MEMSOFT
pour APPLE IIGS

- Ce resume de la documantation est gracieusement offert par ZAHG -

- Thanks to GT -
- Thanks to L.S.D. qui a deja beaucoup fait -
- Thanks to Apple pour le II GS -

CHAPITRE 1 : MEMBASIC

1.1 INTRODUCTION

Ce chapitre expose les caracteristiques generales de l'interpreteur MEMBASIC. Les commandes, fonctions et instructions sont presentees individuellement par ordre alphabetique.

1.2 NOTATIONS UTILISEES

Pour la description de la syntaxe, les regles suivantes sont utilisees :

- Les mots cles du langage sont ecrits en majuscule.
- Tout les signes de ponctuation contenus dans les syntaxes : virgule point-virgule, guillemets, deux-points, parentheses sont obligatoires a l'exception des caracteres "superieur a" (›) , "inferieur a" (‹), "crochet ouvrant" ([) et "crochet fermant" (]).
- Les instructions en minuscules encadrees par ‹ et › doivent etre remplacees par les termes idoines choisis par l'utilisateur.
- Les termes places entre crochets [et] sont facultatifs.
- La notation suivante signifie que l' utilisateur doit choisir entre plusieurs options.

  GOTO ‹numero de ligne›
  ‹etiquette›

- Les termes entre crochets suivis de points de suspensions peuvent etre repetes plusieurs fois dans la limite de la longueur de la ligne.

1.3 COMMANDES, FONCTIONS et INSTRUCTIONS

ABS : Valeur absolue du parametre.
Syntaxe : ABS (‹expression numerique›)

ACOS : Calcul de l'angle dont le cosinus est passe comme parametre. L'angle est exprime en radian ou en degres suivant l'option definie par OPTION ANGLE.
Syntaxe : ACOS (‹expression numerique›)

ANGLE : Calcul de l'angle entre l'axe des abscisses positives et le vecteur joignant l'origine au point de coordonnees indiquees.
Syntaxe : ANGLE (‹expression numerique›,‹expression numerique›)

ASIN : Calcul de l'angle dont le sinus est passe comme parametre. L'angle est exprime en radian ou en degres suivant l'option definie par OPTION ANGLE.
Syntaxe : ASIN (‹expression numerique›)

ASK : Permet de connaitre les parametres d'edition en cours, utilises pour les impressions non formattees.
Syntaxe : ASK ‹element ASK› ‹variable numerique›

ATN : Calcul de l'angle dont la tangente est passee comme parametre. L'angle est exprime en radian ou en degres suivant l'option definie par OPTION ANGLE.
Syntaxe : ATN (‹expression numerique›)

AUTO : Cette commande demande a l'interpreteur de numeroter et indenter automatiquement les lignes au fur et a mesure de la saisie.
Syntaxe : AUTO ON, AUTO OFF

BREAK : Premet de placer des points d'arret dans le programmme.
Syntaxe : BREAK

CASE : Cette instruction est utilisee pour construire des sequences structurees de programme.
Exemple :
```
200 SELECT CASE a$
210 CASE "A" TO "Z"
220       PRINT "c'est une majuscule"
230 CASE "a" TO "z"
240       PRINT "c'est une minuscule"
250 CASE IS ="{", IS ="}", IS = "@"
260       PRINT "c'est un caractere accentue"
270 CASE ELSE
280       PRINT "c'est pas une lettre"
290 END CASE
```

CAUSE EXCEPTION : Simulation d'erreur.
Syntaxe : CAUSE EXCEPTION ‹numero de l'erreur›
Exemple : CAUSE EXCEPTION 3001 simule une division par zero.

CEIL : Arrondi superieur a la nieme decimale du parametre. La forme CEIL a un seul parametre donne le plus petit entier superieur ou egal a la valeur numerique du parametre. La forme CEIL a deux parametre : CEIL (x,n) arrondi la valeur x a la nieme decimale.
Syntaxe : CEIL (‹expression numerique›)
CEIL (‹epression numerique›,‹expression numerique›)

CHAIN : Lance l'execution d'un programme prealablement charge depuis la memoire auxiliaire en lui passant eventuellement des parametres. CHAIN provoque la fermeture des fichiers ouverts, l'effacement de la table des variables apres extraction des eventuels parametres, le chargement puis l'execution du programme indique, l'affectation eventuelle a de nouvelles variables des valeurs passees en parametre.
Syntaxe : CHAIN ‹specification de fichier› [WITH (‹parame› [, ‹param...]]

CHDIR : Change le reperetoire courant.
Syntaxe : CHDIR ‹expresion chainne›

Exemple : CHDIR "A:/MEMSOFT/MEMTOOL"

CHDIR$: Donne de repertoire courant.
Exemple : PRINT CHDIR$
C:/MEMSOFT/TEST/...

CHR$: Cette fonction donne une chaine de UN caractere. Ce caractere correspond au code du jeu de caracteres utilise.
Syntaxe : CHR$ (‹expression numerique›)

CLEAR KEYBOARD : Vide les caracteres d'avance dans la memoire tampon du clavier.

CLOSE : Fermeture d'un fichier ProDos ou d'un fichier de communication. Deux formes pour l'instruction CLOSE existent : CLOSE ou CLOSE ‹numero de fichier› Numero de fichier est une expression numerique indiquant le numero attribue au fichier lors de son ouverture.
Syntaxe : CLOSE
CLOSE ‹numero de fichier›

CLS : Effacement du contenu de la fenetre de travail.

COMSTAT : Donne l'etat actuel du port serie associe a un fichier de communication. La signification des bits de la reponse est :
bit 1 = 1 si un caractere a ete recu.
bit 0 = 1 si le port RS232 est pret a emettre.
Exemple : 1000 OPEN "COM2",1
1010 PRINT "L'etat de la ligne est : "; COMSTAT(1)

COPY : Copie d'un fichier d'un repertoire vers un autre repertoire ou sous un autre nom. La copie peut, si on le desire, etre invisible dans le catalogue.
Syntaxe : COPY ‹expression chaine› TO ‹expression chaine›.
La deuxieme expression commencera par "!" pour l'invisibilite dans le catalogue. Invisibilite seulement sous MEMSOFT; commencera par "@" si l'on desire que le fichier d'arrive ecrase le fichier deja existant de meme nom.
Exemple : COPY "original.msk" TO "nouveau.hlp"
COPY "orig*.*" TO "nouv*.*"
COPY "a:*.*" TO "monrep"

COS : Calcul du cosinus de l'angle passe en parametre.
Syntaxe : COS (‹expression numerique›)

COT : Calcul de la cotangente de l'angle passe en parametre.
Syntaxe : COT (‹expression numerique›)

CSC : Calcul de la cosecante de l'angle passe en parametre.
Syntaxe : CSC (‹expression numerique›)

CURDRIVE : Fixe le disque courant.
Exemple : CURDRIVE "A:" le disque courant devient A.

