TOS avec paramètres". Les arguments sont placés dans la boîte de dialogue qui s'ouvre lors du lancement. La ligne de commande a le format suivant:

```
nom du fichier.bat [argument1 argument2 ...]
```

L'existence d'arguments, leur nombre et leur forme, dépendent évidemment du fichier `'.bat'` appelé.

3.2 Appel sans argument

Considérons le fichier `'monbat.bat'` dont le contenu serait:

```
b:doodle.prg
a:sid.prg
```

Cette séquence de commande n'attend aucun argument du programme `BATCH` dont la ligne de commande serait:

```
monbat.bat
```

NOTE: Le suffixe est optionnel lorsqu'il est de la forme `'.bat'`.

Dans cet exemple, le programme `'doodle'` est lu sur le disque `B` et exécuté. Au retour de ce programme, `'sid'` est lu sur le disque `A` et exécuté.

3.3. Appel avec argument(s)

Lorsqu'une séquence de commandes nécessite des paramètres, ces derniers sont indiqués par un chiffre (de 0 à 9) précédé du caractère `'%'`. `BATCH` remplace le chiffre le plus faible par le premier argument de sa ligne de commande, le chiffre juste supérieur par le second argument, etc...

Ainsi le fichier 'taille.bat' se présentant sous la forme:

```
size68 %1.prg >%2.lst
rm %3
nm68 %1.prg >>%2.lst
```

L'utilitaire BATCH attend au moins trois arguments pour substituer les signes '%1', '%2' et '%3'. L'appel pourrait se présenter ainsi:

```
taille truc sortie essai.prg
```

Le fichier 'taille.bat' s'exécuterait alors de la manière suivante:

```
size68 truc.prg >sortie.lst
rm essai.prg
nm68 truc.prg >>sortie.lst
```

NOTE IMPORTANTE:

Nous tenons à mettre en garde contre une erreur fréquente (d'inattention ou de méconnaissance) et qui entraîne souvent des conséquences funestes. Prenons l'exemple du fichier 'C.BAT' fourni avec la disquette de compilation. Les premières lignes se présentent comme suit:

```
cp68 %1.c %1.1
c068 %1.1 %1.1 %1.2 %1.3 -f
rm %1.1
cl68 %1.1 %1.2 %1.3
```

Le fichier 'C.BAT' a besoin d'un unique argument représentant le fichier source C à compiler. Mais le nom de ce source doit être fourni SANS SON SUFFIXE. Imaginons ce qu'il advient si un suffixe est donné:

```
c.bat essai.c
```

Le fichier 'C.BAT' prend alors la forme suivante:

```
cp68 essai.c.c essai.c.1
c068 essai.c.1 essai.c.1 essai.c.2 essai.c.3 -f
rm essai.c.1
.....
```

Aucun des programmes de cette liste n'est gêné par le double suffixe: il ne prend en compte que le premier. Le déroulement des événements est le suivant:

- * le pré-processeur CP68 considère 'essai.c' comme son fichier d'entrée et réécrit 'essai.c' comme fichier résultat!
- * C068 lit 'essai.c' et le réécrit trois fois, puis efface son fichier intermédiaire qui devrait être 'essai.3' mais est en fait 'essai.c' !!

* RM tente d'effacer le fichier 'essai.c' et affiche un premier message d'erreur!?!

Evidemment, votre fichier source est perdu corps et biens et vous vous maudissez d'autant plus de votre faute d'inattention que, bien entendu, la dernière copie de sauvegarde de votre source date de trois jours!

3.4. BATCH: Interrompre la commande

BATCH teste les appuis de touches et affiche une invitation à clore le processus par le message 'ABORT?(y/n)'. L'appui de la touche 'y' stoppe la séquence de commande, toute autre touche est considérée comme un désir de continuation.

L'appui de touche est évidemment testé quand BATCH contrôle le processus et non pas lorsqu'un programme spécifique s'exécute. Entre chaque programme, BATCH reprend la main et teste le tampon clavier. Si un caractère est disponible dans le tampon, BATCH affiche l'invite d'arrêt et attend un caractère.

N.D.T.: Méfiez vous d'appuis répétés au clavier: peu de programmes pensent à vider le tampon clavier. En prenant l'exemple du fichier '.bat' du paragraphe précédent et en imaginant qu'une erreur s'est produite dans 'CP68' vous obligeant à arrêter le processus de compilation: vous appuyez la barre <espace> un peu trop longuement et l'autorépétition des touches provoque le stockage dans le tampon clavier de deux caractères. Avant de lancer 'C068', BATCH lit ce tampon. Le premier caractère <espace> entraîne l'affichage de la demande d'arrêt, le second caractère <espace> est considéré comme un ordre de continuation. Pour pallier cet inconvénient, pressez uniquement la touche 'y' si vous désirez stopper un processus, et toujours avec parcimonie.

4. L'utilitaire d'affichage: DUMP

DUMP affiche le contenu d'un fichier (binaire, ASCII, exécutable) en notations hexadécimale et ASCII.

4.1. Conventions d'appel

La commande d'appel de DUMP a le format suivant:

```
*a:dump [ -SXXXX] nom de fichier [>fichier de sortie]
```

Les différents arguments ont la signification suivante:

-SXXXX

XXXX est une valeur hexadécimale représentant le décalage à effectuer avant de débiter la lecture du fichier. Si cette option est spécifiée, DUMP commence à afficher le contenu XXXX octets

plus loin que le début du fichier. Par défaut, DUMP débute l'affichage en début de fichier.

nom de fichier

Le nom du fichier à lire.

>fichier de sortie

Le fichier vers lequel doit s'effectuer la sortie. Si cette option n'est pas précisée, la sortie se fait à l'écran.

N.D.T.: Il est également possible d'ajouter les sorties à un fichier déjà existant en utilisant l'option '>>nom du fichier'. Rappelons d'autre part qu'en lançant le programme à partir de 'COMMAND.TOS', le périphérique de sortie peut être modifié. Ainsi: 'a:' ligne de commande' >PRN: oriente les sorties vers l'imprimante.

4.2. Format de sortie

DUMP affiche un contenu de fichier par ligne de 16 octets, en respectant le format suivant:

```
RR OO (FFFFFF): HHHHHHHHHHHHHHHH *AAAAAAAAAAAAAAAA*
```

Les significations des différents éléments sont les suivantes:

RR Numéro d'enregistrement de la ligne courante (un enregistrement correspond à 128 octets).

OO Décalage (en hexadécimal) depuis le début de l'enregistrement courant.

FFFFFF Décalage (en hexadécimal) depuis le début du fichier.

HH..HH Contenu du fichier en hexadécimal.

AA..AA Contenu du fichier en ASCII. Si le caractère n'est pas représentable, un point ('.') est affiché.

5. L'utilitaire SIZE68

SIZE68 donne un certain nombre de renseignements sur un fichier programme. Il fournit:

- * le nombre de segments d'un programme, leurs tailles respectives, leurs emplacements relatifs (contigus ou non-contigus),
- * la taille de la table des symboles, si elle existe,
- * l'état de relogeabilité du programme.

Les valeurs numériques sont fournies aussi bien en décimal qu'en hexadécimal.

5.1. Conventions d'appel

La commande d'appel de SIZE68 a le format suivant:

```
*a:size68 nom de fichier1 [nom de fichier2...][>fichier de sortie]
```

Les différents arguments de la commande ont les significations suivantes:

nom de fichier [....]

Le(s) nom(s) du(des) fichier(s) dont vous désirez les spécifications. En cas de fichiers multiples, SIZE68 affiche les caractéristiques de chaque fichier l'un après l'autre. La taille d'une ligne de commande ne peut excéder 128 caractères, mais il est possible de placer des symboles génériques (?,*) dans les noms de fichier.

>fichier de sortie

Le fichier vers lequel doit s'effectuer la sortie. Si cette option n'est pas spécifiée, les résultats sont affichés à l'écran.

N.D.T.: Il est également possible d'ajouter les sorties à un fichier déjà existant en utilisant l'option '>>nom du fichier'. Rappelons d'autre part qu'en lançant le programme à partir de 'COMMAND.TOS', le périphérique de sortie peut être modifié. Ainsi: 'a:' ligne de commande' >PRN: oriente les sorties vers l'imprimante.

5.2. Exemples de commandes

La commande suivante renvoie des informations sur le programme SIZE68.PRG.

```
*a:size68 size68.prg
```

```
SIZE68.PRG
```

Contiguous		
.text length	=	9510 2460
.data length	=	1220 4C4
.bss length	=	9140 23B4
Symbol table length	=	1484 5CC
Start of .text	=	0 0
File is relocatable.		

Le programme SIZE68.PRG est un programme relogeable et à segments contigus. La longueur de sa zone des programmes (.text) est de 9510 octets, de sa zone des données initialisées (.data) 1220 octets, et de sa zone des données non-initialisées 9140 octets.

La commande suivante utilise des "jokers" pour retourner des informations sur un ou plusieurs programmes:

```
*a:size68 b:FIC*.*
```

FICHER.O:

```
Contiguous
.text length      =    1072      430
.data length      =     188      BC
.bss length       =      0       0
Symbol table length =    1708     6AC
Start of .text    =      0       0
File is relocatable.
```

FICHER.PRГ:

```
Contiguous
.text length      =    9888     26A0
.data length      =    1060     424
.bss length       =    9396     24B4
Symbol table length =      0      0
Start of .text    =      0      0
File is relocatable.
```

FICHE.MSG:

Not a programm file.

Notez que lorsque vous spécifiez un fichier qui n'est pas un programme (FICHE.MSG, un fichier ASCII par exemple), SIZE68 envoie le message "Not a programm file".

6. L'utilitaire de format S-RECORD: SEND

SEND crée un fichier au format "MOTOROLA S-record" à partir d'un fichier programme avec adresses absolues. Le format "S-record" est une méthode de représentation d'un programme en une forme ASCII (Cf. à 6.2.).

6.1. Conventions d'appel

La commande d'appel de SEND a le format suivant:

```
*a:send [-] nom de fichier [fichier de sortie]
```

Les différents arguments ont la signification suivante:

- (signe moins)

Le signe moins ('-') est optionnel. S'il est spécifié, SEND ne crée pas de format "S-record" pour la zone des données non-initialisées (BSS).

nom de fichier

Le nom du fichier que SEND doit convertir. Ce fichier ne doit pas être relogeable et doit correspondre à un fichier produit par L068 ou LINK68 (c'est à dire avec suffixe '.68K').

fichier de sortie

Le nom du fichier au format "S-record". Si ce nom n'est pas spécifié, le fichier créé est affiché à l'écran.

6.2. Le format "S-record" de MOTOROLA

Le format "S-record" est une méthode pour représenter des données binaires en format ASCII. Son utilité principale est de permettre de transporter des programmes d'un ordinateur à un autre. Dans la mesure où les codes ASCII sont une représentation de l'information très répandue, l'utilisation de ce format s'avère être une méthode efficace.

6.2.1. FORMAT D'UN ENREGISTREMENT

Un fichier au format "S-record" consiste en une séquence d'enregistrements de types variables, chaque code étant représenté sous une forme ASCII. Quand un nombre hexadécimal doit être converti dans ce format, chaque chiffre est transformé en un caractère ASCII correspondant. Chaque enregistrement est composé de 5 champs comme indiqués ci-dessous:

S	TYPE	LONGUEUR	ADRESSE	DONNEE	SOMME
1 oct.	1 oct.	2 oct.	2,4,6 oct.	variable	2 oct.

La ligne inférieure donne la largeur du champ en octets.

6.2.2. DESIGNATION DES CHAMPS

La signification des différents champs est la suivante:

S

Le caractère ASCII 'S' signalant le début d'un enregistrement.

TYPE

Un chiffre ASCII entre 0 et 9 (à l'exclusion de 4 et 6), représentant le type de l'enregistrement (Cf.6.2.3.).

LONGUEUR

Le nombre de caractères d'ordre pair dans l'enregistrement, sans compter les 3 premiers champs. Cette valeur indique la moitié du nombre total de caractères des champs 'ADRESSE', 'DONNEE' et

'SOMME'. Elle se présente sous la forme de deux codes ASCII représentant une valeur hexadécimale.

ADRESSE

L'adresse à laquelle doivent être stockées les données de l'enregistrement. La largeur de ce champ dépend de TYPE.

DONNEE

Un champ de largeur variable contenant la donnée à placer en mémoire. Chaque octet est codé par deux codes ASCII représentant une valeur hexadécimale.

SOMME

Une somme de contrôle ("checksum") calculée à partir des champs 'LONGUEUR', 'ADRESSE', 'DONNEE'. Le calcul se fait de façon suivante:

- * additionner les valeurs de caractères d'ordre pair des trois champs pré-cités.
- * prendre le complément à un de la somme obtenue.
- * abandonner l'octet fort de la somme.
- * placer l'octet faible dans le champ sous forme de deux codes ASCII représentant une valeur hexadécimale.

6.2.3. LES TYPES D'ENREGISTREMENTS

Il existe 8 types d'enregistrements. Ceux-ci peuvent être divisés en deux catégories:

- * les enregistrements contenant des données (types 1,2,3).
- * les enregistrements délimiteurs (types 0,5,7,8,9).

Les types 4 et 6, rappelons-le, ne sont pas valides.

TYPE 0

Indique un enregistrement d'en-tête utilisé au début de chaque bloc d'enregistrement. le champ DONNEE peut contenir une information supplémentaire. Le champ ADRESSE comporte deux octets normalement à zéro.

TYPE 1

Indique un enregistrement contenant des données, le champ ADRESSE comporte deux octets.

TYPE 2

Idem que TYPE 1, avec un champ ADRESSE de trois octets.

TYPE 3

Idem que TYPE 1, avec un champ ADRESSE de quatre octets.

TYPE 5

Indique le nombre d'enregistrements de TYPE 1, 2, ou 3 dans un groupe d'enregistrements. Le nombre est placé dans le champ ADRESSE, le champ DONNEE est vide.

TYPE 7

Indique la fin d'un groupe de TYPE 3. Le champ ADRESSE de quatre octets peut éventuellement contenir l'adresse d'un programme prenant le contrôle du processus. Le champ DONNEE est vide.

TYPE 8

Idem que TYPE 7, mais indique la fin d'un groupe de TYPE 2.

TYPE 9

Idem que TYPE 7, mais indique la fin d'un groupe de TYPE 1.

6.3. Exemples de commandes

La ligne de commande fournie en exemple convertit le fichier 'prog.68k' en un fichier au format "S-record" nommé 'prog.sr'. La zone 'BSS' n'est pas convertie.

```
*a:send - prog.68k prog.sr
```

7. L'utilitaire d'affichage de symboles: NM68

NM68 affiche à l'écran, ou écrit dans un fichier, le contenu de la table des symboles (lorsqu'elle existe) d'un fichier objet ou exécutable ('.o' ou '.prg') et fournit des informations sur la localisation de chaque symbole.

7.1. Conventions d'appel

La commande d'appel de NM68 a le format suivant:

```
*a:nm68 nom de fichier [>chemin et nom du fichier de sortie]
```

Les différents arguments ont la signification suivante:

nom de fichier

Le nom du fichier dont la table des symboles doit être affichée. Si la table des symboles n'existe pas, NM68 revient au programme appelant sans aucun affichage.

>chemin et nom du fichier de sortie

Le nom du fichier vers lequel doit s'effectuer la sortie des résultats. Si cette option est omise, la table des symboles est affichée à l'écran.

N.D.T.: Il est également possible d'ajouter les sorties à un fichier déjà existant en utilisant l'option '>>nom du fichier'. Rappelons d'autre part qu'en lançant le programme à partir de 'COMMAND.TOS', le périphérique de sortie peut être modifié. Ainsi:

```
*a:'ligne de commande' >PRN:
oriente les sorties vers l'imprimante.
```

7.2. Description de la table des symboles

La table des symboles contient tous les symboles d'un programme, chaque symbole étant codé sur 7 mots.

Les 4 premiers mots forment le nom ASCII du symbole, complétés par zéro si le nom a moins de 8 caractères.

N.D.T.: Après édition de liens, le premier caractère est *TOUJOURS* le trait de soulignement s'il s'agit d'un symbole 'global', le caractère tilde s'il s'agit d'un symbole 'local'. (Les vocables 'global' et 'local' sont pris ici dans le sens 'C' du terme).

Le cinquième mot représente le type du symbole:

* absolu	(abs)	\$8000
* égal	(equated)	\$4000
* égal registre	(equated register)	\$1000
* référence externe	(external reference)	\$ 800
* zone 'data' relogeable	(data based relocatable)	\$ 400
* zone 'text' relogeable	(text based relocatable)	\$ 200
* zone 'bss' relogeable	(bss based relocatable)	\$ 100

Lorsqu'un symbole possède des caractéristiques multiples, l'éditeur de liens utilise l'instruction OU LOGIQUE. Par exemple, un symbole défini, global et appartenant à la zone 'data' relogeable aura la valeur \$A400.

Les sixième et septième mots forment un long mot contenant la valeur du symbole. Cette valeur peut être une adresse, un numéro de registre, la valeur d'une expression ou toute autre valeur.

7.3. Exemple de commande

Voici un exemple de programme écrit en C, compilé puis "lié" avec LINK68 (en demandant à ce dernier l'intégration des symboles locaux par l'option 'L'), et enfin transformé par RELMOD. Le programme exécutable produit se nomme "essai.prg".

/ ESSAI */*

```
int entier;
int valent = 0x100;
char chail[10];
char chai2[] = "coucou";
float valf;
```

```
main()
{
    register int regl;
    int vale2 = 10;
    char carac;
    printf ("essai\n");
}
```

La ligne de commande de LINK68 est la suivante:

```
link68 [tem[c:],u,s,l] %1.68k=gemstart,%1,gemlib,libf
```

Ne sont donnés ici que les symboles nous intéressant directement, sans fournir ceux, innombrables, des bibliothèques 'GEMLIB' et 'LIBF'. Les commentaires sont rajoutés par nous.

nm68 essai.prg

```
.....
_chai2      27E6  global data
_valent     27EE  global data
_main       FA    global text    /* le 'main' de GEMSTART */
_regl       7     equ reg abs
_vale2      FFFFFFFE equ abs
_carac      FFFFFFFC equ abs
_main       11E   global text    /* le 'main' de notre prog. */
_chail      2B36  global bss
_valf       2B40  global bss
_entier     2B44  global bss
.....
```

Tous les symboles globaux ont comme valeur une adresse de décalage (offset) par rapport au début du programme. Pour les symboles 'equ', l'adresse est absolue mais relativement au pointeur de pile. Le symbole 'regl' est placé dans le registre D7.

8. L'utilitaire MMXREF

MMXREF est un analyseur de références destiné principalement au fichier source écrit en langage C. Deux options sont possibles:

*a: mmxref -t fichier.c

Affiche à l'écran, par ordre alphabétique, les "mot clefs" du fichier 'source.c' et spécifie le(s) numéro(s) de ligne de leur(s) localisation(s). MMXREF considère un "mot clef" comme une chaîne composée de caractères alphanumériques (y compris le trait de soulignement '_'), la première lettre du "mot clef" ne devant pas être un chiffre. Tout autre caractère sert de délimiteur.

*a: mmxref fichier.c

Liste à l'écran le fichier 'source.c' en numérotant les lignes. Puis exécute la même opération qu'à l'option précédente.

Par défaut, la sortie se fait vers l'écran. Il est cependant permis de la diriger vers un fichier spécifié avec l'option:

>chemin et nom du fichier de sortie

N.D.T.: Il est également possible d'ajouter les sorties à un fichier déjà existant en utilisant l'option '>>nom du fichier'. Rappelons d'autre part qu'en lançant le programme à partir de 'COMMAND.TOS', le périphérique de sortie peut être modifié. Ainsi: 'a:' ligne de commande' >PRN: oriente les sorties vers l'imprimante.

9. L'utilitaire RM

RM efface du catalogue d'un disque un fichier spécifié. Cet utilitaire est particulièrement utilisé dans les fichiers '.bat' pour détruire les fichiers devenus inutiles. Cela permet ainsi de ne pas encombrer un disque. Format de la commande:

RM chemin et nom de fichier

EXEMPLE: rm a:truc.1

10. L'utilitaire WAIT

WAIT attend l'appui d'une touche. Cet utilitaire est utilisé dans les fichiers '.bat' pour temporiser et permettre ainsi une lecture de l'écran.

ATTENTION: Si un fichier '.BAT' est exécuté à partir de 'COMMAND.TOS' en définissant l'imprimante comme périphérique de sortie, cet utilitaire ne doit pas être utilisé: le blocage du clavier vous oblige au 'RESET'.

ANNEXE A

LES MESSAGES D'ERREUR

AVANT-PROPOS DES TRADUCTEURS

- A.1. LES MESSAGES D'ERREUR DE CP68
 - A.1.1. MESSAGES DES ERREURS UTILISATEUR
 - A.1.2. ERREUR DE LOGIQUE INTERNE
- A.2. LES MESSAGES D'ERREUR DE C068
 - A.2.1. MESSAGES DES ERREURS UTILISATEUR
 - A.2.2. MESSAGES DE MISE EN GARDE (WARNING)
 - A.2.3. ERREURS DE LOGIQUE INTERNE
- A.3. LES MESSAGES D'ERREUR DE C168
 - A.3.1. MESSAGES DES ERREURS UTILISATEUR
 - A.3.2. ERREURS DE LOGIQUE INTERNE
- A.4. LES MESSAGES D'ERREUR D'AS68
 - A.4.1. MESSAGES DES ERREURS UTILISATEUR
 - A.4.2. ERREURS DE LOGIQUE INTERNE
- A.5. LES MESSAGES D'ERREUR DE LINK68
 - A.5.1. MESSAGES DES ERREURS UTILISATEUR
 - A.5.2. ERREURS DE LOGIQUE INTERNE
- A.6. LES MESSAGES D'ERREUR DE L068
 - A.6.1. MESSAGES DES ERREURS UTILISATEURS
 - A.6.2. ERREURS DE LOGIQUE INTERNE
- A.7. LES MESSAGES D'ERREUR DE AR68
 - A.7.1. MESSAGES DES ERREURS UTILISATEUR
 - A.7.2. ERREURS DE LOGIQUE INTERNE
- A.8. LES MESSAGES D'ERREUR DE DUMP
- A.9. LES MESSAGES D'ERREUR DE SIZE68
- A.10. LES MESSAGES D'ERREUR DE SEND
- A.11. LES MESSAGES D'ERREUR DE SID
- A.12. LES MESSAGES D'ERREUR DE COMMAND

AVANT-PROPOS DES TRADUCTEURS

Les différents messages d'erreur fournis en annexe 'A' n'ont évidemment pas été traduits afin de respecter l'intitulé des messages qui s'afficheront sur votre écran (bien que, s'agissant de messages d'erreur, nous ne vous le souhaitions pas!).