CURDRIVE$: La fonction CURDRIVE$ donne une chaine de deux caracteres indiquant le disque en cours.

CURLINE : Cette fonction sans parametre est un compteur de lignes d'edition pour les instructions MEMSCREEN : LET "?..." et LET "HARDCOPY..."

DATA : Cette instruction permet d'incorporer dans un programme des donnees qui seront lues au moyen de l'instruction READ.

DATE : DATE donne la date du jour exprimee sous la forme numerique AAJJJ,JJ representant le nombre de jour depuis le debut de l'annee.

DATE$: donne la date du jour exprimee sous la forme chaine "AAAAMMJJ".

DEBUG : Rend actif ou inactif le mode DEBUG. Avec DEBUG ON, on rend le mode DEBUG actif et permet l'utilisation des instruction BREAK et TRACE.

DECLARE STRING : Cette instruction permet de declarer la longueur maximum que peuvent prendre une ou plusieurs variables chaines simples.
Exemple : 110 DECLARE STRING *10 a$,b$,c$, *15
a$ et b$ ont une longueur maximum de 10 et c$ de 15
100 INPUT PROMPT "Donner la longueur maxi : ":n
110 DECLARE STRING a$*(n)

DEG : Cette fonction convertit un nombre exprime en radians en degres.
Syntaxe : DEG (‹expression numerique›)

DELETE : destruction de lignes d'un programme.
Syntaxe : DELETE ‹numero›,‹numero›,...
DELETE ‹numero› TO ‹numero›
DELETE TO ‹numero›
DELETE ‹numero› TO

DIM : Declaration des tableaux et de la plage de variation des indices.
Exemple : 1000 DIM TABLEAU (10,10), CA (1980 TO 1985)

DIR : Permet d'obtenir le contenu d'un repertoire.
Syntaxe : DIR [‹expression chaine›]
Example : DIR "B:*.MEM" Affiche tout les fichiers du repertoire en cours de l'unite B dont le suffixe est MEM.
DIR "AB*.M*" Affiche tout les fichiers du repertoire en cours dont le nom commence par AB et dont le suffixe commence par M.

DISKSIZE : Rend la place totale disponible, en octets, sur le disque specifie
Syntaxe : DISKSIZE (‹expression chaine›)

DO...LOOP : Ces intruction sont destinees a construire des boucles de traitement.
Syntaxe : Debut de la boucle Fin de la boucle

DO	LOOP
DO UNTIL ‹condition›	LOOP UNTIL
DO WHILE ‹condition›	LOOP WHILE

ELSE : Utilise dans les instructions IF...THEN...ELSE, ON GOTO, ON GOSUB, et CASE. Se reporter aux instructions IF, ON GOTO, ON GOSUB, CASE.

END : Indique la fin du programme. Cette instruction provoque la fermeture des fichiers qui etaient encore ouverts.

END IF : Indique la fin du dernier bloc d'une instruction IF.
Exemple : IF ‹condition› THEN
...
...
...
ELSEIF ‹condition› THEN
...
...
...

```
        ELSE
             ...
             ...
        END IF
```

END SELECT : Marque la fin du dernier bloc CASE associe a une instruction SELECT.

EPS : Evaluer l'erreur maximale sur un nombre.
Syntaxe : EPS (<expression numerique>)

ERASE ALL : Efface toute les variables d'un programme, les boucles, les sous programmes et ferme tous les masques et fichiers en cours.

EVALUATE : Cette fonction convertit en une valeur numerique l'expression arithmetique simple contenue dans une expression chaine. L'operateur pourra etre "+", "-", "*" ou "/".
Syntaxe : EVALUATE (<chaine>)
Exemple :
```
1000 a$="123.34"
1010 b$="10.12"
1020 c$=a$ & "+" & b$
1030 EVALUATE c$
```

EXECUTE : Permet d'executer une chaine de caracteres contenant une instruction.

EXIT : Permet de sortir d'une boucle et de passer a l'instruction qui suit immediatement la fin de la boucle.
Exemple :
```
1000 FOR i=1 TO N
....
....
1050   IF ABS(x)<0.000001 THEN EXIT FOR  --+
....                                       !
....                                       !
1100 NEXT i                                !
1110 'suite'  <----------------------------+
```

EXKEY : rend le code de la touche clavier, enfonce par l'utilisateur pour sortir d'un masque MEMSCREEN durant la saisie. Si l'utilisateur abandonne et enfonce la touche "Esc" EXKEY rend 1, s'il enfonce la touche "RETURN", EXKEY rend 0, s'il enfonce les touches F1 a F9, EXKEY rend 1 a 9.

EXLINE : Donne le numero de la ligne ou s'est produit une erreur.

EXP : Calcul de la fonction exponantielle.
Syntaxe : EXP (<expression numerique>)

EXTEXT$: Donne le contenu du message d'erreur.
Exemple :
```
500 PRINT EXTEXT$(EXTYPE)
```

EXTYPE : Donne le numero de l'erreur.

EXWAY : Cette fonction permet, en cas de sortie de la saisie d'un masque par une touche de deplacement, de connaitre cette touche.
EXWAY rend les valeurs suivante :

0	pour RETURN
1	pour Fleche Bas
-1	pour Fleche Haut
2	pour Pomme-Fleche-Haut ou Clic en bas a droite de la zone de saisie.
-2	Pour Pomme-Fleche-Bas ou Clic en haut a gauche de la zone de saisie.

EXZONE : A l'issue d'une saisie de donnees par masque MEMSCREEN, EXZONE rend le numero de la zone de saisie ou la sortie s'est effectuee.

FILESIZE : Rend le cumul des tailles des fichiers correspondants au critere. Le critere de selection est une expression chaine pouvant contenir les caracteres speciaux "?" et "*".
Syntaxe : FILESIZE (‹chaine›)

FIND : Cette commande permet de retrouver une suite de caracteres dans un programme.
Syntaxe : FIND ‹type› [‹region›] "‹caracteres›"
‹type› peut etre "&","‹", "›" ou "vide"
‹region› represente la partie du programme dans laquelle la ont effectue la recherche. La syntaxe est : ‹lmini› TO ‹lmaxi›.
lmini et lmaxi sont des numeros de ligne ou des etiquettes, ou bien "+" pour la ligne maxi,"-" pour la ligne mini, "*" pour la derniere ligne validee par RETURN.
‹caracteres› est une suite de caracteres a rechercher a l'exception du caractere " (guillemet). Le caractere ? peut remplacer n'importe quel caractere y compris le guillemet.