Des ajouts et des suppressions ont été effectués par rapport à la documentation anglaise. Nous avons en effet analysé la zone des chaînes de messages (zone 'data') pour chaque programme fourni avec le pack de développement. Les messages explicités dans cette annexe sont le fruit de cette mise à jour.

A lire les messages fournis dans cette annexe, on peut à bon droit être surpris du nombre d'erreurs liées à des facteurs externes à l'utilisateur (messages de logique interne annonçant une bogue du programme utilisé) ou même de celles bien difficiles à évaluer et à corriger (par exemple, le débordement des différentes tables de travail du logiciel employé, nécessitant une fastidieuse tâche de réécriture).

Nos expériences d'utilisateurs des outils de développement nous permettent d'être rassurants. Jamais encore de tels messages ne nous sont apparus sur une pratique commune de plusieurs centaines de milliers de lignes compilées, assemblées et "liées".

Nous tenons par contre à attirer votre attention sur un fait assez fréquent pour être signalé. Lorsqu'un message d'erreur apparaît sans que vous puissiez en déterminer la cause, la détérioration du support peut être logiquement invoquée. Effacez votre programme defectueux et recopiez le à partir de votre original. Plusieurs copies de sauvegarde des logiciels les plus sollicités ne sont d'ailleurs pas un luxe.

Rappelons une fois encore que si, par défaut, les messages d'erreur sont dirigés vers l'écran, il est toujours possible d'en modifier l'orientation. La sortie peut se faire sur fichier (Cf. 'COMMAND.TOS' et les différents modes d'emploi des utilitaires) ou vers un autre périphérique (Cf. 'COMMAND.TOS' et la note des traducteurs au Chapitre A.2.).

A.1. Les messages d'erreur de CP68

Le pré-processeur CP68 retourne deux types de messages d'erreur:

- * Les messages dûs aux erreurs de l'utilisateur (erreur de syntaxe,...), aux limites de CP68 (débordement de pile,...) ou à une lecture/écriture de fichier déficiente. Dans tous les cas, une correction du programme source et/ou une adaptation de l'environnement sont nécessaires.
- * Les messages dûs aux erreurs de logique interne et qui dénotent une bogue de CP68.

Ces messages obéissent au format suivant:

n° texte du message d'erreur

Le signe '#' indique que le message est généré par CP68; suit le numéro de ligne où l'erreur est détectée puis le texte du message décrivant l'erreur.

A.1.1. MESSAGES DES ERREURS UTILISATEUR

n° argument buffer overflow

A la ligne indiquée, une liste d'arguments contient trop de caractères. Le tampon des arguments de CP68 est plein. Réduisez le nombre de caractères.

n° bad argument: arg

A la ligne indiquée, l'argument représenté par la variable 'arg' contient un caractère invalide.

n° bad character 0: valeur

La ligne indiquée contient un caractère illégal. Le code ASCII de ce caractère est fourni en octal.

n° bad define name: nom

Le nom spécifié contient un ou plusieurs caractères illégaux.

n° bad include file

La syntaxe de la directive '#include' est incorrecte. Seuls ces deux formats sont valides:

```
#include <fichier.h>
#include "fichier.h"
```


CP68: MESSAGES D'ERREUR

n° bad include file nom-fichier

A la ligne indiquée, le nom du fichier à inclure soit contient un caractère illégal, soit est composé de plus de 8 caractères.

n° can't open include file: nom-fichier

Le fichier à inclure indiqué ne peut être lu. Soit le chemin est incorrect, soit le fichier n'existe pas.

n° can't open source file: nom-fichier

Le fichier source indiqué ne peut être lu. Soit le chemin est incorrect, soit le fichier n'existe pas.

n° can't create: nom-fichier

Le fichier produit par CP68 ne peut être écrit sur disque. Soit le chemin spécifié est incorrect, soit l'espace disque est insuffisant, soit le disque est protégé contre l'écriture.

n° condition stack overflow

Le code source contient trop de directives de compilation conditionnelle. La pile est saturée. Le numéro de ligne est postérieur au débordement.

n° define recursion

A la ligne indiquée, un nom symbolique est défini par lui-même. Exemple:
#define VALEUR VALEUR+'A'

n° define table overflow

Le code source contient l'une des caractéristiques suivantes:
* trop de noms symboliques
* des noms symboliques trop longs
* trop d'expressions
* des expressions trop longues
Simplifiez avant un nouvel appel à CP68.

n° expression operator stack overflow

A la ligne indiquée, une expression contient de trop nombreux opérateurs. Simplifiez l'expression.

n° expression stack overflow

Trop nombreux termes à la ligne indiquée.

CP68: MESSAGES D'ERREUR

n° expression syntax

L'expression de la ligne indiquée comporte une erreur de syntaxe.

n° includes nested too deeply

La directive '#include' de la ligne indiquée provoque une inclusion emboîtée d'une trop grande profondeur. Cette erreur survient lorsqu'un fichier à inclure inclut lui-même un fichier à inclure qui inclut lui-même..., etc.... provoquant ainsi un emboîtement de plus de 7 fichiers.

n° invalid #else

La directive '#else' est détectée sans qu'une directive '#if' l'ait précédée.

n° invalid #endif

La directive '#endif' est détectée sans qu'une directive '#if' l'ait précédée.

n° invalid preprocessor command

A la ligne indiquée, une commande est invalide ou mal orthographiée.

n° line overflow

La ligne indiquée contient plus de 255 caractères, le maximum admis.

n° macro argument too long

A la ligne indiquée, le nom d'un argument comporte plus de 8 caractères, le maximum admis.

n° no */ before EOF

Un commentaire n'a pas été refermé par les signes '*/'.

n° string too long

La ligne indiquée contient une chaîne de plus de 255 caractères, le maximum admis.

n° symbol table overflow

Le code source utilise de trop nombreux symboles et provoque un débordement de la table des symboles.

CP68: MESSAGES D'ERREUR

n° too many arguments

A la ligne indiquée, l'un des noms symboliques contient plus de 9 arguments, le maximum admis.

n° unexpected EOF

Ce message indique un code source incomplet. Examinez votre listing, vérifiez que le fichier n'est pas détérioré.

n° unmatched conditional

La directive '#if' a été rencontrée mais pas la directive '#endif'.

n° usage: CP68 [-i x] inputfile outputfile

La ligne de commande invoquant CP68 est incorrecte. Le modèle d'une syntaxe valide est affiché.

A.1.2. ERREUR DE LOGIQUE INTERNE

Le message qui suit annonce une erreur fatale dans la logique interne de CP68:

n° too many characters pushed back

Si ce message apparaissait, il vous faudrait avant toute chose "rafraîchir" CP68. Si l'erreur subsistait, contactez ATARI-FRANCE après avoir rassemblé les informations suivantes:

- * la version de votre système d'exploitation
- * la configuration matérielle sur laquelle vous travaillez
- * la description détaillée de ce qui a occasionné l'erreur

A.2. Les messages d'erreur de C068

L'analyseur syntaxique C068 retourne trois types de messages d'erreur:

- * Les messages dûs aux erreurs de l'utilisateur (erreur de syntaxe,...), aux limites de C068 (débordement de pile,...) ou à une lecture/écriture de fichier déficiente. Dans tous les cas, une correction du programme source et/ou une adaptation de l'environnement sont nécessaires.
- * Les messages de mise en garde qui indiquent une syntaxe résolue par C068, mais dont l'ambiguïté prête à confusion. Ces messages sont précédés du texte "(warning)".
- * Les messages dûs aux erreurs de logique interne et qui dénotent une insuffisance de C068.

Ces messages obéissent tous au format suivant:

- * [(warning)]n° texte du message d'erreur

Le signe '*' indique que le message est généré par C068; suit le numéro de ligne où l'erreur est détectée puis le texte du message décrivant l'erreur. Le texte "(warning)" n'est placé qu'en cas de messages de mise en garde.

N.D.T.: C068 est sans aucun doute la phase de compilation entraînant le plus grand nombre d'erreurs, et ceci pour deux raisons:

La première tient au fait que l'analyseur syntaxique est chargé de la plus grosse part du travail de compilation.
La seconde raison provient de la complexité de la tâche engagée qui provoque, à la première erreur rencontrée, une avalanche de messages divers et hors de propos provoquant un intempestif scrolling d'écran, et ce jusqu'à ce que C068 retombe "sur ses pieds" (s'il y retombe).
Ceci nous entraîne à vous donner les conseils suivants:

- * Le premier message affiché est le plus important, il ne faut pas laisser l'écran "scroller". Plusieurs solutions existent:
 - geler l'écran avec 'CONTROL-S' ('CONTROL-Q le dégèle) et ce de préférence juste lorsque CP68 a terminé sa tâche. L'inconvénient est de vous contraindre à rester près de la console.
 - orienter, à partir de 'COMMAND.TOS' les sorties vers l'imprimante. Mais si C068 s'emballe, bonjour le listing!
 - orienter, toujours à partir de 'COMMAND.TOS' les sorties vers un fichier disque. Vous n'avez malheureusement plus connaissance du déroulement du processus. C'est pourtant sans doute la meilleure solution.

- * Ne tenez pas compte des messages qui suivent de trop près la première erreur. Notre expérience nous pousse à dire qu'il faut compter au minimum 10 lignes sans erreur pour qu'un message puisse correspondre à une nouvelle erreur.

C068: MESSAGES D'ERREUR

A.2.1. MESSAGES DES ERREURS UTILISATEUR

La liste des messages d'erreur qui suit est classée par ordre alphabétique. Pour chaque erreur, un commentaire vous fournit explications et suggestions. En cas de doute sur la validité d'une expression, référez-vous à votre manuel de langage C.

* n°: address of register

Vous avez tenté de demander l'adresse d'une variable 'register'. Corrigez avant de recompiler.

* n°: arrays limited to five dimensions

Les tableaux multi-dimensionnels à 'n' dimensions sont limités à une valeur n = 5.

* n°: assignable operand required

A la ligne indiquée, l'opérande destinataire de l'affectation (situé à gauche du signe "=") n'est pas valide. L'opérande par exemple est une constante au lieu d'une variable.

* n°: bad character constant

A la ligne indiquée, une constante caractère est invalide. Une constante caractère est un unique caractère entre apostrophes. Placer plusieurs caractères dans une constante provoque ce message.

* n°: bad indirection

Vous essayez de référencer par adresse et non par valeur, mais l'expression que vous utilisez n'est pas une adresse. Fournissez une valeur ou une adresse valide.

* n°: can't open filename

Soit le nom de fichier, soit le chemin spécifié sont incorrects.

* n°: case not inside a switch block

L'instruction 'case', rencontrée à la ligne indiquée, n'est pas à l'intérieur d'un bloc 'switch'.

* n°: character constant too long

La constante caractère est trop longue. Un seul caractère entre apostrophes est permis.

C068: MESSAGES D'ERREUR

* n°: constant required

L'opération de la ligne indiquée requiert une constante.

* n°: declaration syntax

La syntaxe de la déclaration est incorrecte. Référez vous à votre manuel de langage C.

* n°: default not inside a switch block

L'instruction 'default' rencontrée n'est pas à l'intérieur d'un bloc 'switch'.

* n°: dimension table overflow: arrays limited to five dimensions

Un tableau à 'n' dimensions a été créé. La valeur 'n' trop élevée provoque un débordement. Les tableaux multi-dimensionnels sont limités à une valeur 'n' égale à 5.

* n°: duplicate case value

Dans le même 'switch', deux instructions 'case' sont identiques. Corrigez et recompilez.

* n°: expected label

L'instruction 'goto' renvoie à un label absent.

* n°: expression too complex

Erreur due aux limites internes de C068: l'expression rencontrée est trop complexe pour être évaluée.

* n°: external definition syntax

A la ligne indiquée, la syntaxe de la définition externe est incorrecte. Référez vous à votre manuel de langage C.

* n°: field overflows byte

Un champ de bits réclame plus de bits que n'en contient un octet.

* n°: field overflows word

Un champ de mots réclame plus d'octets que n'en contient un mot.

* n°: function body syntax

Une accolade a été omise en début de fonction.

C068: MESSAGES D'ERREUR

* n°: illegal call

Une variable est incorrectement utilisée comme nom de fonction. Il est possible d'envoyer l'adresse d'une fonction en argument si ce dernier est déclaré comme pointeur [Exemple: (*tit_fonc())]. Dans certains cas, ce message est un avertissement.

* n°: illegal function declaration

La classe d'allocation de la fonction est illégale. Pour une fonction, seules les classes statique et externe sont valides.

* n°: illegal operator '<'

L'opérateur désigné est illégal: employez '<='.

* n°: illegal operator '>'

L'opérateur désigné est illégal: employez '>='.

* n°: illegal type conversion

Une affectation illégale est rencontrée. Cette erreur survient habituellement lorsque l'on tente de convertir un pointeur (32 bits) en un entier (16 bits).

* n°: illegal structure operation

L'opération rencontrée est invalide lorsqu'elle concerne une structure.

* n°: indirection on function invalid

L'opérateur unaire d'indirection (*) est rattaché au nom d'une fonction.

* n°: initializer alignment

Ce message indique habituellement une valeur d'initialisation manquante ou inadéquate.

* n°: initializer list too long

La liste des initialisations est trop longue pour C068.

* n°: invalid break statement

L'instruction 'break' n'est pas dans une boucle ou dans un 'switch'.

C068: MESSAGES D'ERREUR

* n°: invalid character

Un caractère invalide est détecté à la ligne indiquée.

* n°: invalid continue statement

L'instruction 'continue' n'est pas à l'intérieur d'une boucle. Cette erreur peut se produire si l'instruction 'continue' est détectée dans le corps d'un 'switch' qui n'est pas lui-même dans une boucle.

* n°: invalid conversion

Une affectation illégale est décelée. Par exemple entre un long mot et un entier.

* n°: invalid data type

La ligne indiquée contient une expression qui tente d'assimiler deux quantités incompatibles: par exemple un pointeur et un entier.

* n°: invalid declaration

A la ligne indiquée, la déclaration est invalide.

* n°: invalid declarator

Le déclarateur de la ligne indiquée n'est pas reconnu.

* n°: invalid expression

L'expression contient une erreur de syntaxe.

* n°: invalid field size

A la ligne indiquée, le champ a une taille inférieure ou égale à zéro.

* n°: invalid field type description

Un pointeur ou un long mot est placé dans un champ de bits.

* n°: invalid for statement

L'instruction 'for' rencontrée contient une erreur de syntaxe. Vérifiez votre manuel de langage C.

* n°: invalid initializer

A la ligne indiquée, l'expression située à droite du signe '=' est invalide.

C068: MESSAGES D'ERREUR

* n°: invalid label

Un nom de variable est utilisé comme un label.

* n°: invalid long declaration

Vous tentez de déclarer comme un long une valeur qui ne peut pas l'être, par exemple un caractère.

* n°: invalid register specification

Vous tentez de placer dans un registre une valeur illégale, par exemple une structure, un tableau ou une fonction.

* n°: invalid short declaration

Vous tentez de déclarer comme un entier une valeur qui ne peut pas l'être.

* n°: invalide storage class

Une classe d'allocation illégale est rencontrée dans une déclaration.

* n°: invalid structure declaration: nom

La taille de la structure indiquée par la variable "nom" est inférieure ou égale à zéro.

* n°: invalid structure member name

Une référence est faite à un membre d'une structure, et ce membre n'existe pas.

* n°: invalid structure prototype: nom

A la ligne indiquée, une référence est faite à un nom de structure qui n'a pas été déclarée.

* n°: invalid type declaration

La déclaration de type à la ligne indiquée est invalide.

* n°: invalid typedef statement

Plusieurs affectations suivent l'instruction "typedef". Une seule pourtant est permise.

* n°: invalid unsigned declaration

La déclaration 'unsigned' est illégale pour ce type de variable.

C068: MESSAGES D'ERREUR

* n°: invalid ?: operator syntax

Les opérateurs "?" et ":" (opérateurs conditionnels) sont mal utilisés.

* n°: label redeclaration: label

Le label indiqué est utilisé deux fois.

* n°: missing colon

Le signe de ponctuation ":" (deux-points) a été omis.

* n°: missing { in initialization

L'accolade ouvrante a été omise au début d'une initialisation de tableau ou de structure.

* n°: missing) in initialization

L'accolade fermante a été omise en fin d'initialisation d'un tableau ou d'une structure. Cette erreur survient également lorsque le nombre d'éléments initialisant un tableau est supérieur à la capacité de ce dernier.

* n°: missing while

L'instruction 'do' rencontrée à la ligne indiquée n'a pas de 'while' lui correspondant.

* n°: missing semicolon

Le caractère de ponctuation ';' (point-virgule) a été omis.

* n°: no structure name

Une référence est faite à une structure sans indiquer le nom de cette structure.

* n°: no */ before EOF

Le dernier commentaire ouvert par /* n'a pas été refermé.

* n°: not in parameter list: x

La variable "x", déclarée comme un argument de la fonction, n'est pas dans la liste des arguments reçus.

* n°: parenthesized expression syntax

La ligne indiquée contient une erreur de syntaxe relative aux parenthèses.

C068: MESSAGES D'ERREUR

* n°: redeclaration: symbole

La chaîne 'symbole' a été déclarée deux fois. Ce message s'affiche également si un appel de fonction est rencontré avant que cette fonction ait été déclarée.

* n°: string cannot cross line

Un code 'Retour Chariot' ne doit pas "couper" une chaîne. Cela dépend en partie de l'éditeur utilisé: ce dernier ne doit pas rajouter un 'CR' en fin d'écran. Si la chaîne ne peut tenir sur une seule ligne (une ligne est une suite de caractères sans code de 'Retour Chariot'), terminez la première ligne par le caractère '\\' indiquant ainsi que le reste de la chaîne est à la ligne suivante.

* n°: string too long

La chaîne, à la ligne indiquée, dépasse les 255 caractères permis: fractionnez la chaîne en utilisant le caractère '\\' (voir message précédent).

* n°: structure operation not yet implemented

A la ligne indiquée, une opération sur une structure est effectuée. Cette opération n'est pas encore implémentée sur la version actuelle de C068.

* n°: structure table overflow

Votre programme contient trop de structures et un dépassement de capacité de la table des structures est détecté. Éliminez certaines structures.

* n°: symbol table overflow

Votre programme utilise trop de symboles et un dépassement de capacité de la table des symboles est détecté. Éliminez certains symboles.

* n°: temp creation error

A la ligne indiquée, le disque ou le nom de fichier invoqué est incorrect.

* n°: too many cases in switch

Le bloc 'switch' de la ligne indiquée comprend de trop nombreux 'case'. Éliminez-en certains.

C068: MESSAGES D'ERREUR

* n°: too many initializers

Le nombre d'éléments initialisant une structure est supérieur au nombre des membres à initialiser.

* n°: too many parameters

A la ligne indiquée, la fonction déclarée reçoit trop de paramètres. Modifiez le programme source.

* n°: undefined label: label

Le label indiqué n'a pas été défini.

* n°: undefined symbol: symbole

Le symbole indiqué n'a pas été défini.

* n°: unexpected EOF

Cette erreur survient habituellement lorsqu'une fonction n'a pas été refermée par l'accolade fermante '}' ou qu'un commentaire n'a pas été refermé par '*/'. Localisez et corrigez l'erreur.

* n°: unmatched quote

Les guillemets ont été ouverts mais non refermés.

* n°: usage: C068 source asm str

La syntaxe de la commande invoquant C068 est incorrecte.

* n°: { not matched by }

Une accolade ouvrante a été rencontrée mais l'accolade fermante est omise. Cette erreur est détectée habituellement dans une séquence d'initialisation.

* n°: & operator illegal

L'opérateur unaire d'adressage '&' est utilisé illégalement. Cette erreur survient par exemple lorsque vous demandez l'adresse d'une variable de type 'register'.

A.2.2. MESSAGES DE MISE EN GARDE

* n°: (warning) initializer truncated

La constante initialisant une variable a été tronqué pour s'adapter au type de cette variable.

C068: MESSAGES D'ERREUR

* n°: (warning) integral type expected

Une fonction retourne une valeur non-entière mais la variable receptrice est du type entier.

* n°: (warning) null expression encountered

L'instruction 'return' ne renvoie aucune valeur. Chaque fois que la ligne 'return();' est rencontrée, C068 affiche ce message d'alerte. Utilisez 'return' sans parenthèses.

* n°: (warning) old fashion assignment statement

Indique que la syntaxe employée pour une expression n'est plus au goût du jour. Par exemple: a += b; au lieu de a =+ b; Ce message n'est pas à négliger, notamment en ce qui concerne les opérations sur structures.

* n°: (warning) old fashion assignment "<=<"

Indique que la syntaxe '<=<' a été utilisée. Il est préférable d'employer '<=<='.

* n°: (warning) old fashion assignment ">>"

Indique que la syntaxe '>>' a été utilisée. Il est préférable d'employer '>>='.

* n°: (warning) pointer subtraction yields a long result

Indique qu'une soustraction de pointeurs est stockée dans une variable qui n'est pas de type long. Ce n'est pas obligatoirement une erreur, mais cela peut provoquer de nombreux déboires.

* n°: (warning) short assigned to pointer

Une variable de type pointeur reçoit une donnée entière.

* n°: (warning) string initializer truncated

Annonce qu'une chaîne a été tronquée. Par exemple:
char chaîne[7] = "BONJOUR";

Ici, le tableau ne contient en fait que les 6 premiers caractères de la chaîne "BONJOUR", le septième étant le caractère de fin de chaîne.

* n°: (warning) string used to initialize character value

Annonce qu'une variable de type 'char' est initialisée par une chaîne. La variable contient uniquement le premier caractère. Exemple:
char c = "BONJOUR";

C068: MESSAGES D'ERREUR

* n°: (warning) suspect conversion operation

La conversion de format de différents types de variables est suspecte. Ce n'est pas obligatoirement une erreur, mais il convient de vérifier que l'opération est bien celle que vous souhaitez.

* n°: (warning) unsigned char unimplemented, signed char assumed

Le compilateur ALCYON ne reconnaît pas le type 'unsigned char'; la variable est donc convertie en 'char'.

* n°: (warning) unsigned long unimplemented, signed long assumed

Le compilateur ALCYON ne reconnaît pas le type 'unsigned long'; la variable est donc convertie en 'long'. Une astuce consiste à déclarer une variable longue comme un pointeur afin de la rendre 'unsigned'.

A.2.3. ERREURS DE LOGIQUE INTERNE

Ces messages indiquent une erreur fatale dans la logique interne de l'analyseur syntaxique C068.

- * n°: can't copy filename
- * n°: invalid keyword
- * n°: too many chars pushed back
- * n°: too many tokens pushed back

Si l'un de ces messages apparaissait, il vous faudrait avant toute chose "rafraîchir" C068. Si l'erreur subsistait, contactez ATARI-FRANCE après avoir rassemblé les informations suivantes:

- * la version de votre système d'exploitation
- * la configuration matérielle sur laquelle vous travaillez
- * la description détaillée de ce qui a occasionné l'erreur

A.3. Les messages d'erreur de C168

Le générateur de code C168 retourne deux types de messages d'erreur:

- * Les messages dûs aux erreurs de l'utilisateur (erreur de syntaxe,...), aux limites de C168 (débordement de pile,...) ou à une lecture/écriture de fichier déficiente. Dans tous les cas, une correction du programme source et/ou une adaptation de l'environnement sont nécessaires.
- * Les messages dûs aux erreurs de logique interne et qui dénotent une bogue de C168.

Les messages obéissent au format suivant:

**** n° texte du message**

Les deux astérisques indiquent que le message est généré par C168; suit le numéro de ligne puis le texte décrivant l'erreur rencontrée. Il est important que toutes les erreurs détectées par C068 aient été corrigées. Dans le cas contraire, aucune fiabilité n'est à accorder aux messages affichées.

A.3.1. MESSAGES DES ERREURS UTILISATEUR

**** n° can't create nom-fichier**

C168 ne peut créer le fichier indiqué. Soit le chemin spécifié est incorrect, soit l'espace disque est saturé, soit le disque est protégé en écriture.

**** n° can't open nom-fichier**

C168 ne peut ouvrir le fichier indiqué. Soit le chemin spécifié est incorrect, soit le fichier est absent ou détérioré.

**** n° divide by zero**

A la ligne indiquée, une division par zéro est décelée.

**** n° expression too complex**

L'expression de la ligne indiquée est trop complexe pour C168. Simplifiez-la avant de recompiler.

**** n° modulus by zero**

A la ligne indiquée, le second opérande derrière l'opérateur '%' est égal à zéro.

C168: MESSAGES D'ERREUR

** n° structure operation not yet implemented

A la ligne indiquée, l'opération tentée sur une structure est invalide pour la version actuelle de C168.

** n° usage: c168 icode asm [-DLmec]

La ligne de commande invoquant C168 est incorrecte.

A.3.2. ERREURS DE LOGIQUE INTERNE

La liste des messages qui suit annonce une erreur fatale dans la logique interne de C168:

n° code skeleton error: op
 n° intermediate code error
 n° invalid initialization
 n° invalid register expression
 n° no code table for op

Si l'un de ces messages apparaissait, il vous faudrait avant toute chose "rafraichir" C168. Si l'erreur subsistait, contactez ATARI-FRANCE après avoir rassemblé les informations suivantes:

- * la version de votre système d'exploitation
- * la configuration matérielle sur laquelle vous travaillez
- * la description détaillée de ce qui a occasionné l'erreur

A.4. Les messages d'erreur de AS68

AS68 retourne trois types de messages d'erreur:

- * Les messages dûs à une erreur de l'utilisateur (syntaxe, erreur de contexte,...). Ces messages ne stoppent pas l'assemblage, permettant ainsi de détecter les erreurs suivantes. Par contre, une correction du programme source et un nouvel assemblage s'imposent.
- * Les messages d'AS68 qui stoppent le processus d'assemblage. Ce sont particulièrement des erreurs de lecture/écriture sur disque. La plupart n'imposent pas de correction du source mais obligent à modifier l'environnement disque et à relancer la phase d'assemblage.
- * Les messages dûs aux erreurs de logique interne et qui dénotent une insuffisance de AS68.

Ces messages obéissent au format suivant:

& n° texte du message d'erreur

Le signe '&' indique que le message est généré par AS68; suit le numéro de ligne où l'erreur est détectée puis le texte du message décrivant l'erreur.

Le numéro de ligne fait référence au FICHIER SOURCE '.S' et non pas, si vous venez du compilateur, de votre source C. Il est cependant possible d'obliger C168 (Cf. Chapitre 3) à intégrer les numéros de ligne du source C comme des commentaires dans le fichier '.S'.

Les messages sont affichés sur l'écran (sauf si vous avez spécifié une orientation vers un fichier); en fin d'assemblage, est affiché le nombre total d'erreurs rencontrées.

A.4.1. MESSAGES DES ERREURS UTILISATEURS

Les messages qui suivent sont classés par ordre alphabétique.

& n° absolute value required

A la ligne indiquée, une valeur absolue est requise.

& n° assembler confusion...

A la ligne indiquée, l'instruction peut être interprétée de différentes façons.

& n° backward assignment to *

A la ligne indiquée, une expression utilise un symbole non encore défini.

AS68: MESSAGES D'ERREUR

& n° bad use of symbol

Le symbole de la ligne indiquée a été défini comme GLOBAL et COMMON. Les deux types sont antinomiques.

& n° code or data not allowed in bss

Après la directive '.bss', une instruction ou une donnée initialisée a été placée.

& n° constant required

L'expression de la ligne indiquée requiert une constante.

& n° endc expected

La directive '.endc' n'a pas été rencontrée alors qu' elle était attendue.

& n° end statement not at end of source

Derrière la directive 'end', un seul Retour Chariot est permis: ni espaces, ni commentaires ne sont tolérés.

& n° illegal adressing mode

L'instruction de la ligne indiquée utilise un mode d'adressage invalide.

& n° illegal constant

La ligne indiquée contient une constante illégale.

& n° illegal expr

La ligne indiquée contient une expression illégale.

& n° illegal extension

L'instruction de la ligne indiquée demande une extension illégale.

& n° illegal external

La ligne indiquée fait référence à une variable externe sur 8 bits: définissez la référence localement ou utilisez un adressage sur 16 bits.

& n° illegal format

Le format de l'expression ou de l'instruction est illégal.

AS68: MESSAGES D'ERREUR

& n° illegal index register

La ligne indiquée contient un registre d'adresse invalide.

& n° illegal relative address

Le mode d'adressage spécifié n'est pas valide pour l'instruction de la ligne indiquée.

& n° illegal shift count

A la ligne indiquée, une instruction de décalage de bits a pour premier opérande un registre contenant une valeur supérieure à 31, ou une valeur immédiate supérieure à 7.

& n° illegal size

L'instruction de la ligne indiquée requiert l'un des deux spécificateurs de taille: '.b' ou '.l'. La taille '.w' prise par défaut est invalide.

& n° illegal string

La ligne indiquée contient une chaîne illégale.

& n° illegal text delimiter

Le délimiteur de texte rencontré est invalide. Utilisez les apostrophes ('texte') ou les guillemets ("texte").

& n° illegal 8-bit displacement

La ligne indiquée spécifie un déplacement nécessitant plus de 8 bits. Utilisez une instruction sur 16 ou 32 bits.

& n° illegal 8-bit immediate

La ligne indiquée contient un opérande immédiat de plus de 8 bits. Utilisez une instruction sur 16 ou 32 bits.

& n° illegal 16-bit displacement

La ligne indiquée spécifie un déplacement nécessitant plus de 16 bits. Utilisez une instruction sur 32 bits.

& n° illegal 16-bit immediate

La ligne indiquée contient un opérande immédiat de plus de 16 bits. Utilisez une instruction sur 32 bits.

AS68: MESSAGES D'ERREUR

& n° invalid bit range

Une instruction de test de bit ('bclr', 'bset', ...) interroge un bit invalide. Si l'opération s'adresse à un emplacement mémoire, seul les bits 0 à 7 peuvent être testés.

& n° invalid data list

Dans la zone des données, une ou plusieurs valeurs de la ligne indiquée sont invalides.

& n° invalid first operand

Le premier opérande de l'expression est invalide.

& n° invalid instruction length

A la ligne indiquée, l'instruction requiert un des deux spécificateurs de taille: '.b' ou '.l.'. La taille '.w' prise par défaut est invalide.

& n° invalid label

Soit un opérande est manquant, soit le label n'est pas dans un format correct.

& n° invalid opcode

A la ligne indiquée, le mnémonique est absent ou invalide.

& n° invalid second operand

Le second opérande de l'expression est invalide.

& n° label redefined

Un label a été défini deux fois. La ligne indiquée pointe sur la deuxième définition.

& n° missing)

La parenthèse fermante d'une expression est manquante.

& n° no code or data allowed in offset

Après la directive '.offset', une instruction ou une donnée initialisée est rencontrée.

& n° no label for operand

Un label assigné à un opérande n'a pas été défini.

AS68: MESSAGES D'ERREUR

& n° opcode for 68010 only

L'instruction n'est valide que pour le microprocesseur 68010. La ligne de commande d'AS68 offre une option pour accepter ces codes.

& n° opcode redefined

A la ligne indiquée, un label a le même nom qu'un mnémonique.

& n° register required

L'instruction de la ligne indiquée requiert un registre comme source ou destination.

& n° relocation error

L'expression de la ligne indiquée contient plus d'un symbole défini de manière externe. Transformez l'un de ses symboles en local ou évaluez l'expression à l'intérieur du code.

& n° symbol required

A la ligne indiquée, l'instruction requiert un symbole.

& n° undefined symbol in equate

L'un des symboles de la directive '.equ' est indéfini.

& n° undefined symbol

La ligne indiquée contient un symbole indéfini. Le symbole doit appartenir au module ou être défini comme '.global'.

& n° unexpected endc

La directive '.endc' est rencontrée alors qu'une instruction d'assemblage conditionnel n'est pas en cours.

A.4.2. ERREURS STOPPANT L'ASSEMBLAGE

Lorsque ce type d'erreur apparaît, il est fréquent qu'AS68 ait déjà écrit des fichiers intermédiaires sur le disque. Il est recommandé de les effacer avant de relancer l'assemblage.

& cannot create init: AS68SYMB.DAT

Le fichier d'initialisation de l'assembleur ne peut pas être écrit: soit l'espace disque est insuffisant, soit le chemin défini par l'option '-S' est invalide, soit le disque est protégé contre l'écriture.

AS68: MESSAGES D'ERREUR

& can't open nom-fichier errno n°

Le fichier dont le nom est spécifié ne peut pas être ouvert. Soit le chemin indiqué est incorrect, soit le fichier n'existe pas.

& expr opstk overflow

L'expression de la ligne indiquée est trop complexe et provoque un dépassement de capacité de la pile. Simplifiez l'expression.

& expr tree overflow

Les trop nombreux termes d'une expression provoquent un débordement de la mémoire de travail d'AS68.

& I/O error on loader output file

Le disque sur lequel AS68 tente d'écrire un fichier de travail n'a plus d'espace disponible.

& I/O write error on it file

Le disque sur lequel AS68 tente d'écrire ses fichiers intermédiaires n'a plus d'espace disponible.

& it write error

Une erreur d'écriture est détectée. Vérifiez l'espace disponible sur le disque et la qualité du support.

& overflow of external table

Le code source utilise de trop nombreux symboles définis comme '.global' et la table de ces symboles, gérée par AS68, est pleine. Réécrivez le source en réduisant leur nombre.

& symbol table overflow

Le code source définit plus de symboles que n'en peut contenir la table d'AS68. Diminuez-en le nombre.

& temp file creat error nom-fichier

Impossible de créer un fichier temporaire. Vérifiez la validité du chemin et l'espace disque disponible.

& Unable to read init file: AS68SYMB.DAT

AS68 ne peut lire le fichier d'initialisation 'AS68SYMB.DAT'. Vérifiez la présence de ce fichier et la validité du chemin.

AS68: MESSAGES D'ERREUR

& Write error on init file: AS68SYMB.DAT

Le disque sur lequel AS68 tente d'écrire son fichier d'initialisation n'a pas l'espace disponible, ou est protégé contre l'écriture.

& write error on output file

Le disque sur lequel AS68 tente d'écrire le fichier objet (.o) n'a plus d'espace disponible ou est protégé contre l'écriture.

A.4.3. ERREURS DE LOGIQUE INTERNE

La liste des messages qui suit annonce une erreur fatale dans la logique interne de AS68:

& invalid radix in oconst
& seek error on source file

Si l'un de ces messages apparaissait, il vous faudrait avant toute chose "rafraîchir" AS68. Si l'erreur subsistait, contactez ATARI-FRANCE après avoir rassemblé les informations suivantes:

- * la version de votre système d'exploitation
- * la configuration matérielle sur laquelle vous travaillez
- * la description détaillée de ce qui a occasionné l'erreur

LINK68: MESSAGES D'ERREUR

A.5. Les messages d'erreur de LINK68

L'éditeur de liens LINK68 retourne deux types de messages d'erreur:

- * Les messages dûs aux erreurs de l'utilisateur (erreur de syntaxe,...), aux limites de LINK68 (débordement de pile,...) ou à une lecture/écriture de fichier déficiente. Dans tous les cas, une correction du programme source et/ou une adaptation de l'environnement sont nécessaires.
- * Les messages dûs aux erreurs de logique interne et qui dénotent une bogue de LINK68.

Ces messages obéissent au format suivant:

LINK68: n° TEXTE DU MESSAGE D'ERREUR

La chaîne 'LINK68' indique la provenance du message; suit le numéro de ligne où l'erreur est détectée puis le texte du message décrivant l'erreur.

La liste des messages qui suit est classée par ordre alphabétique.

A.5.1. MESSAGES D'ERREUR UTILISATEUR

LINK68: CANNOT OPEN <nom-fichier> FOR INPUT

Le fichier indiqué en entrée est invalide ou inexistant.

LINK68: CANNOT SET DATA OR BSS BASE WHEN USING OVERLAYS

Les options 'BSSBASE' et 'DATABASE' ne sont pas valides lorsque les fichiers à "lier" utilisent des segments de recouvrement.

LINK68: COMMAND LINE TOO LONG

La ligne de commande excède 132 caractères. Réduisez-en la taille ou utilisez un fichier d'entrée de commandes (voir l'utilitaire BATCH).

LINK68: "nom-symbole" DOUBLY DEFINED IN nom-fichier

Le symbole spécifié est défini deux fois. Le nom du fichier indiqué désigne l'emplacement où le symbole est détecté pour la seconde fois.

LINK68: FILE FORMAT ERROR IN nom-fichier

Le fichier indiqué n'est pas un fichier objet ou bien le fichier est détérioré.

LINK68: HEAP OVERFLOW--NOT ENOUGH MEMORY

Il n'y a pas assez de mémoire pour que LINK68 continue sa tâche. Utilisez l'option 'NOLOCALS' ou réécrivez votre source en employant moins de symboles. Vérifiez également qu'un disque virtuel ne confisque pas trop de mémoire à l'application en cours.

LINK68: ILLEGAL CHARACTER: 'caractère'

La ligne de commande contient un caractère invalide.

LINK68: ILLEGAL REFERENCE TO OVERLAY SYMBOL 'nom-symbole' FROM MODULE 'nom-module'

Le module indiqué contient une référence illégale au symbole spécifié.

LINK68: IMPROPERLY FORMED HEX NUMBER: valeur

Le nombre hexadécimal indiqué est invalide.

LINK68: INVALID RELOCATION FLAG IN nom-fichier

Le contenu du fichier indiqué est dans un format incorrect. Soit ce n'est pas un fichier objet, soit il est détérioré.

LINK68: INVALID SYMBOL FLAG IN nom-fichier

Soit le fichier indiqué n'est pas un fichier objet, soit il est détérioré.

LINK68: NESTED COMMAND FILES NOT ALLOWED

LINK68 n'autorise pas l'imbrication des fichiers de commandes (Cf. option 'COMMAND' de LINK68).

LINK68: NO RELOCATION BITS IN nom-fichier

Le fichier indiqué n'est pas un fichier objet ou bien il est détérioré.

LINK68: OVERLAY NESTED TOO DEEPLY

LINK68 n'autorise que 5 niveaux de recouvrement. Les programmes chaînés n'autorisent qu'un niveau de recouvrement.

LINK68: PARSE END BEFORE COMMAND STREAM END

LINK68 a rencontré la fin logique de la ligne de commande avant de rencontrer la fin physique. Corrigez la syntaxe de la ligne de commande.

LINK68: MESSAGES D'ERREURLINK68: READ ERROR ON FILE nom-fichier

Le fichier indiqué n'est pas complet ou a été détérioré.

LINK68: RELATIVE ADDRESS OVERFLOW AT valeur IN nom-fichier

Un débordement d'adresse d'un symbole est détecté. Ce message apparaît lors d'une erreur dans un fichier objet.

LINK68: SHORT ADDRESS OVERFLOW AT valeur IN nom-fichier

Un débordement d'adresse d'un symbole est détecté. Une adresse courte référence un symbole trop éloigné. Utilisez l'option 'IGNORE' de LINK68 ou assemblez le fichier indiqué avec l'option '-L'.

LINK68: SYMBOL TABLE OVERFLOW

Le code objet contient plus de symboles que n'en peut contenir la table des symboles. Utilisez l'option 'NOLOCALS' ou réécrivez le source en définissant moins de symboles.

LINK68: SYNTAX ERROR, EXPECTED: item

Une erreur de syntaxe est détectée dans la ligne de commande.

LINK68: TOO MANY OVERLAYS

LINK68 n'autorise que 255 segments de recouvrement. Simplifiez votre programme.

LINK68: UNABLE TO CREATE FILE: nom-fichier

LINK68 ne peut créer le fichier indiqué. Soit le fichier est orienté sur un mauvais chemin, soit le disque n'a pas l'espace disponible ou est protégé contre l'écriture.

LINK68: UNABLE TO OPEN TEMPORARY FILE nom-fichier

LINK68 ne peut ouvrir le fichier temporaire indiqué. Soit le chemin spécifié par l'option 'TEMPFILES' est incorrect, soit le fichier est inexistant (par exemple, une erreur est apparue lors de son écriture).

LINK68: UNDEFINED SYMBOL(S): liste

Le symbole ou les symboles qui suivent ce message sont indéfinis. Vérifiez que toutes les librairies nécessaires sont placées dans la ligne de commande. Vérifiez qu'une erreur dans votre source n'a pas différencié deux symboles que vous désiriez identiques. Si vos symboles ne sont pas référencés dans le programme lui-même, l'option 'UNDEFINED'

LINK68: MESSAGES D'ERREUR

doit être utilisée à l'édition de liens. Dans la ligne de commande invoquant LINK68, l'ordre des fichiers à "lier" a une grande importance. Ainsi 'LIBF' doit-il être placé dans la plupart des cas APRES 'GEMLIB'.

LINK68: UNEXPECTED END OF COMMAND STREAM

LINK68 a rencontré la fin physique d'une ligne de commande avant sa fin logique. Corrigez la ligne de commande.

LINK68: UNRECOGNIZED OR MISPLACED OPTION NAME: option

L'option indiquée n'est pas reconnue par LINK68 ou bien elle est mal placée. Corrigez la ligne de commande.

LINK68: WRITE ERROR ON FILE: nom-fichier

Le disque sur lequel LINK68 écrit le fichier indiqué n'a plus d'espace disponible ou est protégé contre l'écriture.

A.5.2. ERREURS DE LOGIQUE INTERNE

Le message qui suit annonce une erreur fatale dans la logique interne de LINK68:

LINK68: INTERNAL ERROR IN nom-procédure
LINK68: TEXT SIZE ERROR IN nom-fichier
LINK68: SEEK ERROR ON FILE: nom-fichier
LINK68: UNABLE TO REOPEN FILE nom-fichier

Si l'un de ces messages apparaissait, il vous faudrait avant toute chose "rafraîchir" LINK68. Si l'erreur subsistait, contactez ATARI-FRANCE après avoir rassemblé les informations suivantes:

- * la version de votre système d'exploitation
- * la configuration matérielle sur laquelle vous travaillez
- * la description détaillée de ce qui a occasionné l'erreur

L068: MESSAGES D'ERREUR

A.6. Les messages d'erreur de L068

L'éditeur de liens L068 retourne deux types de messages d'erreur:

- * Les messages dûs aux erreurs de l'utilisateur (erreur de syntaxe,...), aux limites de L068 (débordement de pile,...) ou à une lecture/écriture de fichier déficiente. Dans tous les cas, une correction du programme source et/ou une adaptation de l'environnement sont nécessaires.
- * Les messages dûs aux erreurs de logique interne et qui dénotent une bogue de L068.

Ces messages obéissent au format suivant:

: n° TEXTE DU MESSAGE D'ERREUR

Le signe ':' indique que le message est généré par L068; suit le numéro de ligne où l'erreur est détectée puis le texte du message décrivant l'erreur.

La liste des messages qui suit est classée par ordre alphabétique.

A.6.1. MESSAGES DES ERREURS UTILISATEUR

: 'symbole' duplicate definition 'nom-fichier'

Le symbole spécifié est défini deux fois. Le nom du fichier indiqué désigne l'emplacement où le symbole est détecté pour la seconde fois.

: file format error: nom-fichier

Soit le fichier indiqué n'est pas un fichier objet, soit il a été détérioré.

: File Format Error: Invalid symbol flags = drapeaux

L068 ne reconnaît pas les drapeaux indiqués. Soit le fichier lu n'est pas un fichier objet, soit il est détérioré.

: File Format Error: invalid relocation flag in nom-fichier

Le contenu du fichier indiqué n'a pas un format valide. Soit le fichier n'est pas un fichier objet, soit il est détérioré.

: File Format Error: no relocation bits in nom-fichier

Soit le fichier indiqué n'est pas un fichier objet, soit il est détérioré.

: Illegal option 'option'

L'option indiquée n'est pas reconnue par L068. Corrigez la ligne de commande.

: Invalid lo68 argument list

Ce message indique une erreur de format ou une option illégale dans la ligne de commande. Corrigez cette dernière.

: output file write error

Le disque sur lequel L068 tente d'écrire un fichier n'a pas d'espace disponible, ou le disque est protégé contre l'écriture.

: read error on file: nom-fichier

L068 annonce une erreur de lecture dans le fichier indiqué. Soit ce dernier est détérioré, soit l'étape précédente (en général AS68) a repercuté des erreurs d'écriture.

: symbol table overflow

Le code objet contient un nombre de symboles trop important. Réécrivez le source en utilisant moins de symboles.

: Unable to create: chemin et nom-fichier

Soit le chemin indiqué est incorrect, soit le disque n'a pas d'espace disponible ou est protégé contre l'écriture.

: Unable to open: nom-fichier

Le fichier indiqué est soit invalide, soit inexistant.