FKEY : Cette instruction permet de modifier la signification des touches de fonction F1 a F9.
Syntaxe : FKEY ‹expression chaine›

FOR : Construction d'une boucle dont l'execution est controle par la valeur d'une variable.
Syntaxe : FOR ‹variable›=‹epression› TO ‹expression› [STEP ‹expression›]

FP : Donne la partie decimale d'une valeur numerique.
Syntaxe : FP (‹expression numerique›)

FREE : Donne des informations sur la place disponible en memoire centrale ou sur disque.
Syntaxe : FREE
FREE (‹expression›)

GET KEYBOARD : Attente d'une touche au clavier.

GOSUB : Branchement vers un sous programme interne.
Syntaxe : GOSUB ‹numero de ligne›
GOSUB ‹etiquette›

GOTO : Branchement a un endroit du programme.
Syntaxe : GOTO ‹numero de ligne›
GOTO ‹etiquette›

HELP : Cette instruction permet de definir le fichier d'aide qui sera pris en compte au prochain enfoncement de la touche F0.
Syntaxe : HELP (‹chaine›)
Exemple : HELP "/REP/SOUSREP/AIDE.HLP"

HELP$: Cette fonction sans parametre rend le nom du fichier d'aide explicite en cours.
Exemple : PRINT HELP$
/REP/TOTO.HLP

HTAB : (fonction) Renvoi la position du curseur, compte a partir du bord gauche de la fenetre.

HTAB : (instruction) Permet de positionner le curseur sur une colonne donnee.
Syntaxe : HATB ‹index›

IF : Cette instruction permet d'effectuer soit des branchements conditionnels soit d'executer des instructions selon qu'une condition est satisfaite ou non
Syntaxe : IF ‹relation› THEN ‹clause› ELSE ‹clause›
‹clause› peut prendre l'une des formes suivantes :
- ‹instruction› [:‹instruction›]...
- ‹numero de ligne›
- ‹etiquette›

```
IF ‹relation›
     THEN
          ‹bloc d'instructions›
     ELSE
          ‹bloc d'instructions›
END IF
```

IMAGE : L'instruction IMAGE permet de definir un format que pourra utiliser une instruction PRINT USING. La ligne MEMBASIC qui contient l'instruction IMAGE ne doit contenir aucune autre instruction. Elle peut par contre commancer par une etiquette.
Syntaxe : IMAGE : ‹format›

INPUT : Permet de lire des donnees introduites au clavier.
Syntaxe : INPUT ‹liste de variables›
INPUT PROMT ‹chaine› : ‹liste de variables›

INPUT# : Permet de lire des donnees depuis un fichier ProDos ou un port de communication serie.
Syntaxe : INPUT#‹numero de fichier›, ‹liste de variables›

INPUT$: Permet delire un nombre exact d'octets depuis un fichier Prodos ou port de communication serie.
Syntaxe : INPUT$ (‹nombre d'octets›,‹numero de fichier›)
nombre d'octet : de 1 a 255

INT : Calcul de la plus garnde valeur entiere inferieure ou egale a celle de l'expression numerique donne comme parametre.
Syntaxe : INT (‹expression numerique›)

IP : Donne la partie entire du parametre. IP(x) est equivalent a :
SGN(x) * INT (ABS(x)).
Syntaxe : IP(‹expression numerique›)

KILL : Permet de detruire un fichier sur disque.
Syntaxe : KILL ‹specificateur de fichier›

LBOUND : La fonction LBOUND donne la valeur minimale que peut prendre l'indice d'un tableau.
Syntaxe : LBOUND (‹nom du tableau› [,‹index›])

LCASE$: LCASE$ restitue une chaine ne comportant que des minuscules.
Syntaxe : LCASE$ (‹chaine›)

LEFT$: Extractiond'une sous-chaine a partir des premiers caracteres de la chaine donnee comme parametre.
Syntaxe : LEFT$ (‹chaine›,‹expression numerique›)

LEN : Donne la longueur de l'expression chaine donnee comme parametre.
Syntaxe : LEN (‹chaine›)

LET : L'instruction LET permet d'affecter une valeur a une variable simple.
LET est egalement utilise pour toute les instructions MEMSCREEN et MEMFILE.
Syntaxe : LET ‹variable› = ‹expression›
LET "‹chaine de caracteres"

LINE INPUT# : Permet de lire des donnees pouvant inclure des "," depuis un fichier ProDos ou un port serie.
Syntaxe : LINE INPUT #‹numero de fichier›,‹variable chaine›

LIST : Permet d'obtenir la liste partielle ou totale d'un programme.
Syntaxe : LIST [‹page›] [‹region› [,‹region›]...] ["@‹fichier›]
‹page› peut etre "‹" ou "›"
‹region› peut une ligne ou un ensemble ‹ligne› TO ‹ligne›
le parametre ligne peut etre : un numero de ligne
"+" : ligne maximum
"-" : ligne minimum
"*" : derniere ligne
‹fichier› represente une specification de fichier. Cette option permet de sauver le listing dans un fichier sur disque.

LOAD : La commande LOAD permet de charger un programme stocke sur memoire auxiliaire.
Syntaxe : LOAD "‹specification de fichier›"

LOCNUMBER : La commande LOCNUMBER permet de renumeroter un programme avec les adresses de debut d'instruction comme numero de de ligne.
Syntaxe : LOCNUMBER

LOG : Calcul du logarithme neperien d'un nombre.
Syntaxe : LOG (‹expression numerique›)

LOG10 : Calcul du logarithme decimal d'un nombre.
Syntaxe : LOG10 (‹expression numerique›)

LOG2 : Calcul du logarithme en base 2 d'un nombre.
Syntaxe : LOG2 (‹expression numerique›)

LOOP : Instruction qui termine une boucle DO.

LTRIM$: La fonction LTRIM$ supprime les espaces de debut de chaine.
Syntaxe : LTRIM$ (‹chaine›)

MAT : Permet d'initialiser un tableau ou de copier un tableau dans un autre.
Syntaxe : MAT ‹nom de tableau› = ‹expression›
MAt ‹nom de tableau› = ‹nom de tableau›

MAX : Donne le plus grand des deux parametres.
Syntaxe : MAX (‹expression numerique›,‹expression numerique›)

MAXLEN : Donne la longueur maximum alloue a une variable chaine.
Syntaxe : MAXLEN (‹chaine›)

MAXLINE : Cette fonction rend le nombre de ligne par page de l'imprimante fixe par SET MAXLINE. (66 par defaut).

MAXNUM : Donne la valeur du plus grand nombre representable en MEMBASIC qui est : 9.9999999999999 E+63

MEMBACKUP : Sauvegarde de fichiers d'un repertoire d'un disque dur vers une ou plusieurs disquettes.
Syntaxe : MEMBACKUP ‹chaine› TO ‹chaine›

La premiere chaine represente le repertoire a sauvegarder suivi d'un eventuel critere de selection. La deuxieme chaine indique le lecteur de disquette concerne par la sauvegarde, sous la forme ‹lettre du lecteur› suivi de ":".
Exemple : MEMBACKUP "*.MSK" TO "A:"

MEMCOMPARE : Verification des disquettes de sauvegarde d'un repertoire de disque dur.
Syntaxe : MEMCOMPARE ‹chaine_1› TO ‹chaine_2›.
chaine_1 : Lecteur de disquettes ou seront placees successivement les disquettes a verifier sous la forme "X:".
chaine_2 : Definit le disque dur et le repertoire sur lequel s'effectuera la comparaison.
Exemple : MEMCOMPARE "A:" TO "C:/DISQUE.DUR/COMPTA"

MEMRESTORE : Restaure sur disque dur les fichiers ayant ete sauvegarde par MEMBACKUP.
Syntaxe : MEMRESTORE ‹chaine_1› TO ‹chaine_2›

MID$: MID$ permet d'extraire une sous-chaine d'une chaine.
Syntaxe : MID$ (‹chaine›,‹valeur de depart›,‹nombre de caracteres›)

MIN : Donne la plus petite valeur des deux parametres.
Syntaxe : MIN (‹expression numerique›,‹expression numerique›)

MKDIR : Creation d'un sous-repertoire.
Syntaxe : MKDIR "‹chaine›"

MOD : Donne la valeur d'une expression modulo une autre expression selon la formule : MOD (x,y) = x -y * INT (x/y), soit le reste de la division entiere de x par y.
Syntaxe : MOD (‹expression numerique›,‹expression numerique›)

NEW : Cette instruction efface completement le programme qui etait resident en memoire centrale. Elle remet a zero l'ensemble des tables utilisees par l'interpreteur MEMBASIC. La table des variables est egalement effacee. Les fichiers qui etaient encore ouverts avant l'execution de cette instruction sont fermes.

NEXT : Est utilisee conjointement avec l'instruction FOR pour marquer la fin de la portee de la boucle.

ON...GOSUB : Branchements vers des sous-programmes.
Syntaxe : ON ‹expression› GOSUB ‹liste› ELSE ‹numero de ligne›
‹etiquette›
‹instruction›
‹liste› a la forme : ‹ligne›,‹ligne›,....
‹etiquette›,‹etiquette›,...
‹ligne›,‹etiquette›,....
ON...GOTO : Branchements vers des endroits du programme.
Syntaxe : ON ‹expression› GOTO ‹liste› ELSE ‹numero de ligne›
‹etiquette›
‹instruction›

OPEN : Ouverture d'un fichier Prodos.
Syntaxe : OPEN ‹option›,‹num de fichier›,‹specif. de fichier›[,‹longueur›]
‹option›: expression chaine dont le premier caractere indique le mode d'ouverture du fichier :
"I" pour lire seulement.
"A" pour ajouter en fin de fichier.
"O" pour ecrire dans un fichier apres creation.

"R" pour lire ou ecrire n'importe ou dans le fichier.
‹num de fichier›: est une expression numerique entiere dont la valeur sera associee au fichier.
‹specif. de fichier›: est une expression chaine contenant le nom du fichier a ouvrir avec la syntaxe ProDos. Ce nom pourra contenir des specifications de disque ou de repertoire.
‹longueur›: est un parametre optionnel de valeur implicite 128. Il sert uniquement lorsque l'option "R" a ete choisie pour definir la longueur d'un enregistrement.
Remarque : L'instruction OPEN ne sert pas pour ouvrir des fichiers MEMFILE. Pour ceux-ci, l'ouverture se fait par LET "#OPEN...".

OPEN "COM..." : Ouverture du fichier de communication.
Syntaxe : OPEN ‹expression chaine›,‹numero de fichier›
‹expression chaine› permet de selectionner le port choisi sous la forme "COM1" ou "COM2".
‹numero de fichier› : valeur entiere associee au fichier.

OPTION : Cette declaration, selon la forme utilisee, permet de declarer la valeur implicite de depart des indices, ou l'indice maximum implicite, ou l'unite utilise pour les angles dans les fonctions trigonometriques.
Syntaxe : OPTION BASE ‹index›
OPTION DIM ‹index›
OPTION ANGLE DEGREES
OPTION ANGLE RADIANS

ORD : Donne le code du premier caractere de la chaine parametre.
Syntaxe : ORD (‹chaine›)

PATH : Indique un ou plusieurs chemins pour la recherche des fichiers.
Syntaxe : PATH "‹chemin› [;‹chemin›]..."
‹chemin› doit satisfaire a la syntaxe du systeme d'exploitation.
Exemple : PATH "C:/DISQUE.DUR/COMPTA; A:/DISQUETTE/TRAVAIL"

PATH$: Donne la liste des chemins enregistre par PATH.

PI : Donne la valeur de la constante Pi = 3.1415926535898

POINTEUR# : Permet d'indiquer le numero d'enregistrement dans un fichier ProDos utilise en mode direct.
Syntaxe : POINTEUR # ‹num. de fichier›,‹num. d'enregistrement›
Exemple :
```
100 OPEN "R",1,"fiches",72    ! ouvre le fichier
110 POINTEUR #1,5             ! position au 5ieme rang
120                           ! soit au 72*4 ieme octet
130 LINE INPUT a$
```

POS : Indique la position d'une sous-chaine dans une chaine.
Syntaxe : POS (‹chaine_source›,‹sous-chaine›)
POS (‹chaine_source›,‹sous-chaine›,‹index›)

PRINT : Affiche les information a l'ecran ou sur imprimante.
Syntaxe : PRINT ‹liste d'edition›
La liste d'edition est composee d'elements d'editions separe les uns des autres par une virgule ou par un point.

PRINT # : Permet d'ecrire des informations dans un fichier ProDos ou dans un fichier de communication.
Syntaxe : PRINT #‹num. de fichier›,‹liste d'edition›
Conseil : Utiliser de preference ensemble les couples :
- PRINT # ... LINE INPUT #
- WRITE # ... INPUT #

- PRINT # ... INPUT $

pour retrouver les donnees a la lecture telles qu'elles etaient a l'ecriture.

PRINT USING : Permet d'obtenir des editions formatees sur ecran ou sur imprimante.
Syntaxe : PRINT USING ‹expression chaine› : ‹liste d'edition› [;]
‹ligne image›
‹etiquette›
Le caractere ";" en fin de ligne permet d'eviter un retour a la ligne automatique.