: Unable to open temporary file: chemin et nom-fichier

Soit le chemin indiqué par l'option '-F' est invalide, soit le disque n'a pas d'espace disponible ou est protégé contre l'écriture.

: Undefined symbol(s)

Le symbole ou les symboles qui suivent ce message sont indéfinis. Vérifiez que toutes les bibliothèques nécessaires sont placées dans la ligne de commande. Vérifiez qu'une erreur de frappe dans votre source n'a pas différencié deux symboles que vous désiriez identiques.

L068: MESSAGES D'ERREUR

: Write error on output file

Une erreur d'écriture est détectée. L'espace disponible sur le disque est insuffisant ou le disque est protégé contre l'écriture.

A.6.2. ERREURS DE LOGIQUE INTERNE

Les messages d'erreur suivants proviennent d'une erreur interne de L068. Si vous rencontrez un tel message, pensez d'abord à "rafraîchir" L068. Si l'erreur persiste, contactez ATARI-FRANCE.

```
: asgnext botch
: finalwr: text size error
: relative address overflow at lx in sn
: seek error on file: nom-fichier
: short address overflow
: unable to reopen nom-fichier
```

A.7. Les messages d'erreur d'AR68

L'utilitaire d'archivage AR68 retourne deux types de messages d'erreur:

- * Les messages dûs aux erreurs de l'utilisateur (erreur de syntaxe,...), aux limites de AR68 (débordement de pile,...) ou à une lecture/écriture de fichier déficiente. Dans tous les cas, une correction du programme source et/ou une adaptation de l'environnement sont nécessaires.
- * Les messages dûs aux erreurs de logique interne et qui dénotent une insuffisance de AR68.

La liste qui suit fournit ces différents messages par ordre alphabétique.

A.7.1. MESSAGES DES ERREURS UTILISATEUR

'abi' flags can only be used with 'r'

Les options 'a', 'b', et 'i' ne sont valides qu'avec la commande 'R'.

cannot open nom-fichier

Le fichier indiqué ne peut être ouvert. Soit le chemin est invalide, soit le fichier est inexistant.

invalid option flag: x

La commande ou l'option indiquée par la variable 'x' est invalide. Référez vous au mode d'emploi d'AR68.

not archive format: nom-fichier

Le fichier indiqué n'est pas une librairie.

nom-module not in archive file

Le module objet indiqué n'appartient pas à la librairie.

not object file: nom-fichier

Le fichier indiqué n'est pas un fichier objet. Il ne peut donc être ajouté à la librairie décrite dans la ligne de commande. Rappelons que seuls les fichiers à suffixe '.o' créés par AS68 sont susceptibles d'être intégrés à une librairie.

one and only one of DRTWX flags required

La ligne de commande d'AR68 requiert une et une seule des commandes 'D', 'R', 'T', 'W' ou 'X'.

AR68: MESSAGES D'ERREUR

only one of 'abi' flags can be used with 'r' flag

Une et une seule des options 'a', 'b' ou 'i' peut être utilisée avec la commande 'R'.

nom-module not in library

Le module objet indiqué n'appartient pas à la librairie. Fait double emploi avec le message 'not in archive file'.

Read error on nom-fichier

Le fichier indiqué ne peut être lu. Soit le fichier est détérioré, soit une erreur d'écriture est survenu lors de sa création, soit une erreur matérielle s'est produite.

Write error on nom-fichier

Le disque sur lequel AR68 tente d'écrire le fichier indiqué n'a pas d'espace disponible ou est protégé contre l'écriture.

A.7.2. ERREUR DE LOGIQUE INTERNE

Le message d'erreur suivant provient d'une erreur interne de AR68. Si vous rencontrez un tel message, pensez à "rafraîchir" AR68.

seek error on library

A.8. Les messages d'erreur de DUMP

L'utilitaire DUMP envoie ses messages d'erreur vers l'écran ou dans le fichier de sortie spécifié.

Unable to open nom-fichier

DUMP ne peut ouvrir le fichier indiqué. Soit le chemin est incorrect, soit le fichier est inexistant.

Usage: dump [-shhhhh] file

La ligne de commande est incorrecte, DUMP affiche la syntaxe à employer.

A.9. Les messages d'erreur de SIZE68

L'utilitaire SIZE68 envoie ses messages d'erreur vers l'écran ou dans le fichier de sortie spécifié.

cannot open nom-fichier

SIZE68 ne peut ouvrir le fichier indiqué. Soit le chemin est incorrect, soit le fichier est inexistant.

Not a programm file

Le fichier dont on demande la taille et la configuration n'est ni un fichier objet (.o), ni un fichier programme (.prg). Seuls ces deux types de fichier peuvent être gérés par SIZE68.

Usage is: size68 file-name-list

La ligne de commande est incorrecte, SIZE68 affiche la syntaxe à employer.

A.10. Les messages d'erreur de SEND

L'utilitaire SEND envoie ses messages d'erreur vers l'écran ou dans le fichier de sortie spécifié.

cannot create nom-fichier

SEND ne peut créer le fichier indiqué. Soit le chemin est incorrect, soit le disque n'a pas d'espace disponible.

cannot open nom-fichier

SEND ne peut ouvrir le fichier d'entrée indiqué. Soit le chemin est incorrect, soit le fichier n'existe pas.

read error: nom-fichier

Une erreur de lecture est détectée. Soit le fichier indiqué est détérioré, soit une erreur d'écriture s'est produite lors de sa création.

A.11. Les messages d'erreur de SID

Le logiciel d'aide à la mise au point SID retourne ses messages d'erreur vers l'écran. En dehors des messages fournis ci-dessous et classés par ordre alphabétique, SID peut afficher:

- * la liste de ses commandes disponibles lorsque la première lettre d'une commande n'est pas reconnue.
- * un point d'interrogation (?) lorsque les arguments d'une commande sont invalides.

Address error at HHHHHH

Une erreur d'adresse est détectée et sa localisation est fournie en argument (en hexadécimal). Survient principalement lors de la détection d'une adresse impaire.

Already set

Un point de passage que vous essayez de placer par la commande 'P' est déjà positionné.

Bad or non-existent RAM at HHHH

Cette erreur survient lors d'une tentative d'écriture d'une adresse mémoire invalide indiquée dans le message. Est considérée comme invalide toute valeur supérieure à l'adresse haute de la RAM (\$7FFFF sur 520ST, \$FFFFF sur 1040ST). Si ce message est retourné alors que l'adresse est valide, il peut indiquer une mémoire défectueuse.

Bus error at HHHHHH

Une erreur BUS est détectée à l'adresse indiquée. Ce message indique souvent une commande incongrue dont SID ne traite pas l'invalidité. Ainsi la commande 'E' avec pour argument un programme non-exécutable génère ce message. Selon les cas, cette erreur peut provoquer la sortie de SID ou même un "plantage" du système.

Cannot open file

Lors d'une lecture par la commande 'R', indique que soit le chemin spécifié est incorrect, soit le fichier à lire est inexistant.

Cannot open program file

Lors d'une lecture par la commande 'E', indique que soit le chemin spécifié est incorrect, soit le programme à exécuter est inexistant.

SID: MESSAGES D'ERREUR

'CHK' Exception

Indique une exception 'CHK' (donnée n'appartenant pas à un intervalle).

Divide by zero

Indique une exception 'DIVISION PAR ZERO'.

Encoutered the ILLEGAL-instruction

Indique une exception 'INSTRUCTION ILLEGALE'.

ERROR, no programm or file loaded

En réponse à la commande 'V', indique qu'aucun fichier ou programme n'a encore été lu.

File too big--read truncated

En réponse à la commande 'R', indique que le fichier lu occupe une place mémoire trop importante. La lecture en est donc tronquée.

Illegal size field

En réponse à la commande 'L', indique que la taille des opérandes de l'instruction est illégale. Soit la zone désassemblée n'est pas une zone de programme, soit l'adresse fournie ne correspond pas à un début d'instruction.

No change--Address must be even

En réponse à une commande 'P' avec comme argument une valeur impaire, signale que l'adresse d'un point de passage doit être paire.

Programm load error

En réponse à la commande 'E', indique une erreur de lecture. Le fichier programme lu est sans doute détérioré.

Too many pass points--limit is: 32

En réponse à la commande 'P', indique que vous êtes en train de fixer le trente-troisième point de passage: c'est un de trop.

Trace Exception

Une exception 'TRACE' est détectée.

SID: MESSAGES D'ERREUR

'TRAPV' Exception

Une exception 'TRAPV' est détectée.

Unknown instruction

En réponse à la commande 'L', signale qu'une instruction n'est pas reconnue. Soit la zone désassemblée n'est pas une zone de programme, soit l'adresse fournie ne correspond pas à un début d'instruction.

A.12. Les messages d'erreur de COMMAND.TOS

Par convention, lorsqu'un nom de fichier figure dans un message d'erreur, celui-ci a été remplacé par '...'. Les commandes susceptibles de provoquer chacun des messages ont été placées entre parenthèses en fin de commentaire de chaque message.

(A)bort, (R)etry, or (I)gnore ?

Ce message apparaît lorsqu'une erreur disque s'est produite (absence de disquette dans le lecteur, disquette double face dans un lecteur simple face, disquette non formatée). Appuyer la touche 'A' provoque le retour au COMMAND.TOS sans exécution de la commande. Appuyer la touche 'R' provoque une nouvelle tentative d'accès au disque. Appuyer la touche 'I' provoque l'exécution de la suite de la commande comme si rien ne s'était produit (en général, cela entraîne un nouveau message d'erreur!).

Cette erreur peut se produire avec toutes les commandes comportant au moins un accès disque.

Cannot copy ... to itself

Le nom de fichier à copier et celui du fichier sur lequel copier sont identiques (COPY).

Command is incompletely specified.

Vous n'avez pas indiqué de nom de fichier en paramètre (REN, CAT, TYPE, RM, ERA, DEL) ou bien vous n'avez pas fourni deux noms de fichiers en paramètres (COPY, MOVE).

Command not found.

Votre commande ne correspond pas à une fonction de COMMAND.TOS et aucun programme ou fichier batch ne correspond à ce nom (exemple: vous avez tapé 'TRUC', inconnu de COMMAND, et aucun fichier du catalogue courant ne s'appelle TRUC.BAT ou TRUC.PRG).

Destination is not a valid wild card expression.

Le nom du fichier destination (employé dans l'une des commandes COPY, MOVE ou REN) comporte des caractères génériques comme '*' ou '?' et il est différent de '.*', seul admis pour les noms de fichiers destination.

Directory not found.

Le nom de dossier indiqué après la commande CD n'a pas été trouvé sur le disque.

COMMAND: MESSAGES D'ERREUR

Disk full -- copy failed.

Lors d'une commande COPY ou MOVE, la totalité du fichier n'a pu être recopiée.

Done.

Ce message ne correspond pas à une erreur mais, au contraire, signale qu'une commande s'est correctement exécutée (REN, LS, DIR, COPY, MOVE, CHMOD, RM, DEL, ERA, INIT, PUTBOOT, GET, PUT, WRAP, NOWRAP).

Error creating file

La création du fichier destination n'a pu s'effectuer lors d'une commande COPY ou MOVE.

File not found.

Le fichier source n'a pas été trouvé (COPY, MOVE, REN, CHMOD, CAT, TYPE, RM, DEL, ERA) ou le dossier n'a pas été trouvé (LS, DIR)

Invalid mode specification.

L'argument 'mode' de la commande CHMOD manque ou bien a une valeur supérieure à 7.

... not found.

Le nom de fichier de redirection en entrée (<) ou de passage de paramètres (@) n'a pas été trouvé sur le disque courant.

Unable to change mode on subdirectorys or volumes.

Le fichier dont on désire changer le mode par la commande CHMOD est un dossier ou un nom de volume.

Unable to make directory

Une erreur s'est produite à la création d'un dossier par la commande MD (appeler la commande ERR pour obtenir le type d'erreur)

Unable to remove directory

La destruction d'un dossier n'a pu s'effectuer par la commande RD (appeler ERR pour connaître le numéro de l'erreur). Cette erreur se produit lorsque le dossier n'est pas vide.

Wild cards not allowed in destination.

Le fichier destination de la commande REN contient des caractères génériques (* ou ?) et n'est pas égal à '*. '*.

COMMAND: MESSAGES D'ERREUR

Wild cards not allowed in path name.

Un nom de chemin ne peut contenir de caractères génériques (* ou ?). Peut se produire avec les commandes COPY, MOVE ou REN.

ANNEXE BCODES D'ERREURS CPM-68K

La fonction 'perror' de la librairie 'GEMLIB' et la variable externe 'errno' déterminent la cause d'une erreur détectée. Le fichier <errno.h> contient les définitions symboliques des erreurs retournées.

NOTE: Ces codes d'erreurs ne sont reconnus que si la librairie 'GEMLIB' est utilisée. Ils n'ont aucun rapport avec les codes des erreurs GEMDOS.

Le tableau qui suit fournit le numéro d'erreur, le nom symbolique, et le message disponible pour la fonction 'perror'. Seuls sont indiqués les numéros valides sous CPM-68K.

2	ENOENT	No Such File
5	EIO	I/O Error
7	E2BIG	Arg List too Long
9	EBADF	Bad file Number
12	ENOMEM	Not enough core
13	EACCESS	Permission denied
22	EINVAL	Invalid argument
23	ENFILE	File table overflow
24	EMFILE	Too many open files
25	ENOTTY	Not a typewriter
27	EFBIG	File too big
28	ENOSPC	No space left on device
30	EROFS	Read only file system
35	ENODSPC	No directory space

GEM Resource Construction Set

CHAPITRE 1

Introduction à GEM RCS

GEM Resource Construction Set (RCS) est une application GEM utilisée pour créer les ressources (menus, boîtes de dialogues et d'erreur, etc.) pour les programmes d'applications écrits avec GEM. GEM RCS permet aussi d'incorporer dans les fichiers de ressources les icônes et les bit-images (images dont la précision est de l'ordre du bit, c'est-à-dire du pixel) créées grâce à l'Editeur d'icônes de GEM, GEM IconEdit.

Il n'est pas nécessaire d'être programmeur pour utiliser GEM RCS. Toutes les ressources d'une application peuvent être créées dans un fichier qui sera communiqué à un programmeur pour qu'il les intègre au programme.

Ce manuel comporte les chapitres suivants :

- * Le Chapitre 1, "Introduction à GEM RCS" explique ce qu'est une ressource, décrit l'écran de GEM RCS et les techniques d'utilisation de la souris avec GEM RCS, l'espace mémoire de GEM RCS et son utilisation de la mémoire.
- * Le Chapitre 2, "Tutoriel de GEM RCS" enseigne la terminologie et les techniques de base tout en construisant un menu et une boîte de dialogue.
- * Le chapitre 3, "Manuel de référence de GEM RCS" contient une description détaillée de toutes les caractéristiques de GEM RCS.
- * Le Chapitre 4, "RSCREATE et autres informations techniques" décrit RSCREATE, un programme utilitaire qui permet la modification et la portabilité des fichiers ressources. Ce chapitre décrit aussi comment chaîner des fichiers ressources et comment traiter les objets de type non connu. Il donne également quelques conseils permettant une utilisation plus rapide et plus rentable de GEM RCS.

La plupart des techniques utilisées avec GEM RCS, comme le choix d'une commande dans un menu 'descendant', sont identiques aux techniques utilisées avec le Bureau GEM. Il est nécessaire d'avoir lu le manuel du Bureau GEM et d'en avoir assimilé le fonctionnement de base avant de commencer à utiliser GEM RCS.

Qu'est-ce qu'une ressource ?

Dans une application GEM, une ressource est quelque chose qui apparaît sur l'écran (comme les menus, les boîtes de dialogue, etc.), mais qui ne fait pas partie du programme lui-même. Au lieu de cela, les ressources sont conservées dans un fichier séparé, ce qui comporte de nombreux avantages :

- * Comme nous l'avons indiqué, ce fichier peut être créé par un non-programmeur.
- * Le fichier ressource peut être modifié ou mis à jour (même par un non-programmeur) aussi souvent que nécessaire sans avoir à recompilier le code de l'application.
- * Une application peut être développée pour différents pays en utilisant le même code de programme et différents fichiers de ressources.
- * Une application peut fonctionner avec différents environnements systèmes, en utilisant les ressources générées à partir du même code source.

Les ressources sont constituées d'Objets. Un objet, dans ce cas-là, est un terme technique qui concerne une série d'images pouvant apparaître sur l'écran, parmi lesquelles une boîte vide, une boîte contenant un texte, une chaîne de caractères et bien d'autres choses.

Pour créer une ressource, un menu par exemple, il suffit de combiner les objets sous la forme d'un "arbre d'objets". La relation entre les objets d'un arbre s'effectue à l'aide de termes familiaux : le premier objet est le "parent", les objets contenus par le parent étant les "enfants".

Pour une description complète des objets et des arbres d'objets, consulter le GEM Application Environment Services Reference Guide (Manuel de Référence de GEM AES), Chapitre 6.

En utilisant GEM RCS, on peut créer les catégories suivantes d'arbres d'objets :

MENU Les caractéristiques des menus descendants d'une application GEM sont contenues dans le fichier ressource. Aucun menu ne peut occuper plus du quart de la taille de l'écran. Cela permet d'écrire la partie de l'écran cachée par le menu dans un tampon-mémoire spécial. Cela permet également à GEM AES de redessiner l'écran à partir de ce tampon-mémoire. Un tracé de

l'écran à partir du tampon mémoire MENU/ALERT est plus rapide qu'un tracé effectué par l'application.

DIALOG

Pour l'utilisateur, un DIALOG est le moyen, pour une application GEM, de fournir ou d'obtenir des informations, mais une application GEM peut utiliser des DIALOGs dans d'autres buts. Par exemple, la boîte à outils de GEM RCS (Figure 1-1) est un DIALOG dans le fichier ressource.

Les DIALOG n'utilisent pas le tampon-mémoire MENU/ALERT et peuvent de ce fait occuper plus du quart de la taille de l'écran. Les objets dans un DIALOG se "calent" aux limites des caractères afin de faciliter l'alignement des éléments du DIALOG.

PANEL

Un PANEL est similaire à un DIALOG, à la différence qu'il ne possède pas ce calage automatique aux limites du caractère. Un PANEL peut donc être utilisé pour toute chose qui nécessite un positionnement précis des objets. Par exemple, les boîtes d'éléments de l'écran de GEM RCS (Figure 1-1) sont des PANELs dans le fichier de ressource.

ALERT

Les ALERTs (une série spécialisée de DIALOGs) sont des moyens, pour une application GEM, d'afficher des messages d'information, de mise en garde ou d'erreur. Les ALERTs ont un format fixe et une taille limitée au quart de celle de l'écran. (Ils utilisent le tampon-mémoire MENU/ALERT dont nous avons déjà parlé.) Le format comprend une icône, une chaîne de 200 caractères ASCII (5 lignes de 40 caractères maximum), et trois options de sortie maximum. Chacune de ces options de sortie contient au plus 20 caractères.

FREE

Cette catégorie comporte les chaînes de caractères et les bit-images que l'on désire inclure dans une application, sans faire partie du programme lui-même. On peut ainsi changer le texte ou l'image sans modifier le programme, ni le recompiler.

Un exemple de ce type est une commande de menu, autre que la commande par défaut, dépendant du contexte d'utilisation. Par exemple, le Menu Global de GEM RCS comporte une commande dont l'état par défaut est 'Hide Parts'. La version par défaut de ce menu est contenue dans l'arbre MENU, tandis que les autres options ('Show Parts') sont contenues dans une chaîne de type FREE.

L'écran de GEM RCS

Lorsqu'on lance GEM RCS, on obtient un écran qui se présente comme le schéma ci-dessous. Les différentes légendes sont décrites après le schéma.

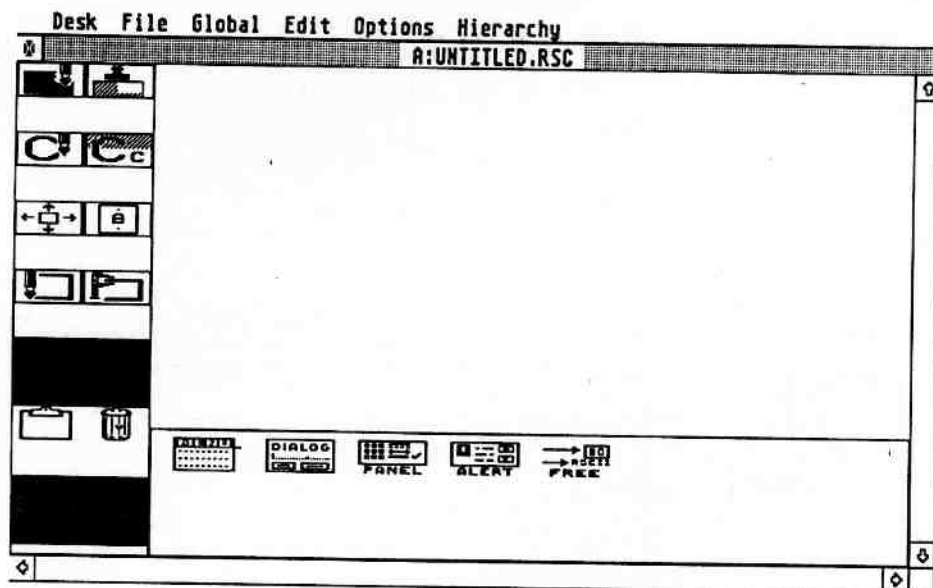


Figure 1-1 L'écran de GEM RCS

Barre de Menu (Menu Bar)

La barre de menu est la ligne supérieure de l'écran. Elle contient le titre des différents menus (File, Global, Edit, Options, Hierarchy) et RCS.

Lorsqu'on pointe le titre d'un menu, le menu correspondant s'affiche sous la barre de menu. On peut alors choisir une commande en cliquant son nom dans le menu affiché.

Les commandes du menu de GEM RCS sont décrites en détail dans le chapitre 3. La plupart des commandes peuvent aussi être exécutées en tapant l'une des combinaisons de touches listées dans la table 3-4.

Barre de titre (Title Bar)

La barre de titre indique quel est le niveau dans lequel on se trouve :

- . Niveau de base
- . Niveau objet

Le niveau de base (root level) est le niveau supérieur du fichier ressource. A ce niveau, la barre de titre identifie le chemin d'accès du répertoire courant et le nom du fichier ressource (par exemple C:\TOOLS\MYAPP.RCS) et la fenêtre de visualisation contient les icônes pour les arbres d'objets (MENU, DIALOG, etc.) qui ont été placés dans le fichier ressource.