PRINTER : (fonction) Rend le numero de peripherique de sortie en cours.
- Fenetre d'execution en cours sur l'ecran..... 0
- Imprimante en slot..... 1
- Imprimante en slot..... 2

PRINTER : (instruction) Indique que les instructions DIR, PRINT, LIST et les editions par masques MEMSCREEN sont destinees a un nouveau preripherique de sortie.
Syntaxe : PRINTER ‹numero de peripherique›
PRINTER ‹expression chaine›

PROGRAM : Cette commande donne un nom au programme et definit les variables destinees a recevoir des parametres au lancement du programme.
Syntaxe : PROGRAM ‹nom› [(‹parametre› [,‹parametre› ...])]
‹nom› chaine indiquant le nom du programme.
‹parametres› noms des variables simples ou noms des tableaux. Les noms des tableaux doivent etre suivis de () (,) ou (,,)

RAD : Cette fonction convertit un nombre exprime en degres en radians.
Syntaxe : RAD (‹expression numerique en degres›)

RANDOMIZE : Permet de changer le point de depart d'une sequence de nombres pseudo aleatoires.

READ : Permet de lire des donnees qui sont incorporees au programme au moyen de l'instruction DATA.
Syntaxe : READ ‹variable›,‹variable›,...
READ IF MISSING THEN ‹numero de ligne› : ‹variable› [,...]
READ IF MISSING THEN EXIT DO : ‹varaiable› [,‹variable›]...
READ IF MISSING THEN EXIT FOR : ‹variable› [,‹variable›]...
Si une instruction READ est executee et que les donnees non encore lues dans les instructions DATA sont en nombre insuffisant pour satisfaire la liste de variables de l'instruction READ, alors deux cas se presentes :
- L'option IF MISSING est presente et alors il y a branchement au numero de ligne indiquee ou sortie de la boucle de lecture en cours par EXIT DO ou EXIT FOR.
- L'option est absente et l'erreur no 8001 est detectee.

REM : L'intruction REM permet d'introduire des commentaires dans un programme
Syntaxe : REM ‹commentaire›
! ‹commentaire›

REMAINDER : Donne le reste de la division entiere des deux parametres.
Syntaxe : REMAINDER (‹expression numerique›,‹expression numerique›)

RENAME : Permet de renomer un fichier.
Syntaxe : RENAME ‹ancien nom› TO ‹nouveau nom›
Si le nouveau nom est precede de @, un eventuel fichier existant portant ce nom sera detruit.

RENUMBER : Permet de renumeroter les numeros des lignes d'un programme. Par defaut, RENUMBER renumerote l'ensemble du programme a partir de 100 avec un pas de 10. Le pas peut etre modifier grace a l'instruction STEP.

REPEAT$: Permet de construire une chaine par repetition de la chaine donnee comme parametre.
Syntaxe : REPEAT$ (‹expression chaine›,‹nombre de fois›)

REPLAY : Permet de rejouer une sequence clavier enregistree.
Syntaxe : REPLAY ‹expression chaine›

RESTORE : Permet de relire des donnees au moyen de l'instruction READ.
Syntaxe : RESTORE
RESTORE ‹numero de ligne›
RESTORE ‹etiquette›

RETRY : Cette instruction est utilisee dans la routine d'anomalie pour demander une nouvelle execution de l'instruction qui a provoque l'erreur.

RIGHT$: La fonction RIGHT$ extrait une sous-chaine a partir des derniers caracteres de la chaine passee en parametre.
Syntaxe : RIGHT$ (‹expression chaine›,‹index›)

RMDIR : Cette instruction permet de supprimer un repertoire vide.
Syntaxe : RMDIR ‹expression chaine›

RND : Cette fonction donne le prochain nombre pseudo aleatoire dans une sequence pres definie entre 0 et 1.

ROUND : Effectue un arrondi sur la position souhaitee.
Syntaxe : ROUND (‹expression numerique›,‹index›).

RTRIM$: Cette fonction supprime les espaces situes a droite de chaine passee comme parametre.
Syntaxe : RTRIM$ (‹expression chaine›).

RUN : Lance l'execution du programme en memoire.
Syntaxe : RUN
RUN ‹specification de fichier›
RUN ‹etiquette›

SAVE : Sauvegarde du programme sur disque dur ou disquette.
Syntaxe : SAVE "⌊@⌋ ⌊$⌋ ‹specification de fichier"
@ pour ecraser un fichier deja existant.
$ pour proteger un programme.

SEC : Calcul de la secante de l'angle passee en parametre.
Syntaxe : SEC (‹expression numerique›)

SELECT : Cette instruction permet de demarrer une serie de blocs CASE et devaluer l'expression dont la valeur sera utilisee pour les tests.
Syntaxe : SELECT CASE ‹expression›

SET : Permet de fixer un certain nombre de parametres utilises par MEMSOFT :
- MARGIN : la longueur de la ligne.
- ZONEWIDTH : Longueur du chanp d'edition.
- CURLINE : Ligne d'edition pour MEMSCREEN.
- MAXLINE : Nombre de lignes par page d'imprimante.

Syntaxe : SET ‹parametre› ‹valeur›
Remarque : l'instruction ASK ‹parametre› permet de connaitre les valeurs par defaut.

SGN : SGN renvoi -1,0 ou 1 selon que l'expression donnee comme parametre est negative, nulle ou positive.
Syntaxe : SGN (‹expression numerique›).

SIN : Calcul du sinus de l'angle passe comme parametre.
Syntaxe SIN (‹expression numerique›).

SIZE : Donne la taille d'un tableau ou de chacun de ses indices.
Syntaxe : SIZE (‹nom de tableau›)
SIZE (‹nom de tableau›,‹index›)

SKIP : Permet un branchement rapide vers la fin d'une boucle FOR ou DO pour passe a l'element suivant.
Syntaxe : SKIP FOR
SKIP DO

SPC : Cette fonction speciale ne peut etre utilisee que pour l'instruction PRINT; elle permet d'imprimer une suite d'espaces.
Syntaxe : SPC (‹nombre d'espaces›).

SQR : Calcul de la racinne carree d'un nombre.
Syntaxe : SQR (‹expression numerique›).

STATUS : Donne le resultat de la derniere operation d'Entree/Sortie effectuee sur fichier ou sur masque.

STEP : Permet de fixer le pas implicite pour les instruction AUTO et RENUMBER
Syntaxe : STEP (‹pas›).

STR$: Convertit une valeur numerique en chaine de caracteres.
Syntaxe : STR$ (‹expression numerique›).

SYSTEM : Fait quitter MEMSOFT pour revenir au niveau systeme d'exploitation.

TAB : Ne peut etre utilisee qu'avec l'instruction PRINT et permet de positionner le curseur ou la tete d'impression a la colonne indiquee.
Syntaxe : TAB (‹colonne›).

TAN : Calcul de la tangente de l'angle passe comme parametre.
Syntaxe : TAN (‹expression numerique›).

TIME : Donne l'heure compte en secondes.

TIME$: Donne l'heure au format HH:MM:SS.