Le niveau Objet est le niveau auquel on travaille sur les objets d'un arbre (par exemple, ajouter des chaînes de caractères, des entrées de commandes ou des options de sortie). A ce niveau, la barre de titre contient le nom de l'arbre (par exemple FILEMENU) sans aucune information de chemin d'accès.

Quelques règles sur le nom des fichiers ressources

- . Tous les fichiers ressources s'appellent UNTITLED.RSC jusqu'au moment où on les sauvegarde en leur donnant un autre nom.
- . L'extension d'un fichier ressource est RCS (ReSource). Ne pas la confondre avec le nom de l'application GEM RCS (Resource Construction Set).

Boîte de fermeture (Close Box)

- . Lorsqu'on se trouve au niveau Objet, la boîte de

fermeture ferme l'objet et revient au niveau de base du fichier ressource.

- Lorsqu'on se trouve au niveau de base, elle ferme le fichier ressource. Si des modifications ont été effectuées, mais non sauvegardées, GEM RCS affiche un DIALOG demandant si l'on souhaite abandonner les modifications ou les sauvegarder.

Note : Lorsqu'on travaille sur un nouveau fichier qui ne possède pas encore de nom et qui n'a pas encore été sauvé, la boîte de fermeture n'a aucun effet au niveau de base.

La boîte de fermeture et la commande Close du menu FILE peuvent être utilisées indifféremment.

Toolkit (boîte à outils)

La boîte à outils (décrite en détail dans le Chapitre 3) contient les icônes des différents outils utilisés pour personnaliser les objets de l'arbre. Par exemple, on peut changer la couleur d'un objet, son motif interne ou son alignement.

Fenêtre de visualisation (View Window)

La fenêtre de visualisation est l'endroit où l'on conçoit la structure d'un arbre et la personnalisation de ses objets.

Boîtes d'éléments (Parts Box)

Les boîtes d'éléments (décrites en détail dans le chapitre 3) contiennent les icônes des objets qui peuvent être incluses dans un arbre d'objets. Leur contenu change en fonction du type de l'arbre sur lequel on travaille.

Barres de défilement et glissières (Scroll bars and Sliders)

Les barres de défilement et les glissières de GEM RCS fonctionnent de la même manière que celles du Bureau GEM. Dans la plupart des cas, le défilement se fait aussi bien horizontalement que verticalement.

Techniques utilisées avec la souris

La plupart des techniques utilisées avec la souris pour GEM RCS, comme cliquer ou tirer une icône, sont identiques à celles utilisées avec le Bureau GEM ou d'autres applications GEM. Cependant, les effets de ces techniques sont souvent spécifiques à GEM RCS comme le montre la table ci-dessous.

Table 1-1 La souris et GEM RCS

Technique	Effet
Cliquer	Choix d'un arbre ou d'un objet. GEM RCS indique le choix d'un arbre en affichant son icône en inversion vidéo, et le choix d'un objet en définissant son "étendue" par une ligne pointillée.
Shift-cliquer	Choix de plusieurs objets.
Ctrl-cliquer	Choix du <u>parent</u> d'un objet.
Double-cliquer	Ouvre un arbre ou un objet en vue d'une modification.
Tirer	Copie un arbre ou un objet depuis les boîtes d'éléments dans la fenêtre de visualisation. Déplace un arbre ou un objet dans la fenêtre de visualisation. Déplace un arbre ou un objet, depuis ou vers la tablette (Clipboard).
Ctrl-tirer	Déplace le parent d'un objet et tous les enfants de ce parent.
Shift-tirer	Copie un arbre ou un objet dans la fenêtre de visualisation. Copie un arbre ou un objet, depuis ou vers la tablette.
Ctrl-Shift-tirer	Copie le parent d'un objet et tous les enfants de ce parent.

Note : Lors de l'utilisation des techniques de traction pour déplacer ou copier un arbre ou un objet, prendre en compte les éléments suivants :

* Lorsque l'on copie depuis les boîtes d'éléments ou lors de copie ou du déplacement dans une fenêtre de visualisation, la ligne pointillée de l'arbre ou de l'objet doit être entièrement dans la fenêtre de visualisation ou un objet parent. Voici deux exemples :

- . Tirer une icône d'arbre des boîtes d'éléments vers la fenêtre de visualisation : si une partie de la ligne se trouve dans la barre de titre ou la boîte à outils lors du relâchement du bouton de la souris, l'arbre n'est pas recopié dans la fenêtre de visualisation.
- . Tirer un arbre de MENU depuis les boîtes d'éléments vers la boîte de menu dans une fenêtre de visualisation : Lorsqu'une partie de la ligne de l'entrée est en dehors de la boîte du menu (son parent) au moment de relâcher le bouton de la souris, l'entrée n'est pas copiée dans la boîte du menu.

* Lors de la copie ou du déplacement d'un arbre ou d'un objet vers la tablette, toujours tirer depuis le coin supérieur gauche de l'arbre ou de l'objet. Dans le cas contraire, si une partie de la ligne est en dehors du bord gauche de la boîte à outils, l'arbre ou l'objet ne sera pas copié sur la tablette. (La tablette est décrite en détail dans le Chapitre 3.)

Noter également que lors d'une copie d'un arbre ou d'un objet, la copie ne conserve pas les noms d'origine associés aux arbres ou aux objets.

Définition de la taille d'un objet

Pour changer la taille d'un objet, sélectionner d'abord cet objet. GEM RCS affiche l'objet choisi avec une ligne pointillée et, dans le coin inférieur droit, un rectangle noir qui représente la taille de l'objet (le contrôleur de taille). Placer le pointeur sur ce rectangle et le tirer jusqu'à ce que l'objet ait la taille désirée.

Si l'objet possède des enfants, GEM RCS n'autorise pas une taille qui l'empêcherait de contenir ses enfants.

Suppression d'arbres ou d'objets

Pour supprimer un arbre ou un objet, utiliser l'une des méthodes suivantes :

* Le tirer hors de la fenêtre de visualisation vers l'icône de la corbeille située dans la boîte à outils. Lorsque la corbeille

apparaît en inversion vidéo, relâcher le bouton de la souris.

- * Cliquer sur l'arbre ou l'objet pour le sélectionner. Déplacer le pointeur vers l'icône de la corbeille (sans tirer l'arbre ou l'objet), puis cliquer sur la corbeille.
- * Sélectionner l'arbre ou l'objet, puis choisir la commande DELETE dans le Menu EDIT.

Une fois supprimée, un item placé dans la corbeille ne peut plus être récupéré.

Espace mémoire et mémoire utilisée.

Lors du lancement de GEM RCS, le programme nécessite une partie de la mémoire de l'ordinateur comme zone de travail. En fonction de la mémoire vive disponible, cette zone de mémoire peut atteindre 64K octets.

Au fur et à mesure de l'ajout d'arbres ou d'objets, la zone de travail disponible se réduit. Cependant, la suppression d'un arbre ou d'un objet ne permet pas de récupérer la place qu'il occupait. Le seul moyen pour récupérer cette place consiste à sauvegarder le fichier. Les commandes SAVE ou SAVE AS (décrites dans le chapitre 3) permettent de sauvegarder le fichier pour ensuite le recharger.

Pour connaître la taille de la zone de mémoire disponible, utiliser la commande INFO du menu Options. Le DIALOG d'information indique la place occupée par le fichier, les arbres, les objets et la place disponible.

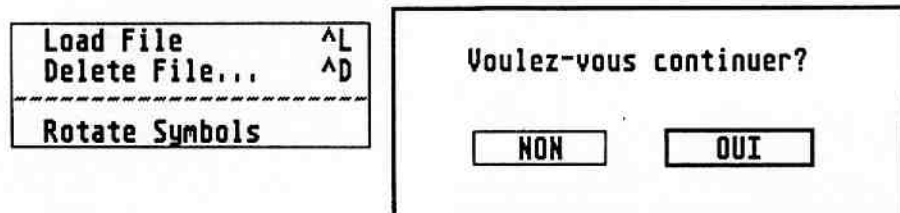
Note : La taille du fichier indiquée n'est qu'une approximation de la taille du fichier ressource pour les raisons suivantes :

- * Les ALERTS prennent moins de place dans le fichier final puisqu'ils sont écrits sous forme de chaînes ASCII.
- * Les copies d'objets faites avec Shift-tirer peuvent utiliser plus de place dans le fichier final.

CHAPITRE 2

Tutoriel de GEM RCS

Dans l'introduction du tutoriel de GEM RCS, nous allons lancer GEM RCS, puis créer le menu et la boîte de dialogue ci-dessous :



Puis nous sauvegarderons le fichier ressource et sortirons de GEM RCS.

Lancer GEM RCS

Pour lancer GEM RCS, lancer le Bureau GEM, puis :

1. Ouvrir le dossier TOOLS et rechercher l'icône RCS.APP.
2. Effectuer un double cliquage sur l'icône RCS.APP.

Créer un menu

Le menu que nous allons créer dans les étapes suivantes contient des commandes de texte simples. Cette partie du tutoriel montre l'ouverture d'un arbre, l'ajout d'objets à l'arbre, la définition d'attributs et le tri d'objets.

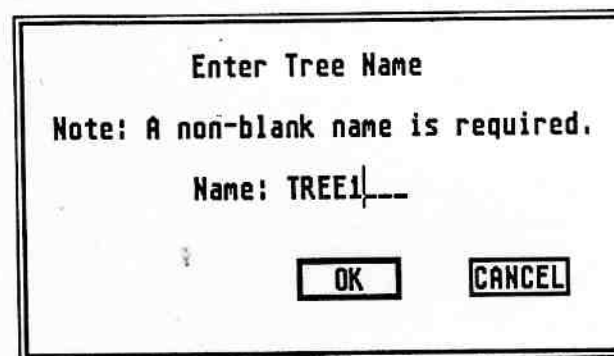
Chgisir et nommer une icône d'objet de MENU

Les boîtes d'éléments contiennent les icônes des cinq sortes d'arbre que l'on peut utiliser avec GEM RCS.



Tirer l'icône du MENU dans la fenêtre de visualisation. Il n'est pas utile de positionner cette icône précisément : au moment de relâcher le bouton de la souris, GEM RCS place automatiquement une copie de l'icône dans le coin supérieur gauche de la fenêtre. (Bien se souvenir que GEM RCS ne copie pas l'icône dans la fenêtre de visualisation si une partie de la ligne que l'on tire se trouve dans la boîte à outils ou la barre de titre.)

A chaque fois que l'on place un nouvel arbre dans la fenêtre de visualisation, GEM RCS affiche un DIALOG qui demande de lui donner un nom :



Noter que le DIALOG contient déjà le nom TREE1. Pour utiliser ce nom pour l'arbre, il suffit de cliquer sur l'option de sortie OK ou d'appuyer sur la touche Enter. Ici, cependant, nous allons donner un nom différent à l'arbre. Appuyer sur la touche Esc pour effacer TREE1, taper le nom APPMENUS, puis cliquer sur la case OK (ou presser la touche Enter). (Consulter le Manuel du Bureau GEM pour une description complète de la saisie et de la modification des textes dans un DIALOG.)

Ouvrir l'arbre

Faire un double cliquage sur l'icône APPMENUS. Les changements suivants interviennent :

- * La barre de titre indique maintenant APPMENUS.
- * La fenêtre de visualisation contient maintenant une barre de menu avec deux titres de menus (FILE et DESK) et une zone ombrée sur laquelle les menus seront affichés.
- * Les boîtes d'éléments comportent maintenant les quatre objets qui peuvent apparaître dans un arbre de MENU : le TITRE (TITLE)

du menu, l'ENTREE (ENTRY : chaque chaîne de commande est une entrée), un motif "ligne" pour séparer les différentes entrées et une boîte dans laquelle seront placés les items du menu qui ne sont pas des textes (motifs de remplissage ou lignes par exemple).

Dans ce tutoriel, le menu est créé depuis le début. Dans le chapitre 3, nous décrirons comment utiliser les menus FILE et DESK existants.

Ajouter un titre de Menu

Tirer l'objet TITLE depuis les boîtes d'éléments vers la barre de menu pour la placer après "File". Bien faire attention de ne pas le tirer dans la barre de titre ou la boîte à outils. Sinon, GEM RCS efface ce qui a été tiré.

Cliquer deux fois sur TITLE. Le DIALOG suivant apparaît sur l'écran :

Edit Unformatted String Object

Text: STRING-----

^ ^ ^

10 20 30 characters

(Optional) Object Name: -----

Respecter les étapes suivantes pour modifier et donner un nom au titre du menu :

1. Appuyer sur la touche Esc pour effacer le texte courant.
2. Appuyer sur la barre d'espacement, taper Options, puis appuyer à nouveau sur la barre d'espacement. (Lorsque l'on ne presse pas la barre d'espacement, il n'y a pas assez d'espaces entre le titre des différents menus.)
3. Déplacer le curseur dans le champ du Nom de l'Objet (Object Name) et taper OPTITLE.
4. Cliquer la case Ok (ou appuyer sur Enter).

Ajouter des entrées dans le Menu

Pour ajouter des entrées (commandes) au Menu, respecter les étapes suivantes :

1. Cliquer sur le titre du menu Options pour sélectionner le titre et afficher une petite boîte de menu sous la barre de menu.
2. Placer la pointe de la flèche juste à l'intérieur du coin inférieur droit de la boîte du menu et appuyer sur le bouton de la souris. Lorsque l'icône représentant un doigt apparaît, tirer vers le coin inférieur droit afin d'agrandir la boîte du menu. Pour le moment, il est préférable de la faire la plus grande possible. Sa taille sera ajustée ultérieurement.
3. Tirer ENTRY depuis les boîtes d'éléments et la placer dans le coin supérieur gauche de la boîte de menu. Ne pas trop la coller dans le coin. Bien s'assurer que la ligne que l'on tire est entièrement à l'intérieur de la boîte du menu.
4. Copier cette entrée à l'aide de Shift-tirer. Placer la copie juste en-dessous de la première entrée.
5. Tirer une ligne de séparation depuis les boîtes d'éléments et la placer juste en-dessous des deux entrées.
6. Copier une autre entrée à l'aide de Shift-tirer. Placer la nouvelle copie sous le trait de séparation.

Modifier les entrées

La prochaine étape consiste à convertir chaque entrée en une commande que l'utilisateur puisse choisir. Pour cela, respecter les étapes suivantes :

1. Cliquer deux fois sur la première entrée du menu. GEM RCS affiche le DIALOG de Modification des Objets Non Formattés (Edit Unformatted String Object).
2. Appuyer sur la touche Esc pour effacer le texte courant.
3. Appuyer sur la barre d'espacement deux fois (pour mettre deux espaces devant la commande) puis taper Load File.... (Les trois points sont une convention de GEM indiquant que lorsqu'on choisit cette commande, un DIALOG est affiché.) Ne pas appuyer sur la case OK pour le moment.
4. La plupart des applications de GEM utilisent des équivalences de touches pour les commandes de menu. Par exemple, l'utilisateur peut taper Ctrl-L pour produire le même effet que la commande Load File.... Pour indiquer cela dans le menu, appuyer sur la barre d'espacement plusieurs fois, puis taper ^L suivi d'un seul espace.

Note : Les menus des applications GEM suivent les conventions suivantes : ^ représente la touche Ctrl et un losange plein la touche Alt. (La combinaison Ctrl-G produit un losange plein dans le DIALOG Edit Unformatted String Object.) Les menus des applications de GEM utilisent aussi la convention qui stipule que le texte d'une commande est précédé de deux espaces et suivi d'un espace.

5. Déplacer le curseur dans le champ Object Name et taper OPSLOAD.

6. Cliquer sur la case OK ou appuyer sur la touche Enter.

Suivre les mêmes étapes pour la seconde entrée avec les éléments suivants :

- * La commande est Delete File...
- * L'équivalent est la combinaison de touches ^D
- * Le nom de l'objet est OPSDELT.

Ne pas oublier les espaces avant et après la commande. Ne pas s'inquiéter si ^L et ^D ne sont pas correctement alignés dès la première fois. On peut corriger cela en ajoutant ou retirant des espaces dans le DIALOG Edit Unformatted String Object. Les techniques de modification sont les mêmes que pour tous les DIALOGs des applications GEM.

Faire la même chose pour la troisième entrée :

- * La commande est Rotate Symbols.
- * Il n'y a pas de combinaison de touches équivalente
- * Le nom de l'objet est OPSROTE.

Définir la taille de la boîte du menu

Pour ajuster la taille de la case de menu, sélectionner la boîte en cliquant à l'intérieur, assez loin des commandes ou des lignes de séparation. Le cliquage est correct lorsqu'apparaît le rectangle dans le coin inférieur droit de la boîte.

Tirer ensuite ce rectangle jusqu'à l'obtention de la taille désirée.

On peut aussi ajuster la taille de la boîte du menu sans la sélectionner au départ. Il suffit d'utiliser la même technique que celle décrite à l'étape 2 de "Ajouter des entrées dans le Menu".

Ajuster la taille des commandes

Puisque GEM AES n'éclaire et n'autorise que la zone contenue dans

la zone d'une entrée, il est nécessaire d'ajuster la taille de chaque commande pour qu'elle occupe toute la largeur de la boîte de menu. Dans le cas contraire, les menus d'une application auraient un aspect visuel et fonctionnel que l'utilisateur trouverait certainement déplaisant.

Pour ajuster la taille d'une commande, utiliser l'une des méthodes suivantes :

- * Choisir la commande, tirer son contrôleur de taille jusqu'au bord droit de la boîte de menu.
- * Choisir la commande, afficher le menu d'alignement (voir ci-dessous) et choisir l'option Fill Horizontal.



- * Dans le champ Text de l'Edit Unformatted String Object, ajouter des espaces à la fin de chaque texte d'entrée.

Ajuster la longueur de la ligne de séparation

Pour que la ligne de séparation occupe toute la largeur de la boîte de menu, respecter les étapes suivantes :

1. Effectuer un double cliquage sur la ligne de séparation. GEM RCS affiche le DIALOG Edit Unformatted String Object. Les tirets épais dans le champ Text sont équivalents à la ligne de séparation ombrée du menu.
2. Pour allonger la ligne, taper plusieurs fois Ctrl-S dans le champ Text.
3. Cliquer sur la case OK ou presser la touche Enter.

Si la ligne n'est pas encore assez longue, répéter ces étapes. Lorsque la ligne est trop longue, GEM RCS affiche un message qui indique que cet objet ne peut être contenu dans son parent. Cliquer sur la case OK, puis à nouveau cliquer deux fois sur la ligne de séparation. Utiliser la touche Backspace pour effacer les tirets supplémentaires jusqu'à l'obtention de la longueur désirée.

Classement des objets dans un menu

La dernière chose à faire avant de fermer un arbre est de trier les objets qu'il contient. Dans le cas contraire, au moment de son exécution, l'application dessinera les objets à chaque niveau dans l'ordre où ils ont été créés, ce qui peut ne pas être agréable visuellement. En triant les objets, on peut déterminer dans quel ordre ils seront dessinés.

Pour trier les objets de l'arbre de menu, respecter les étapes suivantes :

1. Ctrl-cliquer sur l'une des commandes pour sélectionner la boîte du menu.
2. Afficher le menu **Hierarchy** et choisir la commande **Sort Children...** GEM RCS affiche un DIALOG qui comporte quatre options de tri : single-row, single-column, double-column, double-row.
3. Choisir l'option single-column (la deuxième en partant de la gauche) et appuyer sur la touche Enter ou cliquer la case OK.

Fermeture de l'arbre des objets du Menu

Le menu est maintenant terminé. Pour en fermer l'arbre des objets et revenir au niveau de base du fichier ressource, cliquer sur la case de fermeture ou choisir la commande Close du menu FILE.

Créer une boîte de dialogue

La boîte de dialogue que nous allons créer dans les étapes suivantes contient une chaîne de caractères, une série de "boutons radio", et deux options de sortie. De plus, pour en montrer encore plus sur la définition des attributs et le tri des objets, cette partie du tutoriel illustre une tâche avec un arbre à 3 niveaux, où l'un des enfants d'un objet de base possède lui-même des enfants.

Choisir et nommer une icône d'objet de DIALOG

La première étape de la création d'une boîte de dialogue est la même que celle que nous avons utilisée pour la création d'un menu : il faut choisir et donner un nom à l'icône de l'objet.

Tirer l'icône du DIALOG depuis la boîte d'éléments dans la fenêtre de visualisation. Lorsque GEM RCS affiche un DIALOG qui demande de lui donner un nom, appuyer sur la touche Esc pour effacer le nom par défaut (ici TREE2, TREE1 était utilisé pour le MENU), taper PTYDIAL et cliquer sur la case OK (ou presser la touche Enter).

Ouvrir l'arbre

Pour ouvrir l'arbre de la boîte de dialogue, faire un double cliquage sur l'icône PTYDIAL. Remarquer les changements suivants qui interviennent sur l'écran :

* Le nom de l'arbre, PTYDIAL, apparaît dans la barre de titre, en haut de la fenêtre de visualisation.

- * La fenêtre de visualisation est vide.
- * Les boîtes d'éléments ont changé et comportent maintenant les objets qui peuvent apparaître dans une boîte de dialogue.

Choisir des objets pour la boîte de dialogue

Pour assembler les objets de la boîte de dialogue, respecter les étapes suivantes :

1. Tirer l'objet STRING depuis les boîtes d'éléments pour le placer dans le coin supérieur gauche de la fenêtre de visualisation.
2. Tirer la boîte vide (voir ci-dessous) de la boîte d'éléments pour la placer au centre de la ligne supérieure de la fenêtre de visualisation.



3. Cliquer sur la boîte pour la sélectionner. Remarquer le contrôleur de la taille des objets (décrit dans le Chapitre 1, "Définir la taille des objets") dans le coin inférieur droit.
4. Tirer la boîte du contrôleur de taille un petit peu vers le bas et la rapprocher, du bord droit de la fenêtre de visualisation afin d'obtenir un rectangle long de faible hauteur.
5. Tirer l'objet BUTTON de la boîte d'éléments pour le positionner dans le coin supérieur gauche du rectangle qui vient d'être élargi.

Personnaliser l'objet STRING

Effectuer un double-cliquage sur l'objet STRING dans la fenêtre de visualisation. Lorsque GEM RCS affiche le dialogue Edit Unformatted String Object, appuyer sur la touche Esc pour effacer le texte courant, puis taper :

Voulez-vous commencer?

Etant donné qu'il n'y a pas de nom d'objet, cliquer sur la case OK ou appuyer sur la touche Enter. L'objet STRING de la boîte de dialogue contient maintenant le texte qui vient d'être saisi. (Si le message déborde de la boîte ou sur l'objet BUTTON, ne pas en tenir compte : nous repositionnerons l'ensemble ultérieurement.)