TRACE : Selectionne ou enleve le mode trace dans lequel les lignes executees sont listees.
Syntaxe : TRACE ON
TRACE OFF.
Lorsque le mode TRACE est actif et que le programme n'attend pas de donnees au clavier, deux actions sont possibles :

- L'enfoncement de la touche OPTION suspend l'affichage des lignes tracees dans la fenetre TRACE. Cela permet un acces plus rapide a la partie du programme a mettre au point. Les lignes sont a nouveau tracees des que la touche OPTION est relachee.
- La touche SHIFT suspend l'execution du programme. Tant que l'une des touches SHIFT est enfoncee , l'execution du programme est arretee. Ont peut alors :
  - avancer d'une instruction en enfoncant puis en relachant la touche Controle (sans relache la touche SHIFT).

- passer en mode "gestion des fenetres" en enfoncant la touche OPTION Clic-Souris pour consulter plus facilement la liste des lignes tracees.
- stopper le programme en enfoncant "POMME CONTROLE C" (sauf si le programme comporte une gestion des exceptions).

Au remachement de la touche SHIFT, le programme reprend son execution et sa trace normalement.

TRUNCATE : Cette fonction effectue une troncature de la valeur donne comme parametre.
Syntaxe : TRUNCATE (‹expression numerique›,‹index›).

UBOUND : Cette fonction rend la valeur maximale d'un indice.
Syntaxe : UBOUND (‹nom de tableau›,‹index›)
UBOUND (‹nom de vecteur›).

UCASE$: Restitue une chaine ne comportant que des majuscules.
Syntaxe : UCASE$ (‹expression chaine›).

USING$: Donne une chaine obtenue en formant le resultat d'une expression.
Syntaxe : USING$ (‹expression chaine›,‹expression›).
‹expression chaine› est une chaine format.

VAL : Cette fonction convertit la valeur numerique representee par une chaine en un nombre.
Syntaxe : VAL (‹expression chaine›).

VERSION$: Rend une cahine indiquant le nom de la machine et de la version de MEMSFT. La fonction ne comporte pas de parametre. La cahine rendue contient :
‹nom de la machine›,‹nom du produit›:‹version›
Exemple : PRINT VERSION$
APPLE II GS,MEMSOFT:2.08

VTAB : (fonction) Donne la ligne ou est situe le curseur dans la fenetre d'execution.

VATB : (instruction) Fixe la ligne du curseur dans la fenetre d'execution.
Syntaxe : VTAB (‹index›).

WHEN : Permet de traiter une erreur detectee lors de l'execution d'un programme.
Syntaxe : WHEN EXCEPTION GOTO ‹no ligne›
‹etiquette›
WHEN EXCEPTION BREAK

WRITE # : Permet d'ecrire des informations dans un fichier ProDos ou dans un fichier de communication.
Syntaxe : WRITE #‹no de fichier›,‹liste d'expressions›

CHAPITRE 2 : MEMSCREEN

Toutes les instruction de MEMSCREEN ont la forme suivante :

LET ‹exp. chaine›

Pour les mots cles, la premiere lettre seule est significative :

"#O", "#o", "#OPEN" et "#open" sont equivalents.

2.1 CREATION OU MODIFICATION D'UN MASQUE

Un masque est associe a une fenetre ecran. Pour chaque masque, il faut definir : - les specification de la fenetres :
- taille
- cadre
- position
- couleur
- ...

- le texte du masque.
- les zones de saisie et d'affichage.

L'instruction de creation d'un masque est ..: LET "#NEW,M,<nom du masque>"
Celle de modification est .....: LET "UPDATE,M,<nom du masque>"

2.2 LES ZONES D'UN MASQUE

La definition d'une zone se fait en seux temps :
- Position et longueur.
- Caracteristiques.

2.2.1 Position et longueur

Le debut de la zone sur l'ecran sera note : [
Les autres caracteres de la zone seront notes :]
F6 permet de detruire la zone ou se trouve le curseur

2.2.2 Caracteristiques

F4 permet de fixer les caracteristiques de la zone ou se trouve le curseur.
Voici les questions posees :

```
Nom....
Cond...
En sortie N (O/N)        Longueur : 002
Numerique N (O/N)        Re calcul N (O/N)

A effacer N (O/N)        A droite. N (O/N)
Date..... N (O/N)        'O'ou'N'. N (O/N)
Espaces.. N (O/N)        Majuscul. N (O/N)
```

Il faut repondre a ces questions.

2.3 MASQUES ET IMPRIMANTE

MEMSCREEN offre un large choix de commandes et instructions pour obtenir des copies papier de tout ou partie d'un masque.

2.3.1 COPIE MANUELLE

En mode "FENETRE", la fenetre designee par la main sera recopiee sur imprimante en enfoncant la touche "H".

2.3.2 COPIE PAR PROGRAMME

C'est l'instruction LET "HARDCOPY...", decrite plus loin qui permet l'impression d'une partie rectangulaire de taille quelconque.

2.4 FICHIERS D'AIDE

Plusieurs fichiers d'aide sont disponibles avec le systeme MEMSOFT :

- Des fichiers standard qui contiennent des informations sur la syntaxe des instruction utilisees dans MEMBASIC , MEMFILE et MEMSCREEN.
- Des fichiers d'aide que le developpeur peut facilement construire pour aider l'utilisateur de programmes d'application fonctionnant sous MEMSOFT.

2.4.1 FICHIERS D'AIDE APPLICATION

Dans un programme en cours, le fichier d'aide appele au moyen de la touche F10 est ene priorite le fichier defini par l'instruction HELP. Si aucun fichier n'a ete defini par cette instruction, le fichier d'aide recherche est d'abord celui qui porte le meme nom que le masque MEMSCREEN utilise, puis celui portant le nom du programme en cours et enfin le fichier d'aide FRSOS.HLP . Les fichiers d'aide ont toujours le suffixe .HLP. Par exemple, si le programme ESSAI.PRG contient les lignes :

```
100 LET "#OPEN,1,M.MA1"
110 LET "INPUT,1"
```

Si l'utilisateur enfonce la touche d'aide F10 ou POMME0 pendant l'execution de la ligne 110, le fichier d'aide chercher sera :

- d'abord le fichier MA1.HLP
- s'il n'est pas trouve, le fichier ESSAI.HLP
- enfin FRSOS.HLP

2.4.2 CREATION DE FICHIERS D'AIDE

Le format des fichiers d'aide est celui des masques MEMSCREEN. En fait, tout masque MEMSCREEN peut servir de fichier d'aide.

2.5 SEQUENCE CLAVIER

Les sequences clavier pre-enregistrees permettent de memoriser puis de rejouer des sequences de touches tapees au clavier. Ces sequences sont sauvegardees sur disque avec un suffixe .AUT
L'enregistrement et la relecture de sequences clavier se font a n'importe quel moment, quel que soit le programme et ne demande aucune programmation.

2.6 INSTRUCTIONS MEMSCREEN

LET "#CLEAR..." : Supprime un masque de la memoire et le fait disparaitre de l'ecran. Pour utiliser a nouveau ce masque, il faut le reouvrir par la commande LET "#OPEN..."
Syntaxe : LET "#CLEAR,‹indentificateur›"

LET "#CLEAR,$" : Supprime de la memoire tout les masque.