Personnaliser l'Objet BUTTON

Pour changer le texte situé à l'intérieur de l'objet BUTTON et lui donner un nom auquel on pourra se référer dans le programme, respecter les étapes suivantes :

1. Effectuer un double cliquage sur le bouton. GEM RCS affiche le dialogue Edit Unformatted String Object.
2. Appuyer sur la touche Esc pour effacer le texte courant.
3. Taper OUI.
4. Cliquer sur le champ "Object Name". Le curseur se déplace vers cette ligne. (Il est également possible de déplacer le curseur en appuyant sur la touche Tab ou la touche Curseur-bas.)
5. Taper OUIBTN. (Pour rendre le programme le plus lisible possible, utiliser des noms d'objet qui indiquent la fonction de cet objet.)
6. Cliquer la case Ok (ou appuyer sur Enter).

Définir un Bouton radio

La case "OUI" et la suivante (qui va être prochainement créée) forment une série de "boutons radio". Tout comme les poussoirs d'un autoradio, les boutons radio possèdent les caractéristiques suivantes :

- * Dans une série, un seul bouton est sélectionné à un instant donné, mais il y a toujours un bouton de sélectionné. Il ne peut y avoir de cas où aucun bouton n'est sélectionné.
- * Sélectionner un bouton dé-sélectionne automatiquement le bouton précédent.

Pour définir le bouton comme un bouton radio, respecter les étapes suivantes :

1. Cliquer sur le bouton "OUI" pour le sélectionner.
2. Cliquer sur l'icône du menu des attributs (voir ci-dessous) et cliquer sur le "Bouton radio" du menu qui apparaît.

Créer des boutons radio supplémentaires

Ce dialogue possède un bouton radio supplémentaire, "NON". Pour créer ce bouton, respecter les étapes suivantes :

1. Shift-tirer le bouton "OUI" deux fois, pour placer les deux copies à droite de l'original.
2. Effectuer un double cliquage sur la copie immédiatement à droite de l'original. Dans le dialogue Edit Unformatted String Object, changer le texte en "OUI" et le nom de l'objet en "NONBTN".

Concevoir le bouton de radio sélectionné

Lorsque le dialogue apparaît au moment de l'exécution, un des boutons de radio doit être pré-sélectionné (affiché en inversion vidéo). Pour pré-sélectionner le bouton "OUI", respecter les étapes suivantes :

1. Cliquer sur la case pour la sélectionner.
2. Cliquer sur l'icône des attributs et choisir le mot "Selected".

Personnalisation de la boîte Parent

Avant de travailler avec la boîte Parent, il faut d'abord savoir pourquoi elle est nécessaire.

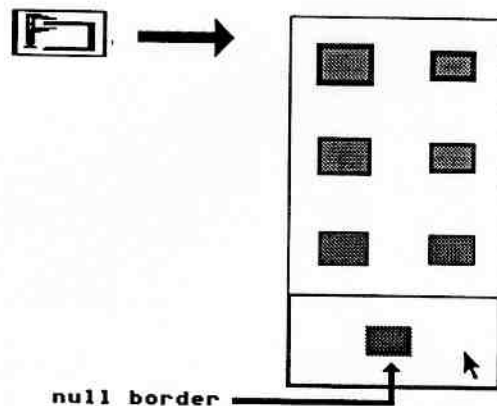
Lorsqu'on possède plusieurs jeux de boutons radio dans un dialogue, chaque jeu doit être compris dans une boîte parent. Dans le cas contraire, cliquer sur un bouton désactive tous les autres boutons du dialogue, et pas seulement ceux du jeu actif.

Dans notre dialogue qui ne comporte qu'un seul jeu de boutons radio, il n'est pas réellement nécessaire de posséder une boîte parent, mais le dialogue peut avoir d'autres jeux comme "Groupes par âge" et "Tranche de salaire". Cette étape montre ce qu'il faut faire lorsqu'il faut utiliser une boîte parent.

Tout d'abord, disposer les boutons radio dans la boîte parent. (Le bouton "OUI" doit être dans le coin supérieur gauche de la boîte.) Sélectionner alors la boîte parent (si elle n'est pas déjà sélectionnée) pour la rendre aussi petite que possible.

Dans le dialogue final, la boîte parent ne doit pas être visible. Pour la rendre invisible, respecter les étapes suivantes :

1. Sélectionner la boîte (si elle n'est pas sélectionnée)
2. Cliquer sur le côté de l'icône de menu (voir ci-dessous)
3. Dans ce menu, choisir le "cadre nul" (illustration en page suivante).



Lorsqu'on dé-sélectionne la boîte parent en cliquant partout ailleurs dans la fenêtre de visualisation, on constate que la boîte parent est maintenant invisible. Cependant, on peut la re-sélectionner en faisant Ctrl-cliquer sur l'un des boutons radio. On peut la déplacer (avec les boutons radio) en faisant Ctrl-tirer sur l'un des boutons.

Créer des options de sortie

Le dialogue nécessite deux options de sortie pour que l'utilisateur puisse quitter le dialogue en cliquant sur l'un ou l'autre des boutons.

Pour créer ces options (boutons) de sortie, respecter les étapes suivantes :

1. Tirer un objet BUTTON depuis les boîtes d'élément et le placer en-dessous des boutons radio.
2. Shift-tirer ce bouton, pour placer une copie à côté de l'original.
3. Modifier et définir les attributs pour le premier bouton :
 - . Changer son texte en "OK".
 - . Lui donner pour nom OKBTN.
 - . Définir ses attributs à "Exit" et "Default".

"Default" signifie que l'utilisateur peut, pour cette option, soit cliquer la case, soit presser la touche Enter. (La case OK de l'Edit Unformatted String Object que nous avons déjà utilisée possède cette option par défaut.)

4. Faire de même pour la seconde option de sortie :
 - . Changer son texte en "Cancel"
 - . Lui donner pour nom CANCLBTN.
 - . Définir ses attributs à "Exit".
5. Enfin, positionner ces deux cases là où on le désire.

Modifier la taille de l'arrière-plan de la boîte de dialogue

Déplacer le pointeur vers une zone libre de la fenêtre de visualisation (loin du texte et des boutons) et cliquer. Le contrôleur de taille de la boîte de dialogue apparaît dans le coin inférieur droit de la fenêtre de visualisation.

Placer le pointeur sur le contrôleur de taille et tirer vers le haut et la gauche. Au fur et à mesure que l'on tire, le fond de la boîte de dialogue s'amenuise. Lorsque ce fond possède la taille désirée, relâcher le bouton de la souris.

Classer les objets dans le dialogue

La dernière chose à faire avant de fermer cet arbre est de trier les objets qu'il contient. La procédure est essentiellement la même qu'avec un dialogue simple.

Note : Il est nécessaire de classer tous les niveaux (séries) d'enfants. Par exemple, ce dialogue comporte deux niveaux. Les boutons radio sont les enfants d'une boîte vide, et la boîte vide, la chaîne de caractères et les options de sortie sont les enfants de la boîte de dialogue extérieure.

Pour trier les enfants de ce dialogue, respecter les étapes suivantes :

1. Ctrl-cliquer sur l'une des boutons radio pour sélectionner son parent.
2. Afficher le menu Hierarchy et choisir la commande Sort Children.... Choisir l'option single-row (la première en partant de la gauche) et appuyer sur la touche Enter ou cliquer la case OK.
3. Sélectionner la boîte de dialogue extérieure.
4. Afficher le menu HIERARCHY et choisir à nouveau la commande Sort Children.... Choisir l'option double-row (la dernière à droite) et appuyer sur la touche Enter ou cliquer la case OK.

Classer les objets permet de les dessiner dans l'ordre suivant :

1. Chaîne "Parti politique?"
2. Les boutons radio, "Démocratique" en premier
3. L'option OK

4. L'option Cancel.

Fermeture de l'arbre des objets du Dialogue

Le DIALOG est maintenant terminé. Pour en fermer l'arbre des objets et revenir au niveau de base du fichier ressource, cliquer sur la case de fermeture ou choisir la commande Close du menu FILE.

Sauvegarder un fichier ressource

Afficher le menu FILE et choisir la commande Save as.... Le sélecteur d'Item apparaît sur l'écran.

Taper TEST, puis cliquer sur la case OK ou appuyer sur la touche Enter. GEM RCS sauvegarde alors les fichiers suivants :

TEST.RSC Le fichier ressource courant qui fait partie du programme d'application. Il n'est pas utile de saisir l'extension, GEM RCS la rajoutant automatiquement.

TEST.DFN Fichier auxiliaire qui identifie les arbres et les objets du fichier ressource. Lorsqu'on travaille avec GEM RCS, il est indispensable que le fichier de ressource et son fichier .DFN soient tous deux dans le même répertoire.
Les fichiers créés avec les premières versions de GEM RCS possèdent l'extension .DEF. Pour pouvoir les utiliser avec les versions récentes de GEM RCS, changer l'extension .DEF en .DFN.

Dans sa configuration initiale, GEM RCS crée également TEST.H, un fichier en langage C que l'on doit utiliser au moment de la compilation du programme. Le fichier .H est un fichier ASCII qui liste les arbres et les objets du fichier ressource et les numéros des objets qui leur ont été assignés par GEM RCS. Ce fichier peut également être lu pour savoir dans quel ordre les objets ont été classés. Le classement des objets, dont nous avons parlé dans ce tutoriel, sera décrit plus en détail dans la description du menu HIERARCHY, chapitre 3.

Note : On peut également produire des fichiers similaires pour Pascal, BASIC et FORTRAN-77, ainsi qu'une version ASCII du fichier ressource (fichier .RSH). Consulter la description des commandes Output... et Save Preferences du menu GLOBAL (chapitre 3) et la description de RSCREATE (chapitre 4).

Quitter GEM RCS

Pour interrompre GEM RCS, utiliser la commande Quit du menu File. On se retrouve alors sous contrôle du Bureau GEM.

CHAPITRE 3

Manuel de Référence de GEM RCSRapport d'aspect et Résolution d'écran

Les applications GEM peuvent fonctionner sous divers rapports d'aspects et résolutions d'écran. La table ci-dessous liste les combinaisons les plus courantes :

Table 3-1 Rapport d'aspect et résolution d'écran

Résolution	Nombre de pixels	Rapport d'aspect
Basse	640 x 200	3,2 / 1
Haute (*)	640 x 400	1,6 / 1
Haute (*)	720 x 350	2,06 / 1
Carrée (*)	indéfini	1 / 1

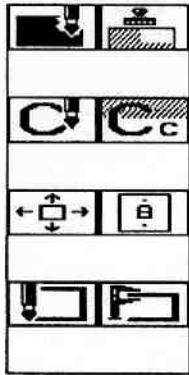
(*) Bien que le rapport soit différent, les applications GEM utilisent le même écran pour toutes ces résolutions.

Lorsqu'une icône ou une bit-image est affichée sur un système qui possède un rapport d'aspect différent de celui sur lequel il a été créé, l'icône ou la bit-image apparaît plus grande ou plus petite qu'à l'origine. Cela peut occasionner des problèmes lorsque les icônes ou les bit-images ont été placées très près les unes des autres ; les différences de rapport d'aspect peuvent provoquer un chevauchement ou un éloignement.

Ce problème peut être résolu par l'une des méthodes suivantes :

- * On peut créer un seul fichier ressource et compenser les différents rapports d'aspects du système sur lequel l'application est exécutée.
- * On peut créer différents jeux d'icônes et de bit-images pour chaque rapport d'aspect. On peut ainsi créer des fichiers ressources associés à chaque rapport d'aspect à partir d'un seul fichier "maître" en chargeant le jeu approprié d'icônes et de bit-images dans chaque version du fichier ressource final.

Le schéma ci-dessous montre une partie de la boîte à outils de GEM RCS qui a été créée en haute résolution. Remarquer qu'en haute résolution, les bit-images sont écrasées par rapport à la basse résolution.



GEM RCS	GEM RCS	GEM Paint	GEM Paint
basse résolution	haute résolution	basse résolution	haute résolution

Lorsque l'on crée des fichiers ressources adaptés à certains rapports d'aspect, l'application peut déterminer, au moment de son exécution, le rapport d'aspect de l'écran et sa résolution, puis charger le fichier approprié.

Nous reparlerons plus loin dans ce chapitre des icônes et des bit-images.

Note : Les différences de rapport d'aspect ne provoquent aucun problème pour les textes et autres objets dans un arbre.

Les menus FILE et DESK

Comme nous l'avons indiqué dans le tutoriel, lors de l'ouverture d'un arbre d'objets de menus, les menus FILE et DESK existent déjà dans la barre de menu.

Le menu FILE

Le menu FILE est optionnel ; on peut très bien s'en débarrasser en tirant son titre dans la corbeille.

Pour ajouter des commandes dans le menu FILE, respecter les étapes suivantes :

1. Cliquer sur le titre du menu pour en sélectionner le titre et afficher la boîte du menu qui ne contient qu'une seule commande, Quit.
2. Placer le pointeur dans le coin inférieur droit de la boîte du menu et faire Ctrl-tirer vers le bas et la droite pour agrandir la boîte du menu, sans affecter la taille ou l'emplacement de la commande.
3. Tirer la commande Quit vers le bas ou la placer à l'endroit désiré.

A partir de ce moment, on peut créer le menu comme cela a été expliqué dans le tutoriel.

Le menu DESK

Ce menu est indispensable ; il ne faut pas en tirer le titre vers la corbeille. Cependant, lors de l'exécution du programme, son nom et son emplacement peuvent varier comme suit :

- * Lorsque l'application est exécutée sous une version 1.X de GEM AES, le menu appelé DESK apparaît sur le côté gauche de la barre de menu.
- * Lorsque l'application est exécutée sous la version 2.X de GEM AES, le menu apparaît sur le côté droit de la barre de menu, et son nom est le nom du fichier exécutable de l'application. Par exemple, pour une application dont le nom du fichier est BINGO.APP, le nom du menu est BINGO.

Les différences de nom et d'emplacement sont totalement gérées par GEM AES.

Le menu DESK contient plusieurs commandes "inamovibles". La première "Your message here" peut être modifiée (le message peut contenir jusqu'à 20 caractères), mais ne peut pas être effacée. L'utilisation la plus courante de cette commande est l'affichage d'un message d'information sur l'application, comportant le numéro de version.

Les nombres situés sous la ligne de séparation sont des places réservées pour des accessoires de bureau. Le menu peut en accepter 6 au maximum. On ne peut modifier ou supprimer ces entrées. GEM AES gère l'emplacement de ces noms d'accessoires dans le menu et fixe la taille du menu en fonction du nombre d'accessoires qu'il contient.

Utiliser les boîtes d'éléments

Les boîtes d'éléments contiennent des icônes pour les objets qui peuvent être inclus dans un arbre. Ces objets possèdent des attributs pré-définis (motifs de remplissage, couleurs, chaînes de caractères, etc.) qui peuvent être modifiés avec les outils disponibles dans la boîte à outils. On peut modifier un texte dans un objet ou effectuer d'autres modifications en effectuant un double-clic sur l'objet pour afficher l'un de ses nombreux dialogues.

Pour placer un objet dans l'arbre courant, tirer son icône depuis la boîte d'éléments jusqu'à la fenêtre de visualisation.

Le contenu de la boîte d'éléments dépend du type d'arbre que l'on crée.

Objets de l'arbre MENU

Objet 1 : Titre du menu à inscrire dans la barre de menu.

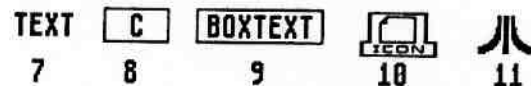
Objet 2 : Chaîne de caractères de commande pour le menu.

Objet 3 : Ligne de séparation pour le menu. Utiliser la ligne de séparation pour diviser les commandes en groupes logiques pour le confort de l'utilisateur.

Objet 4 : Boîte vide que l'on place dans un menu descendant. On peut utiliser cette boîte vide pour définir l'extension choisie (en inversion vidéo) d'une bit-image d'un menu. Par exemple, voir le menu GALLERY de GEM GRAPH.

Objets des arbres DIALOG et PANEL

La même série d'objets est utilisée pour les arbres DIALOG et PANEL. Les noms entre parenthèses sont les types d'objets décrits dans le chapitre 6 du Manuel de Référence de GEM AES.



Objet 1 : (G_BUTTON) - Chaîne contenue dans une boîte que l'utilisateur sélectionne pour indiquer un choix parmi plusieurs options.

Objet 2 : (G_STRING) - Chaîne de caractères hors boîte, contenant habituellement un texte d'explication destiné à l'utilisateur.

Objet 3 : (G_FTEXT) - Champ de texte formaté que l'utilisateur peut modifier.

Objet 4 : (G_FBOXTXT) - Champ de texte formaté inclus dans une boîte. L'utilisateur peut changer ce champ texte.

Objet 5 : (G_IBOX) - Boîte vide au travers de laquelle l'utilisateur peut voir le motif de remplissage ou le texte en dessous.

Objet 6 : (G_BOX) - Boîte opaque (non transparente).

Objet 7 : (G_TEXT) - Texte formaté. On peut choisir la taille, la couleur, la police de caractères et les masques de ces objets.

Objet 8 : (G_BOXCHAR) - Caractère unique dans une boîte opaque.

Objet 9 : (G_BOXTEXT) - Idem à TEXT, mais entouré par une boîte opaque.

Objet 10 : (G_ICON) - Utilisé pour afficher une icône (donnée plus masque) créée avec GEM IconEdit.

Objet 11 : (G_IMAGE) - Utilisé pour afficher une bit-image (champ de donnée uniquement) créée avec GEM IconEdit.

Objets des arbres ALERT

Un ALERT utilise une icône d'attention optionnelle, jusqu'à cinq lignes de messages et jusqu'à trois options (boutons) de sortie. La description des icônes (ci-dessous) suggère l'emploi possible de chacune d'elle, mais cette "philosophie" est entièrement optionnelle.

Button **Message Line**

1

2



3



4



5

- Objet 1 : L'état par défaut d'une option de sortie d'un ALERT est défini dans le corps du programme.
- Objet 2 : Une chaîne de texte non formaté qui indique à l'utilisateur la nature du problème. Cinq lignes de 40 Caractères chacune peuvent être affichées au maximum dans un ALERT.
- Objet 3 : Icône NOTE. C'est l'icône par défaut. Ce type de message informe l'utilisateur qu'il a tenté une commande que l'application ne permet pas. Ce message peut n'avoir comme sortie que l'option OK, afin que l'utilisateur puisse indiquer qu'il a pris connaissance du message.
- Objet 4 : Icône ATTENTE. Cette sorte de message peut informer l'utilisateur qu'il a tenté une action autorisée, mais que celle-ci ne peut être effectuée dans les circonstances actuelles, peut-être parce que la porte de l'unité est ouverte ou qu'un disque n'est pas présent dans l'unité. Le message peut suggérer des solutions. Ce type de message comporte deux options de sortie, Cancel et Retry.
- Objet 5 : Icône STOP. Ce type de message attire l'attention de l'utilisateur sur le fait que sa commande, bien qu'autorisée, peut provoquer une perte de données. Le formatage d'un disque en est un exemple. Ce type de message comporte deux options de sortie, Ok et Cancel.

Objets des arbres FREE**Free-string**

1

2

1. Chaîne de texte non formaté que l'on désire afficher.
2. Une bit-image (champ de données) créée avec GEM IconEdit.

Classes d'Objets

Les objets dans une boîte d'éléments peuvent être divisés en cinq classes :

- * Chaînes non formatées
- * Type boîte
- * Texte formaté
- * Bit-images
- * Icônes.

La table ci-après liste les classes d'objets, les types d'objet dans chaque classe et les arbres dans lesquels ils peuvent apparaître.

Table 3-2 - Classes d'Objets

Classe d'objet	Type d'objet	Arbre (*)
<u>Chaîne non formatée</u>	TITLE	M
	ENTRY	M
	Ligne de séparation	M
	BUTTON	D, P, A
	STRING	D, P
	Ligne de message	A
	Chaîne libre	F
<u>Type boîte</u>	Toutes les boîtes y compris les boîtes extérieures de DIALOG ou PANEL	
	Caractère unique dans une boîte	M, D, P
		D, P
<u>Texte formaté</u>	TEXT	D, P
	BOXTEXT	D, P
	texte modifiable (EDIT:___)	D, P
	texte modifiable inclus dans une boîte	D, P
<u>Bit-images</u>	IMAGE	D, P, F
<u>Icônes</u>	ICON	D, P

(*) M = MENU, D = DIALOG, P = PANEL, A = ALERT, F = FREE

Chaînes non formatées et textes formatés

GEM RCS dispose de deux moyens pour entrer du texte dans des arbres d'objets : les chaînes non formatées et les textes formatés. La table ci-dessous liste les caractéristiques et les différences entre ces deux types de texte.

Table 3-3 - Chaînes non formatées et textes formatés

Chaînes non formatées	Des attributs peuvent être définis. Ils peuvent être alignés dans leur parent. Police de caractères et couleur du système uniquement. Mode transparent uniquement (couleur du fond et visualisation du motif au travers). Toujours aligné à gauche.
Texte Formaté	Des attributs peuvent être définis. Ils peuvent être alignés dans leur parent. Couleurs de texte disponibles. Police de petits caractères disponibles. Mode transparent ou de repositionnement (la couleur de fond et le motif ne sont pas visualisés au travers). Le texte peut être aligné à gauche, à droite ou centré. Différentes couleurs de cadre possibles (texte formaté dans les boîtes uniquement). Différentes épaisseurs de cadre possibles (texte formaté dans les boîtes uniquement).

Ajouter des objets dans des arbres

Pour ajouter un objet dans un arbre, tirer son icône depuis la boîte d'éléments vers la fenêtre de visualisation. On peut alors shift-tirer l'objet dans la fenêtre de visualisation pour effectuer des copies supplémentaires.

La technique qui consiste à shift-tirer un objet est très utile, spécialement lorsque l'on désire deux ou plusieurs objets

possédant les mêmes attributs. Par exemple, lorsqu'on désire trois boutons avec les mêmes attributs, tirer un objet BUTTON depuis la boîte d'éléments, définir les attributs communs aux trois, puis le recopier deux fois. Ceci est plus rentable que de tirer l'objet de la boîte d'éléments trois fois et de définir les attributs trois fois.

Cas particuliers

Arbres MENU L'objet TITLE ne peut être placé que dans la barre de titre. Les ENTRY, lignes de séparation et boîtes vides ne peuvent être placés que dans une boîte MENU.

On ne peut pas ajouter ou retirer des objets du menu DESK. La première ligne du menu DESK est la seule qui peut être modifiée.

Arbres ALERT Un arbre ALERT ne peut contenir qu'une seule icône d'attention. Pour remplacer l'icône NOTE par défaut, tirer une autre icône de la boîte d'éléments et la positionner à l'intérieur de l'ALERT. GEM RCS supprime automatiquement l'icône NOTE et positionne la nouvelle icône. Si aucune icône n'est désirée, tirer l'icône existante vers la corbeille.

Au maximum, un ALERT ne peut disposer que de trois options de sortie et de cinq lignes de 40 caractères chacune. GEM RCS ajuste automatiquement la taille de la boîte d'alerte à la longueur du texte du message.

Déplacement, copie et suppression d'objets

Le déplacement et la copie d'objets ont été décrits dans la Table 1-1. Voir aussi "Utilisation de la tablette" dans ce chapitre.

Pour supprimer un objet, respecter l'une des méthodes suivantes :

- * Tirer l'objet vers la corbeille.
- * Sélectionner l'objet et cliquer l'icône de la corbeille.
- * Sélectionner l'objet et choisir la commande Delete du menu EDIT.

Fixer la taille des objets

On peut définir la taille de la plupart des objets en le sélectionnant et en tirant son contrôleur de taille.

Certains objets, parmi lesquels les objets STRING et ceux qu'on

modifie grâce à l'Edit Unformatted String Object, changent automatiquement de taille lorsqu'on augmente le texte de l'objet. Cependant, une telle modification peut déstabiliser la relation de taille parent-enfant (décrite ci-dessous).

Conservation de la relation de taille parent-enfant

Les règles sur la structure des objets imposent que l'objet parent contienne toujours ses enfants (Voir chapitre 6 du Manuel de Référence de GEM AES). Cela signifie que l'objet enfant ne peut pas être plus grand que son parent et qu'il ne peut en aucun cas dépasser les limites de son parent.

Dans plusieurs cas, GEM RCS protège la relation de taille parent-enfants. Par exemple, on ne peut pas réduire la boîte de menu si les entrées ou lignes de séparation dépassent les limites de cette boîte. De la même manière, on ne peut pas augmenter la taille d'un objet BUTTON au-delà des limites de la boîte de dialogue.

Cependant, en modifiant des textes dans l'Edit Unformatted String Object, on peut rendre un texte trop long pour qu'il tienne encore dans son objet parent :

- * une STRING dans un dialog ou un panel.
- * une ENTRY dans une boîte de menu.

Par exemple, on peut saisir un texte dans une ENTRY de telle façon que la commande déborde des limites de la boîte de menu. Dans ce cas, GEM RCS affiche un message indiquant que l'enfant ne tient pas dans son parent. Deux options sont possibles :

OK GEM RCS accepte la longueur de ENTRY et augmente automatiquement la taille de l'étendue de la chaîne pour contenir la chaîne complète. On doit ensuite augmenter la taille de la boîte de menu (sans cela, on sait que GEM AES n'écrit dans le tampon-mémoire MENU/ALERT et ne redessine que la partie située sous la boîte du menu. Tout ce qui se trouve hors de cette boîte restera sur l'écran.)

Cancel GEM RCS ignore les modifications et ne tient compte que du texte original de l'ENTRY.

Icônes et bit-images

Les icônes et les bit-images sont toutes deux créées avec GEM IconEdit.

Une icône est constituée de DATA (l'image) et d'un MASK, habituellement une représentation solide de DATA. La fonction du MASK est d'empêcher les couleurs de fond et les motifs d'être vus au travers du DATA. On peut éditer une icône pour y inclure une

chaîne ou un caractère unique à plusieurs emplacements de l'icône (voir DIALOG Edit Icon Object dans ce chapitre.)

Une bit-image est constituée uniquement de DATA. Puisqu'elle ne comporte pas de MASK, une bit-image permet à une couleur de fond ou à un motif de voir au travers. A l'inverse des icônes, une bit-image ne peut pas être éditée pour contenir une chaîne ou un caractère.

Chargement des icônes

Pour charger une icône dans un arbre DIALOG ou PANEL, tirer l'icône ICON depuis la boîte d'éléments vers la fenêtre de visualisation. Sélectionner ensuite l'icône et choisir la commande Load... du menu OPTIONS.

GEM RCS affiche d'abord un message demandant si l'on souhaite charger pour l'icône, les DATA, le MASK ou les deux ensemble. Lorsque l'on charge uniquement les DATA, les couleurs du fond et les motifs seront vus au travers de l'icône. Lorsque l'on charge MASK uniquement, on fait une ébauche avec la couleur du fond ou le motif, mais l'image de l'icône n'apparaît pas.

Lorsqu'on choisit à la fois DATA et MASK, GEM RCS affiche deux fois le Sélecteur d'Objet. La première fois, sélectionner les DATA de l'icône. Lorsque GEM RCS affiche une seconde fois le Sélecteur d'Objet, sélectionner le MASK de l'icône.

Note : Les fichiers des DATA et des MASK possèdent tous deux la même extension .ICN. Il est nécessaire de se fixer une convention personnelle pour les distinguer, comme ICOND.ICN et ICONM.ICN.

Chargement des bit-images

Pour charger une bit-image dans un arbre DIALOG ou PANEL, tirer l'icône IMAGE depuis la boîte d'éléments vers la fenêtre de visualisation. Sélectionner alors l'icône, puis choisir la commande Load... du menu OPTIONS. GEM RCS affiche le Sélecteur d'Objet, et l'on peut alors choisir le fichier de la bit-image.

Comme pour les icônes, les fichiers des bit-images possèdent l'extension .ICN.

Utilisation des arbres FREE

Les arbres FREE peuvent être utilisés pour des éléments dépendant du contexte lors de l'exécution d'une application (voir chapitre 1). Nous allons utiliser GEM RCS comme exemple d'utilisation des chaînes FREE et des images FREE.

Note : On peut assembler les chaînes FREE et les images FREE dans autant d'arbres FREE que l'on souhaite. Cependant, lorsque GEM RCS

sauvegarde le fichier ressource, il combine toutes les chaînes FREE dans un arbre unique appelé FRSTR1, et toutes les images FREE dans un autre arbre appelé FRIMG1.

Chaînes FREE

Le fichier ressource de GEM RCS, contient plusieurs chaînes dans l'arbre FRSTR1, y compris les commandes suivantes :

- * Hide Parts
- * Show Parts

La forme par défaut de cette commande (Hide Parts) est contenue dans l'arbre des menus de GEM RCS. Lorsque l'utilisateur choisit cette commande, l'application change le pointeur pour la chaîne de commande, de la commande par défaut de l'arbre MENU en la commande alternée de l'arbre FRSTR1. La prochaine fois que le gestionnaire d'écran de GEM AES dessinera le menu, il utilisera Show Parts de FRSTR1. Ainsi, en fonction du contexte, l'application choisira la forme de la commande dans FRSTR1.

Note : Lorsque l'on change la longueur de la chaîne dans l'arbre contenant la commande par défaut, ne pas oublier d'effectuer les mêmes modifications pour les chaînes FREE dépendantes.

Images FREE

La tablette de GEM RCS (décrite ultérieurement dans ce chapitre) est un objet lié au contexte. Son aspect par défaut est une bit-image dans l'arbre des objets de la boîte à outils. L'arbre FRIMG1 contient son autre aspect (indiquant que l'utilisateur a placé quelque chose sur la tablette) et une copie de la version par défaut.

Lorsque l'utilisateur place quelque chose sur la tablette, GEM RCS utilise la bit-image alternée de FRIMG1. La prochaine alternance utilisera la copie de FRIMG1. Ainsi la bit-image utilisée sera l'une des deux versions contenues dans FRIMG1.

Utiliser les dialogues d'édition d'objets

Lorsqu'on ouvre un objet dans la fenêtre de visualisation, GEM RCS affiche un des messages (DIALOG) suivants :

- * Edit Unformatted String Object
- * Edit Box type Object
- * Edit Formatted Text Object
- * Edit Bit Image Object
- * Edit Icon Object

Le message affiché dépend de la classe de l'objet ouvert (voir table 3-2).

Tous les messages d'édition d'objet sont différents (voir ci-dessous les différentes descriptions), mais dans tous les cas on utilise un message pour donner un (nouveau) nom à l'objet sur lequel on travaille. Pour cela, cliquer sur la ligne suivant la mention "Object Name" et saisir le nom que l'on veut affecter à l'objet. On peut alors se référer à ce nom dans le corps du programme.

DIALOG Edit Unformatted String Object

Le texte par défaut, BUTTON ou STRING, apparaît dans le champ "text". Pour modifier ce nom, appuyer sur la touche Esc pour effacer le texte courant, puis saisir le nouveau texte. On peut déplacer le curseur en avant et en arrière dans ce champ, en utilisant les touches curseur-droit et curseur-gauche. On peut effacer des caractères avec les touches Backspace et Delete.

Pour visualiser la longueur de la chaîne, des repères indiquent les longueurs de 10, 20, 30 et 40 caractères.

Nous avons déjà décrit, dans ce chapitre, les différences entre les chaînes non formatées et les textes formatés.

DIALOG Edit Box type Object

Lorsque l'objet est un caractère unique dans une boîte (type G_BOXCHAR), le champ "Character" contient le caractère par défaut. On peut modifier ou effacer ce caractère si on le désire. Pour tous les autres types de boîtes, le champ "Character" est vide, et GEM RCS ignore les caractères qui pourraient y être saisis.

DIALOG Edit Formatted Text Object

Ce DIALOG existe sous deux formes : une pour les textes que l'on peut éditer (G_FTEXT et G_FBOXTEXT) et une pour les textes que l'on ne peut pas éditer (G_TEXT et G_BOXTEXT). La valeur ob_spec des quatre types d'objets est un POINTEUR vers la structure TEDINFO (voir Chapitre 6 du Manuel de Référence de GEM AES).

Nous avons déjà décrit, dans ce chapitre, les différences entre les chaînes non formatées et les textes formatés.

Le DIALOG pour un texte formaté qui ne peut être édité ne comprend qu'un seul champ, PTEXT, dans lequel on saisit le texte.

Le DIALOG pour un texte formaté qui peut être édité comprend trois champs : PTMPLT (gabarit), PVALID (validation) et PTEXT (saisie du texte). Ces champs sont décrits en détail dans le Manuel de Référence de GEM AES (chapitre 6, sous le paragraphe "structure de TEDINFO"). Remarquer que dans GEM RCS, PTMPLT, PVALID et PTEXT utilisent un tildé (~) au lieu du caractère de soulignement (_)

pour éviter les confusions entre le champ lui-même et les emplacements des caractères à éditer.

L'exemple suivant illustre comment PTMPLT, PVALID et PTEXT créent un champ de texte éditable dans lequel l'utilisateur peut saisir une date :

```
PTMPLT>Date du jour:  __/__/__
PVALID>-----99~99~99
PTEXT>-----01~01~86
```

PTMPLT Pour éditer le champ, presser la touche Esc, puis saisir la nouvelle chaîne comme indiqué ci-dessus. Les tildés occupent la place des caractères qui seront saisis par l'utilisateur lors de l'exécution.

PVALID Le champ validation contrôle l'emplacement et le type des caractères qui seront saisis par l'utilisateur. Un "9" indique que l'utilisateur ne peut saisir qu'un chiffre (0 à 9) à cet endroit. Les tildés représentent des caractères littéraux qui doivent apparaître exactement comme dans PTMPLT.

PTEXT Les chiffres situés sous les "9" du champ validation représentent la valeur par défaut pour ce champ. L'utilisateur peut les écraser pour changer la date.

DIALOG Edit Bit Image Object

Ce DIALOG contient uniquement le champ "Object Name" et un pense-bête sur la commande Load... (menu options) pour charger les données d'une nouvelle bit-image. Nous avons déjà décrit, dans ce chapitre, le chargement d'une bit-image.

DIALOG Edit Icon Object

En plus de donner un nom à l'objet, ce DIALOG est aussi utilisé pour :

- * Saisir et positionner une chaîne que l'on veut voir apparaître comme un élément de l'icône.
- * Saisir et positionner un caractère unique que l'on veut voir apparaître comme un élément de l'icône.

Le DIALOG contient deux boîtes d'emplacement : une pour le champ "Text" et une pour le champ "Character". On peut positionner un texte en haut, au milieu ou en bas de l'icône. On peut positionner un caractère unique dans l'une des 9 positions relatives d'une icône.

Nous avons déjà décrit, dans ce chapitre, le chargement d'une icône.

Utilisation de la tablette



1



2

1. La tablette est un endroit de stockage pour les arbres et les objets de la fenêtre de visualisation. On peut déplacer ou copier des items sur la tablette, mais un seul objet peut être présent sur la tablette à un instant donné. Le déplacement ou la copie écrase ce qui se trouvait précédemment sur la tablette.

2. Lorsqu'on place un item sur la tablette, GEM RCS l'indique en y dessinant une feuille de papier cornée.

Pour déplacer un arbre ou un objet vers la tablette, utiliser l'une des méthodes suivantes :

- * Tirer son icône sur la tablette.
- * Sélectionner l'arbre ou l'objet, puis cliquer sur l'icône de la tablette.
- * Sélectionner l'arbre ou l'objet, puis choisir la commande Cut du menu EDIT.

Déplacer un arbre ou un objet de la tablette le fait disparaître de la fenêtre de visualisation.

Pour copier un arbre ou un objet sur la tablette, utiliser l'une des méthodes suivantes :

- * Shift-tirer son icône
- * Sélectionner l'arbre ou l'objet, puis utiliser la commande Copy du menu EDIT.

Copier un arbre ou un objet sur la tablette laisse l'icône originale dans la fenêtre de visualisation.

Pour déplacer un arbre ou un objet de la tablette vers la fenêtre de visualisation, utiliser l'une des méthodes suivantes :

- * Tirer l'icône depuis la tablette.
- * Afficher le menu EDIT, appuyer sur le bouton de la souris lorsque la commande Paste est en inversion vidéo, et tirer depuis le menu. Utiliser cette technique lorsque la boîte à outils a été retiré de l'écran par la commande Hide Tools.

Toute autre technique vide la tablette.

Pour recopier un arbre ou un objet dans la fenêtre de visualisation, shift-tirer l'icône de la tablette. La tablette

contient toujours l'arbre ou l'objet d'origine.

A chaque fois que l'on déplace ou copie un arbre à partir de la tablette, GEM RCS affiche le message "Enter Tree Name" afin de lui donner un nom unique.

On peut aussi utiliser la tablette comme un emplacement réservé pour un objet de la boîte d'éléments (d'un type d'arbre donné), puis ajouter l'objet à un autre arbre. Par exemple, on peut ajouter des bit-images à un arbre MENU en respectant les étapes suivantes :

1. Ouvrir un arbre DIALOG.
2. Mettre une bit-image dans la fenêtre de visualisation de l'arbre DIALOG, puis la déplacer ou la copier sur la tablette.
3. Fermer l'arbre DIALOG et ouvrir l'arbre MENU.
4. Déplacer ou copier la bit-image de la tablette dans le menu. On peut souhaiter que la bit-image soit l'enfant d'une boîte vide. (Voir la description d'une boîte vide ci-dessus, dans le paragraphe "Objets de l'arbre MENU".)

Note : il faut être prudent lorsqu'on effectue cette opération. Certains objets et arbres sont incompatibles entre eux, et leur association peut conduire à des résultats imprévisibles (voir "Types d'objets d'arbre inconnus" dans le chapitre 4.)

GEM RCS efface la tablette lorsqu'on utilise la commande New qui débute une nouveau fichier ressource et lorsqu'on utilise la commande Open... pour modifier un fichier ressource existant. Ces deux commandes appartiennent au menu FILE.

Utilisation de la boîte à outils (Toolkit)

La boîte à outils contient les outils qui permettent d'effectuer de nombreuses modifications dans un objet donné, parmi lesquelles :

- * changer la couleur de l'objet
- * changer le motif de remplissage de l'objet
- * aligner les objets avec leur parent
- * définir certains attributs pour l'objet
- * changer l'épaisseur de la ligne de l'objet

Un outil est désactivé lorsque son utilisation n'a aucun sens pour l'objet choisi. Par exemple, l'outil de motif de remplissage est désactivé pour les objets STRING, car un objet STRING ne peut pas posséder de motif de remplissage.

Lorsqu'on déplace le pointeur sur la boîte à outils, GEM RCS n'affiche en inversion vidéo que les outils compatibles avec l'objet courant. Pour utiliser un outil, cliquer sur l'icône lorsqu'elle est en inversion vidéo. GEM RCS affiche alors un menu des options de cet outil.

Pour choisir une option du menu, tirer dans le menu et cliquer lorsque l'option recherchée apparaît en inversion vidéo. Lorsqu'on clique en dehors du menu, celui-ci disparaît et l'objet est inchangé.

Description des outils

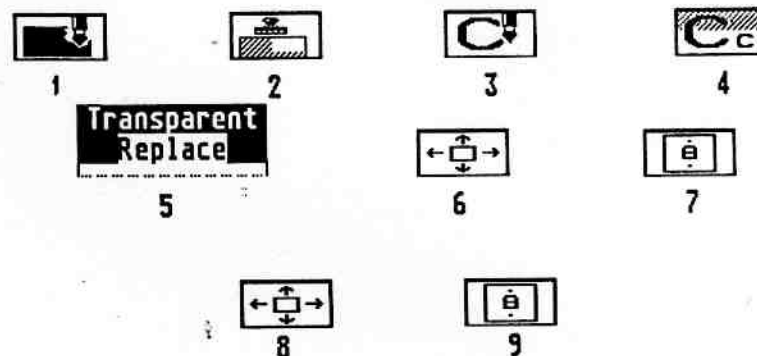


FIGURE 1 : Choisit une couleur de fond pour l'objet. Le menu apparaît en couleur si l'ordinateur dispose d'un moniteur couleur. Dans le cas contraire, définir la couleur en utilisant le numéro correspondant de la couleur (décrit dans le paragraphe "Couleur des objets", chapitre 6 du Manuel de Référence de GEM AES.)

FIGURE 2 : Choisit un motif de remplissage pour un objet. **Note :** Pour qu'un motif de remplissage soit visible, il doit également posséder une couleur. Les couleurs utilisées par défaut sont le blanc avec un motif de remplissage transparent.

FIGURE 3 : Sélectionne une couleur pour les objets TEXT, BOXTEXT, FTEXT ou FBOXTEXT. Le menu apparaît en couleur si le moniteur est en couleur.

FIGURE 4 : Aligne un texte, change la couleur du texte ou du fond pour les objets TEXT, BOXTEXT, FTEXT ou FBOXTEXT uniquement. La plupart des commandes s'expliquent d'elles-mêmes.

FIGURE 5 : Rend chaque case de caractère (le fond du texte) transparente. La couleur et le motif de remplissage de la boîte contenant le texte sont vus par transparence.

FIGURE 6 : Rend toutes les cases de caractères du texte opaques. Le texte apparaît sur un fond blanc.

FIGURE 7 : Aligne ou redéfinit la taille d'un objet en fonction de son parent. Les options d'alignement s'expliquent d'elles-mêmes ; les options Fill et Snap sont décrites ci-dessous :

FILL HORIZONTAL : Redéfinit la taille de l'objet de telle façon qu'il occupe toute la largeur de l'objet parent.

FILL VERTICAL : Redéfinit la taille de l'objet de telle façon qu'il occupe toute la hauteur de l'objet parent.

CHARACTER SNAP : Permet de positionner l'objet d'un arbre PANEL à la limite du caractère le plus proche. Ceci est utile pour obtenir un alignement précis des objets d'un PANEL qui autrement peuvent être placés n'importe où.

FIGURE 8 : Sélectionne les attributs d'un objet. GEM RCS place une marque près de tout attribut utilisé pour l'objet courant. Les attributs de la moitié supérieure du menu définissent les drapeaux des objets (ob_flag) et affectent l'interaction de l'objet avec l'appel FORM_DO. Les attributs de la moitié inférieure du menu concernent les états de l'objet (ob_state) et concernent le tracé de l'objet par l'Object Library. (voir Manuel de Référence de GEM AES, chapitres 6 et 7.)

Les différents attributs sont décrits ci-dessous :

SELECTABLE : L'utilisateur peut choisir l'objet en le cliquant. Le fait de cliquer fait apparaître l'objet en inversion vidéo.

EXIT : Cliquer sur l'objet réalise une condition de sortie et demande à FORM_DO de finir le traitement et de retourner une valeur. Cet attribut est habituellement associé à une option de sortie dans un DIALOG. Les objets qui possèdent l'attribut Exit doivent aussi posséder l'attribut Selectable. Ils peuvent également posséder l'attribut Default, mais celui-ci est optionnel.

DEFAULT : Lorsque cet attribut est utilisé, cet objet est automatiquement sélectionné lorsque l'utilisateur presse la touche Enter. Si plusieurs objets d'un même arbre possèdent l'attribut Default, aucune erreur n'est signalée, mais le résultat est imprévisible.

Note : Les attributs Exit et Default ajoutent un pixel à l'ensemble du cadre d'un objet BUTTON. Ainsi, une option qui ne possède que l'attribut Exit a un cadre de deux pixels d'épaisseur, tandis qu'un objet qui possède également l'attribut Default a un cadre de trois pixels d'épaisseur.

RADIO BUTTON : Définit l'objet comme un élément d'une série de "boutons radio". Les boutons radio sont semblables au sélecteur de gamme d'un autoradio ; appuyer sur un bouton relâche automatiquement celui qui était sélectionné, ce qui signifie qu'un seul bouton peut être actif à un instant donné. Tous les éléments d'une série de boutons radio sont imbriqués au même niveau d'un objet parent commun.

TOUCHEXIT : Appuyer sur le bouton de la souris lorsque le pointeur est sur l'objet réalise une condition de sortie et demande à FORM_DO de finir le traitement et de retourner une valeur. Cette action est instantanée. Les objets qui possèdent l'attribut Touchexit ne doivent pas posséder l'attribut Selectable.

EDITABLE : L'utilisateur peut modifier l'information de l'objet lors de l'exécution. Cet attribut ne peut être utilisé qu'avec des objets textes que l'on peut éditer.

HIDDEN : Cache l'objet afin qu'il ne soit pas visible à l'écran. Utiliser la commande Unhide Children du menu HIERARCHY pour afficher les objets cachés.

SHADOWED : Dessine une zone ombrée autour de l'objet (habituellement une boîte). Il y a conflit entre cet attribut et l'attribut Outline (voir ci-dessous.)

CHECKED : Dessine un triangle dans la marge gauche de l'objet.