LET "#DELETE..." : Detruit un masque sur disque.
Syntaxe : LET "#DELETE,M,‹nom›"

LET "#NEW..." : Creer un masque. Cette instruction donne la main a l'utilisateur qui doit definir son masque : la taille, le texte, les zones,
Syntaxe : LET "#NEW,M,⌊@⌋‹nom"

LET "#OPEN..." : Ouvre un masque, c'est a dire le charge en memoire. Le masque peut alors etre utilise pour faire des saisies ou des affichages.

```
Syntaxe : LET "#OPEN [<option>],<identificateur>,M,<nom>"
          <option> : /V Le masque reste visible apres utilisation.
                     /U Le masque devient invisble en fin d'utilisation.

LET "#UPDATE..." : Permet de modifier un masque qui a deja ete cree au
prealable par LET "#NEW..."
Syntaxe : LET "#UPDATE,M,<nom>"

LET "?..." : Rechercher et recopier une ligne du dernier masque utilise sur
le peripherique de sortie implicite.
Syntaxe : LET "?,<repere>"
          <repere> est un caractere du jeu ISO superieur a 32.
Exemple : soit le masque :    XVoici un test de masque et d'edition
                              AGiboulees de Mars
                              BAvril
                              ACette ligne n'est pas imprime

          LET "?,A" imprimera : Giboulees de Mars.

LET "#CHARGE..." : Affiche le texte du masque, et le texte seul. Le masque
doit deja etre ouvert.
Syntaxe : LET "#CHARGE [<option>],<identificateur>"

LET "#HARDCOPY..." : Recopie sur le peripherique de sortie, une partie d'un
masque.
Syntaxe : LET "#HARDCOPY [<option>],<identificateur>, [<lmini> [,<lmaxi>
                                                  [,<cmini> [,<cmaxi>]]]]"
          <option> peut etre soit /K soit rien. /K permet d'editerla ligne
          sans 'RETOUR CHARIOT'. Ceci permet d'avoir plusieurs largeurs de
          caracteres sur une meme ligne.

LET "INPUT..." : Saisir des zones du masque qui ont ete definies en entree.
Syntaxe : LET "INPUT [/K],<identificateur>,[<zentree>[,<zmini>[,<zmaxi>]]]"
          </K> interdit la fin de la saisie par une autre touche que
          "Esc" ou une touche de fonction.
          <zentree> zone de debut de la saisie
          <zmini> et <zmaxi> indiquent les numeros des zones ou doit
          s'effectuer la saisie.

LET "KEYBOARD" : Attendre une validation ou annulation sans saisir de zone de
masque.
Syntaxe : LET "KEYBOARD [/K]"

LET "MODE..." : Change l'option /U, /V d'un masque.
Syntaxe : LET "MODE [<option>], <identificateur>"

LET "OUTPUT..." : cette instruction sert a afficher le contenu des variables
d'un masque.
Syntaxe : LET "OUTPUT [<option>],<identificateur>[,<typzone>
                                                  [,<zonemini>
                                                  [,<zonemaxi>]]]"
          <option> peut etre /U ou /V
          <typzone> peut etre A : toutes les variables.
                              I : les variables en entrees seulement.
                              O : les variables en sorties seulement.

LET "PRINT..." : Afficher le texte d'un masque et le contenu des variables
d'un masque.
Syntaxe : LET "PRINT  [<option>],<identificateur>[,<typzone>
                                                  [,<zonemini>
                                                  [,<zonemaxi>]]]"
```

LET "SCROLL..." : Fait defiler ou efface des lignes d'un masque et insere une ligne blanche.
Syntaxe : LET "SCROLL [<option>],<identificateur>,<lmni>,<lmaxi>"
<option> peut etre : /U defilement vers le haut.
/D defilement vers le bas.
/C effacement.

LET "TAKE..." : Remet a jour les variables dans MEMBASIC en accord avec le contenu actuel d'un masque. Le masque doit etre ouvert.
Syntaxe : LET "TAKE [<option>],< identificateur>"
<option> peut etre : /Z une remise a zero a lieu aussi sur l'ecran qu'en memoire.
rien toute les variables en entree du masque sont mises a jour en fonction de leur valeur dans le masque.

```
Exemple : 100 ! nouvelle entree
          110 LET "TAKE/Z,1"  ! vide zones et variables
          120 LET "INPUT,1"   ! saisie sur zones vides
          ...
          ...
          400 LET "INPUT,1"   ! saisie des valeurs
          410 x=1             ! detruit une des valeurs saisie en 400
          420 LET "TAKE,1"    ! restitue cette valeur
```

LET "Y..." : Permet de faire deplacer verticalement, par l'utilisateur, une barre horizontale en fond inverse dans un rectangle defini d'un masque.
Syntaxe : LET "Y [<option>],<identificateur>[,<pos>,<lmini>,<lmaxi>
<cmini>,<cmaxi>] "
<option> peut etre /K ou rien
<pos> position initiale de la barre.

CHAPITRE 3 : MEMFILE

3.1 INTRODUCTION

MEMFILE est un gestionnaire de fichier. Il permet de sauver et de retrouver sur disque de grandes quantites de donnees classees sous formes de de fiches ou enregistrements.
Un fichier MEMFILE peut se decrire simplement de la facon suivante :

- D'une part l'ensemble de toute mes fiches.
- D'autre part, un ensemble de cles permettant de retrouver les fiches.

3.2 OPERATIONS GLOBALES

Voici les differentes operations dites globales sur un fichier :

- Creation : Definit la structure d'une fiche et prepare le fichier sur le repertoire selectionne du disque. Le fichier ne contient Le fichier ne contient alors aucune fiche.
- Destruction : Le fichier est efface du catalogue du repertoire et les fiches qu'il pouvait contenir sont perdues.
- Ouverture : Avant d'utiliser un fichier , il est indispensable de le declarer au systeme. Cette operation s'appelle " ouverture du fichier". MEMFILE renseigne alors le programme sur la structure d'une fiche et se prepare a effectuer des operations de lecture ou d'ecriture sur celui-ci.
- Fermeture : Les operations d'ecritures ou de mise a jour d'un

fichier sont 'tamponnees'.Autremenr dit, les donnees a ecrire sur un fichier transitent par une zone de memoire intermediaire qui n'est recopiee sur disque que lorsqu' elle est saturee. Il se produit donc assez souvent un decalage dans le temps entre la demande d' ecriture et sa realisation effective . En fin d'utilisation d'un fichier, afin d'eviter de perdre les ecritures 'en cours', il y a lieu de demander la fermeture du fichier.
A noter que, lorsqu'un programme MEMBASIC termine son execution tous les fichiers ouverts sont automatiquement fermes.