OUTLINE : Tracé une ligne extérieure autour d'un objet. Cet attribut est incompatible avec l'attribut Shadowed et n'a aucun effet sur les boîtes possédant des cadres "ouverts".

CROSSED : Trace une croix sur l'objet dans la couleur du fond. Ne peut être utilisé qu'avec les objets contenus dans des boîtes.

DISABLED : Trace l'objet en demi-intensité (gris). L'utilisateur ne peut pas sélectionner un objet qui possède l'attribut Disabled.

SELECTED : Cet objet est pré-sélectionné (il est affiché en inversion vidéo) lorsque l'utilisateur visualise l'arbre.



FIGURE 1 : Sélectionne une couleur pour le cadre de l'objet. Le menu apparaît en couleur si l'ordinateur possède un moniteur couleur.

FIGURE 2 : Sélectionne une épaisseur de cadre pour tout objet situé dans une boîte, à l'exception des BUTTON. L'épaisseur par défaut est de un pixel sur le pourtour externe de l'objet (en bas de la colonne gauche du menu). Cette colonne gauche permet également des épaisseurs de deux ou trois pixels, à l'extérieur. La colonne droite permet le tracé d'un cadre de un, deux ou trois pixels d'épaisseur à l'intérieur de l'objet. L'option située en bas et au centre du menu est une bordure nulle (invisible). Les motifs de remplissage dans ce menu sont uniquement destinés à l'illustration et n'affectent en aucun cas le motif de l'objet. Le schéma correspondant à cet attribut a été donné dans le chapitre 2.

Commandes des menus de GEM RCS

Note : Les commandes sont désactivées (muettes) lorsque leur choix n'a aucun sens dans le contexte présent de GEM RCS. Par exemple, les commandes du menu EDIT (Cut, Copy, Paste et Delete) sont désactivées lorsqu'aucun arbre ou objet n'est sélectionné.

File	
New	^W
Open...	^O
Merge...	^N

Close	^C
Save	^V
Save As...	^M
Abandon	^A

Quit	^Q

New Efface la fenêtre de visualisation pour débiter un nouveau fichier ressource.

Open... Cette commande n'est disponible qu'au niveau de base lorsqu'aucune icône d'arbre n'est sélectionnée. Cette commande affiche le Sélecteur d'Objet qui permet d'ouvrir un fichier ressource existant.

Merge... Affiche le sélecteur d'Objet qui permet d'ouvrir un fichier ressource existant en vue de le chaîner avec le fichier ressource courant.

Le chaînage de fichiers ressources peut entraîner des conflits de noms lorsque le fichier chaîné contient des noms déjà utilisés dans le fichier courant. GEM RCS crée des nouveaux noms pour les doublons. (Imprimer les fichiers .H ou .I est un bon moyen de vérifier l'existence des doublons. Voir la commande Output... du menu GLOBAL.)

Close Ferme l'arbre ou le fichier ressource courant. Fermer un fichier provoque son effacement de l'écran. (Lorsque les modifications effectuées n'ont pas été sauvegardées, GEM RCS affiche un message demandant si on souhaite les sauvegarder ou ne pas en tenir compte. Voir ci-dessous **Save** et **Abandon**.) Fermer un arbre permet de revenir au niveau de base du fichier. Cliquer sur la case de fermeture a le même effet que l'utilisation de cette commande.

Save Sauvegarde le fichier ressource courant. Le fichier reste dans l'espace de travail et on peut continuer à le modifier. Utiliser cette fonction périodiquement afin de ne perdre qu'un minimum de modifications en cas de coupure de courant ou d'un mauvais fonctionnement de l'ordinateur.

Save as... Sauvegarde le fichier ressource sous le nom qu'on lui donne. Utiliser cette commande dans l'une des situations suivantes :

- * La première fois, pour donner un nom et sauvegarder le fichier ressource.
- * Pour sauvegarder la version modifiée d'un fichier existant sous un nouveau nom et/ou répertoire. Le nouveau chemin d'accès et/ou nom de fichier apparaît dans la barre de titre. La version d'origine du fichier reste sur le disque sous l'ancien nom.

Abandon Annule toutes les modifications du fichier courant effectuées depuis la dernière sauvegarde.

Quit Quitte GEM RCS et revient au Bureau GEM.

Note : Lorsqu'on édite un fichier, nouveau ou existant, et que l'on utilise les commandes **New**, **Open...**, **Close**, **Abandon** ou **Quit** sans avoir préalablement sauvegardé les modifications, GEM RCS affiche un message demandant si l'on souhaite annuler ces modifications. Trois options de sortie sont proposées :

- * Annuler les modifications (**Abandon**)
- * Sauvegarder le fichier (**Save**)
- * Annuler la commande (**Cancel**)

Global

Output...	WO
Protection...	WS
Save Preferences	WR

Hide Parts	WP
Hide Tools	WH

Output... Sélectionne le type des fichiers de sortie (en plus des fichiers .RSC et .DFN) que GEM RCS crée lorsqu'on sauvegarde des fichiers ressources. On peut choisir des fichiers pour le langage C (type par défaut), Pascal, BASIC et Fortran-77, aussi bien qu'un fichier source .RSH qui sera utilisé avec RSCREATE (voir chapitre 4).

Protection... Détermine le niveau de protection que GEM RCS applique lors de l'édition des fichiers ressources. Cette commande possède les options suivantes :

* **LOCKED :** Autorise l'édition et la modification de taille, mais interdit les modifications de structure des arbres. Cette option est destinée à des modifications post-production, comme la traduction des textes pour la création de versions internationales du programme. Les valeurs associées aux noms d'arbres et d'objets ne sont pas modifiées. Ainsi, il n'est pas nécessaire de recompiler l'application après avoir effectué ces modifications.

* **NORMAL :** C'est le niveau de protection par défaut. Avant de réarranger un arbre, GEM RCS affiche un message. Cette option permet toutes les opérations, mais prévient lorsqu'une modification des relations parent-enfants ou un effacement de la mémoire de travail est imminent.

* **EXPERT :** Cette option n'émet aucun message et permet de faire toutes les modifications désirées.

Save Preferences

Les définitions et les sélections courantes des DIALOG de protection et de sortie seront considérées comme les valeurs par défaut pour les futures sessions de GEM RCS. Cette commande les sauvegarde dans le fichier RCS.INF du dossier GEMAPPS. Pour rétablir les valeurs par défaut initiales, supprimer le fichier RCS.INF.

Hide Parts

Rend invisible la boîte d'éléments. Cette commande alterne entre Hide Parts et Show Parts.

Hide Tools

Rend invisible la boîte à outils. Cette commande alterne entre Hide Tools et Show Tools.

Edit

Cut	⌘C
Copy	⌘Y
Paste	⌘A
<hr/>	
Delete	⌘D

Cut

Déplace l'arbre ou l'objet sélectionné sur la tablette. L'original n'apparaît plus dans la fenêtre de visualisation.

Copy

Recopie l'arbre ou l'objet sélectionné sur la tablette. L'original reste dans la fenêtre de visualisation.

Paste

Déplace un arbre ou un objet de la tablette sur la fenêtre de visualisation en la tirant à partir du menu EDIT (Utiliser cette commande lorsque la boîte à outils est invisible.)

Delete

Supprime l'arbre, l'objet ou le groupe d'objets.

Options

Info...	⌘I
Name...	⌘N
Type...	⌘T
Load...	⌘L

Info...

Affiche une information sur le fichier courant ou l'arbre ou l'objet sélectionné. Indique aussi comment l'espace de travail est utilisé et la place disponible.

Name...

Affiche le DIALOG utilisé pour donner un (nouveau) nom à un arbre.

Type...

Affiche le DIALOG utilisé pour changer le type d'un arbre.

Load...

Affiche le Sélecteur d'Objet afin de remplacer un objet IMAGE ou ICON par celui créé avec GEM IconEdit.

Hierarchy

Sort Children...	⌘F
Unhide Children	⌘U
<hr/>	
Remove Parent	

Sort Children...

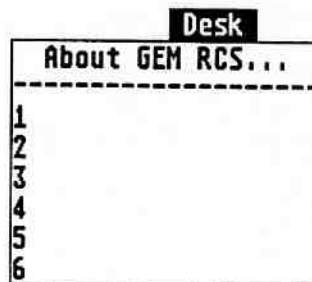
Affiche un DIALOG dans lequel on peut indiquer comment les enfants seront classés dans un parent. Le classement des objets permet de les dessiner dans un ordre logique. Sans classement, les objets sont dessinés dans l'ordre où ils ont été créés. Ceci n'a aucune importance du point de vue fonctionnel, mais peut être visuellement désagréable.

Unhide children

Affiche tous les enfants cachés de l'objet courant sélectionné.

Remove Parent Retire l'objet courant sélectionné sans retirer ses enfants. (Tirer un objet vers la corbeille retire aussi ses enfants.)

Note : Reclassez les enfants après avoir compilé le programme peut, dans certains cas, renuméroter les objets de l'arbre, et rendre les références à ces objets incorrectes dans le programme. (On peut vérifier cela en regardant les fichiers .H créés par GEM RCS.) Lorsque des références incorrectes existent, il est indispensable de recompiler le programme.



About GEM RCS...

Affiche un message qui donne des informations sur GEM RCS, en particulier le numéro de version, le copyright et le nom des auteurs.

Desk accessory Représente le nom des accessoires de bureau disponibles sur le système. Le menu peut comporter jusqu'à six accessoires de bureau.

Table 3-4 - Combinaisons de touches équivalentes des Commandes du Menu

Commande	Touches Equivalentes	Menu
New	Ctrl-W	File
Open...	Ctrl-O	File
Merge...	Ctrl-N	File
Close	Ctrl-C	File
Save	Ctrl-V	File
Save as...	Ctrl-M	File
Abandon	Ctrl-A	File
Quit	Ctrl-Q	File
Output...	Alt-O	Global
Protection...	Alt-S	Global
Save Preferences	Alt-R	Global
Hide/Show Parts	Alt-P	Global
Hide/Show Tools	Alt-H	Global
Cut	Alt-C	Edit
Copy	Alt-Y	Edit
Paste	Alt-A	Edit
Delete	Alt-D	Edit
Info...	Alt-I	Options
Name...	Alt-N	Options
Type...	Alt-T	Options
Load...	Alt-L	Options
Sort Children...	Alt-F	Hierarchy
Unhide Children	Alt-U	Hierarchy

GEM Resource Construction Set

Combinaisons de touches équivalentes des Commandes du Menu

La plupart des commandes de GEM RCS peuvent être exécutées en cliquant sur la commande du menu ou en tapant une combinaison de touches. La table ci-dessous liste les commandes dans l'ordre où elles apparaissent dans les menus et donne les combinaisons de touches équivalentes.

Note : Une combinaison de touche équivalente n'est active que lorsque la commande elle-même est active.

CHAPITRE 4

RSCREATE et autres informations techniquesUtiliser RSCREATE

RSCREATE est un programme utilitaire en langage C qui crée un fichier ressource. Il est principalement destiné à deux utilisations :

- * Pour créer un fichier ressource à partir d'un fichier édité "manuellement".
- * Pour porter des fichiers ressources dans différents environnements de microprocesseurs.

RSCREATE utilise le fichier .RSH (décrit ci-dessous) comme fichier source. Pour générer un nouveau fichier .RSC avec RSCREATE, utiliser le fichier .RSH dans RSCREATE.C, compiler, chaîner et exécuter RSCREATE.

Le fichier .RSH

Le fichier .RSH, qui est un fichier ASCII pouvant être édité avec un éditeur de texte ou un programme de traitement de texte, est un des fichiers de sortie optionnels de GEM RCS. Pour que GEM RCS crée un fichier .RSH, choisir la commande Output... du menu GLOBAL, cliquer sur la mention "*.RSH". Le fichier .RSH peut être l'option par défaut des fichiers de sortie en utilisant la commande Save Preferences du menu GLOBAL (voir chapitre 3).

GEM RCS ne produit des fichiers de sortie que comme un élément du traitement de sauvegarde d'un fichier ressource. Pour cette raison, l'ouverture d'un fichier ressource dans le but de produire un fichier .RSH (en d'autres mots, sans éditer le fichier) ne peut se faire qu'en utilisant la commande Save as... et en lui donnant le nom du fichier courant dans le Sélecteur d'Objet. Puisque le fichier ressource n'a pas été édité, fermer le fichier avec la commande Close ou la case de fermeture ne sauvegarde pas le fichier et ne crée pas le fichier .RSH.

Fichiers ressources édités "manuellement"

Pour éditer "manuellement" un fichier ressource, générer un fichier .RSH et faire les modifications du fichier .RSH à l'aide d'un programme de traitement de texte ou d'un éditeur de texte. Cependant, avant de faire cela, il est indispensable d'avoir assimilé le contenu des paragraphes "Object Library" et "Form library" du Manuel de référence de GEM AES.

Pour éditer le fichier .RSH, garder en mémoire les points suivants :

- * Lorsqu'on insère un nouvel objet, TEDINFO, etc, l'insérer à la fin des entrées courantes.
- * Lorsqu'on insère de nouveaux objets dans des arbres existants, mettre à jour les définitions de base de l'arbre.
- * GEM RCS considère les arbres dans l'ordre suivant : base d'abord, suivi par ses enfants, de gauche à droite, cette règle s'appliquant de façon récursive. Bien s'assurer que l'on entre les nouveaux arbres dans ce même ordre.
- * Lorsqu'on crée un fichier ressource avec GEM RCS et que l'on s'attend à l'éditer "manuellement" plus tard, ne pas donner de nom d'objets dans GEM RCS. Attendre que le fichier .RSH ait été édité et exécuté par RSCREATE. Puis lire le nouveau fichier .RSC dans GEM RCS et attribuer les noms d'objets.

Lorsqu'on saisit des noms d'objets lors de la première passe dans GEM RCS, les fichiers .H et .DFN sont liés aux numéros des objets définis dans cette première passe. RSCREATE change ces numéros d'objet, et GEM RCS s'avère alors incapable d'utiliser les fichiers .DFN pour tout nouveau travail sur le fichier ressource.

- * GEM RCS définit automatiquement le drapeau LASTOB dans la structure OBJECT du dernier objet de chaque arbre (voir Manuel de référence de GEM AES, chapitre 6). Lors de l'édition "manuelle" d'un arbre, bien s'assurer que ce drapeau est défini dans le dernier objet de l'arbre. Dans le cas contraire, l'application peut se planter lors de l'exécution.

Porter un fichier ressource

Pour porter un fichier ressource d'un environnement à un autre, respecter les étapes suivantes :

1. Déplacer le fichier .RSH dans le nouvel environnement.
2. Ajouter un en-tête qui rend le fichier .RSH compatible avec l'environnement de destination. Lorsque l'on porte vers un environnement 68K, l'en-tête est commentée dans RSCREATE.
3. Saisir le fichier .RSH au format destination comme fichier d'entrée dans RSCREATE.
4. Compiler, chaîner et exécuter RSCREATE dans l'environnement de destination.

Notes sur le compilateur

Compiler RSCREATE avec Lattice C ou une autre implémentation complète du langage.

Certains compilateurs C (parmi lesquels Lattice) regroupent les doublons lorsqu'ils compilent RSCREATE. Cela peut présenter un problème lors de l'exécution si les chaînes doivent être éditées. Dans ce cas, ré-exécuter le fichier ressource à travers GEM RCS pour supprimer les doublons.

Chainer des ressources

GEM AES impose une limite de 64K pour les fichiers ressources. Lorsqu'un fichier plus important est nécessaire, on peut chaîner deux fichiers ressources ensemble. Cependant, pour utiliser le second fichier, il faut effacer de AES toutes les références au premier fichier, et effacer le premier fichier de la mémoire en faisant un appel à RSRC_FREE avant de charger le second fichier.

Types d'objets d'arbre inconnus

Lorsqu'on charge un fichier ressource et que l'on ne dispose pas de son fichier .DFN, GEM RCS affiche chaque arbre comme UNKNOWN (inconnu, représenté par un point d'interrogation) ou comme ALERT. Avant de pouvoir travailler sur ce fichier, il faut convertir les arbres inconnus dans l'un des types autorisés. Il faut également donner un nom à chaque arbre pour le fichier .DFN qui sera créé au moment de la sauvegarde du fichier.

Pour attribuer un type et un nom aux arbres, respecter les étapes suivantes :

1. Cliquer sur le premier UNKNOWN (TREE1) pour le sélectionner.
2. Afficher le menu OPTIONS, cliquer la commande Type... et choisir l'option DIALOG. Ceci permet de remplacer l'icône du point d'interrogation par l'icône DIALOG.
3. Faire un double-clic sur l'icône TREE1 pour ouvrir l'arbre et voir alors ce qu'il contient.

Si l'arbre est vraiment du type DIALOG :

1. Fermer l'arbre pour revenir au niveau de base.
2. Sélectionner la commande Name... du menu Options.
3. Donner un nom à l'arbre.

Si l'arbre n'est pas du type DIALOG :

1. Fermer l'arbre pour revenir au niveau de base.
2. Sélectionner la commande Type... du menu OPTIONS
3. Choisir le type correspondant à l'arbre.
4. Sélectionner la commande Name... du menu OPTIONS
5. Donner un nom à l'arbre.

Pour les arbres de type ALERT, il suffit de les ouvrir pour en étudier le contenu, puis d'utiliser la commande Name... pour leur donner un nom.

En utilisant la commande Type... du menu OPTIONS, on peut changer le type d'un arbre ou d'un objet. Attention, MENU et ALERT n'acceptent que quelques objets spécifiques qui pour la plupart sont incompatibles avec les autres arbres. On peut changer le type d'un arbre comportant de nombreuses restrictions à un type en comportant moins (par exemple, changer ALERT en DIALOG), mais il est déconseillé de faire le changement inverse, celui-ci pouvant conduire à des résultats imprévisibles.

Arranger et aligner des objets

Dans un dialog, particulièrement, on souhaite disposer plusieurs objets d'une façon particulière. Par exemple, on désire que trois ou quatre chaînes de caractères soient alignées sur la droite. Pour d'une boîte, comme ceci :

1. Tirer une boîte depuis la boîte d'éléments vers la fenêtre de visualisation. La faire aussi grande qu'on le souhaite.
2. Ajouter les objets STRING dans la boîte, puis les éditer.
3. Choisir toutes les chaînes de la boîte avec Shift-cliquer.
4. Afficher le menu d'alignement et choisir celui que l'on désire. (Tant que les chaînes sont dans leur boîte parent, il est possible de les déplacer en groupe.)
5. Choisir la boîte parent, puis la commande Remove Parent du menu HIERARCHY. Le fait de retirer la boîte parent permet de réduire à la fois la taille de l'arbre et celle du fichier ressource.

Créer un fichier bibliothèque

Lorsqu'on utilise fréquemment certains arbres, bit-images ou icônes, on peut les regrouper dans un fichier ressource "bibliothèque". Grâce à la commande Merge... du menu FILE, on peut chaîner ce fichier au fichier ressource courant. On peut ensuite utiliser la corbeille ou la tablette pour se débarrasser des objets inutiles.

INDEX ALPHABETIQUE

abort	22
abs	23
access	24
accessoire	18
accstart	16-17, 80-81
aes	81, 224, 235
aesbind	81
alerte	237-238
appstart	16-17, 20, 80-81
ar68	136-141, 197-198
arbre	212, 218, 221, 226
as68	95-105, 183-189
atan	75
atof	25
atoi	25
atol	25
bat	93, 141, 151
batch	93, 151-153
bibliothèque	20, 263
bios	82
bit_image	242-244, 246
brk	26
bss (zone)	12, 14, 87, 98, 155
bureau	15
c068	91, 169-180
c168	92, 181-182
calloc	27
ceil	28
chmod	29
chown	29
classe	85, 87
clearerr	39
close	30
command	93, 141-151, 205-207
compilateur	20, 84, 85, 86, 121
compilation	92-93
constante	85, 86-87
cos	31
cp68	21, 90-91, 165-168
cpm68K	108, 209
creat	32
creata	32
creatb	32
ctype	33
data (zone)	12-14, 16-17, 87, 98, 155

directive	98
dump	153-154, 199
en-tête	12, 15, 83
etoa	34
exit	35
_exit	35
exp	36
fabs	37
fclose	38
fdopen	42-43
feof	39
ferror	39
fflush	38
fgetc	46
fgets	48
file	30, 39, 42
fileno	39
floor	40
fmod	41
fopen	42-43
fprintf	57-58
fputc	59
fputs	60
fputw	59
fread	44
free	27
freopen	42-43
fscanf	64-65
fseek	45
fsel_input	18
ftell	45
ftoa	34
fwrite	44
gemdos	12, 21, 82, 108
gemlib	20, 33, 80, 81
gemstart	16, 17, 20, 80, 81
getc	46
getchar	46
getl	46
getpass	47
gets	48
getw	46
icône	242-243, 246
index	49
isatty	50
jeu d'instruction	103
libf	20
librairie	20, 138, 139

link68	108-112, 190-193
lo68	112-115, 194-196
log	51
longjmp	66
lseek	52
main	17
Malloc	17, 27
malloc	17, 27
mktemp	53
mmxref	161-162
module	83, 87, 139, 140
nm68	159-161
objet	212, 218, 225, 241
open	54
opena	54
openb	54
osbind	82
page de base	13-14, 16
perror	55
Pexec	15
pointeur	86-87
portab.h	20, 21
portabilité	21, 83, 84, 261
pow	56
printf	57-58
putc	59
putchar	59
putl	59
puts	60
qsort	61
rand	62
Random	62
rce	211-263
read	63
realloc	27
redirection	143
relmod	115
relocation	13
réservation	18
résolution	233
ressource	212-263
rewind	45
rindex	49
rm	162
rscreate	260-263
rsh	260
sbrk	26

scanf	64-65
screen manager	15
send	156-159, 201
setjmp	66
sid (sid68)	117-134, 202-204
signal	67
sin	31
sinh	68
size68	154-156, 200
sprintf	57-58
sqrt	69
srand	62
s_record	156, 157-159
sscanf	64-65
stderr	43
stdin	43, 48, 64
stdio.h	20, 23, 42, 46, 59
stdout	43, 57, 59, 60
strcat	70
strcmp	71
strcpy	72
strlen	73
strncat	70
strncmp	71
strncpy	72
swab	74
symbole	12, 13, 120, 127, 154, 155, 159-161
tan	75
tanh	68
tedinfo	245-246
tell	52
text (zone)	14-14, 16-17, 87, 101, 155
tpa (zone)	12-14, 17, 96
trap 1	82
trap 13	82
trap 14	82
ttyname	76
type	85, 87
ungetc	77
unlink	78
v_opnvwk	80
vdi	80
vdibind	81, 81
wait	162
write	79
xbios	82