3.3 LE DICTIONNAIRE

Le dictionnaire sert a decrire la structure d'une fiche pour un fichier determine. Il est defini au moment de la creation du fichier et est conserve de maniere permanente dans le fichier afin d'etre relu au moment des ouverture. Il peut etre visualise par l'ordre : LET "ENTER,‹indentificateur›" suivi de l'ordre LET "VISUALISE".

3.3.1 Les differentes zones d'un dictionnaire

- Les zones de type cle. Elles contiennent des informations qui permettent de retrouver rapidement la fiche dans le fichier. Une fiche peut ne posseder qu'un seul moyen d'acces. On dit alors que le fichier est mono cle.Le nombre de moyens d'acces a une fiche peut etre multiple. On parle alors d' un fichier multicles. Dans les deux cas, chaque moyen d'acces peut etre forme de plusieurs variables de differents types.
- La zone de type enregistrement. C' est une zone qui ne sert pas de moyen d'acces, mais qui peut contenir de grandes quantites de donnees. Cette zone se definit sous la forme d'une liste de variables MEMBASIC separees des zones cles par le symbole "=".

3.3.2 DEFINITION D'UN DICTIONNAIRE

Le dictionnaire d'un fichier est memorise sur disque avec le fichier. Il n'est donc a specifier qu'une seule fois lors de la creation du fichier.
La syntaxe d'un dictionnaire est la suivante :

- vc represente les Variables Cles.
  ve represente les Variables Enregistrements.

- Dans le cas d'une cle unique :
  LET "›vc1,vc2,...,vcn = ve1,ve2,...vep"

- Dans le cas de plusieurs cles :
  LET "›vc11,...,vc1n & vc21,...,vc2p & ... = ve1,...vek"

3.4 ACTIONS AU NIVEAU DE LA FICHE

3.4.1 Operations d'ecriture

L'ecriture d'un enregistrement ne doit se faire que lorsque toutes les variables cle et enregistrement sont pretes.
Deux modes sont possibles suivant que l'on autorise ou pas les cles homonymes (ordre ADD ou WRITE).
dans le cas ou les homonymes sont permis, on peut preciser si l'insertion de la fiche s'effectue en tete (option /FIRST) ou en queue (/LAST) de la liste d'homonymes correspondente.
Contrairement a l'ecriture, la mise a jour (ordre UPDATE) d'une fiche ne modifie que la partie enregistrement de celui ci.

3.4.2 Operations de lecture

MEMFILE permet deux modes de lecture :

- La lecture d'une fiche (ordre READ), suite a une recherche sur une des cles. Dans ce cas, et s' il existe des homonymes, on ne peut atteindre que la premiere fiche de la liste des homonymes (option /FIRST), ou bien la derniere (option /LAST)
- La lecture sequentille (ordre NEXT), ou sequentielle inverse (ordre NEXT option PREVIOUS).

Pour chacune des cles d'un fichier, MEMFILE gere un curseur courant qui :

- est place au debut du fichier a l'ouverture.
- est mis a jour a chaque operation d'ecriture de type WRITE ou de lecture de type READ.
- est avance d'une cle par suite d'un ordre NEXT execute normalement.
- est recule d'une cle par suite d'un ordre NEXT/P.

Le curseur associe a chaque cle peut etre replace en tete (ordre ZERO/FIRST) ou en fin de fichier (ordre ZERO/LAST).

3.5 SYNTAXE

Toutes les instruction de MEMFILE ont la forme suivante :

LET ‹exp. chaine›

Pour les mots cles, la premiere lettre seule est significative :

"#N", "#n", "#NEW" et "#new" sont equivalents.

3.6 LES INSTRUCTIONS

LET "#CLEAR..." : Cette instruction ferme un fichier.
Syntaxe : LET "#CLEAR,‹identificateur›"

LET "#CLEAR,$" : Ferme tout les fichiers ouverts. Ferme egalement les masques MEMSCREEN ouverts.

LET "#CLEAR,$$" : Ferme tout les fichiers ouverts. Tout les fichiers ouverts par LET"#OPEN..." sont fermes. Cette instruction n'efface pas de la memoire les masques ouverts, ni le tampon des dictionnaires. C'est la seule difference entre LET "#CLEAR,$" et LET "#CLEAR,$$".

LET "#CLEAR,$‹d›" : ferme tout les fichiers ouverts sur une unite de disque.
Syntaxe : LET "#CLEAR,$‹disque›"
‹disque› aura la forme : A: B: C: ...

LET "#DELETE,F..." : Detruit un fichier MEMFILE sur disque. Chaque fichier MEMFILE est compose de deux fichiers simples sur disque de suffixes .MFK et .MFR . Les deux fichier ‹nom›.MFK et ‹nom›.MFR sont detruits.
Syntaxe : LET "#DELETE,F,‹nom›
Exemple : LET "#DELETE,F,/SUB/FIC1"

LET "#GARBAGE..." : Compactage d'un fichier MEMFILE.
Syntaxe : LET "#GARBAGE,F,‹nom› [› ‹chemin›]"
Exemple : LET "#GARBAGE,F,ecriture›B:/poubelle"

LET "#NEW..." : Creation d'un fichier MEMFILE. La creation d'un fichier necessite que le dictionnaire du fichier ait ete defini au prealable.
Syntaxe : LET "#NEW,F,‹nom›"
Exemple : LET "›a=b,c" ! Cle numerique 'a' et un enregistrement de deux champs 'b' et 'c'.
LET "#NEW,F,test" ! creation du fichier 'test'.

LET "#OPEN..." : Ouverture d'un fichier MEMFILE. Le fichier est ouvert aussi bien pour la lecture, l'ecriture ou la mise a jour.
Syntaxe : LET "#OPEN,‹identificateur›,F,‹nom›"

LET "+..." : Complete une definition de dictionnaire. La chaine qui suit le signe plus est ajoute a celle deja contenue dans le tampon.
Syntaxe : LET "+‹constante chaine›"

LET "›..." : Met une chaine dans le tampon des dictionnaires. Deux formes sont possibles : LET "›" vide le tampon.
LET "›..." vide le tampon et y met la chaine.
Syntaxe : LET "›‹constante chaine"

LET "ADD..." : Ecriture d'un nouvel article dans un fichier MEMFILE avec la possibilite de d'homonymie.
Syntaxe : LET "ADD ⌊‹option›⌋,‹identificateur›"
‹option› peut etre /F ou /L

LET "BORNE..." : Permet de fixer une borne superieure ou inferieure aux lectures sequentielle d'un fichier MEMFILE suivant l'une des cles.
Syntaxe : LET "BORNE ⌊‹option›⌋,‹identificateur›⌊,‹cle›⌋"
‹option› peut etre /F, /L ou /D

LET "DELETE..." : Detruit un enregistrement dans un fichier MEMFILE.
Syntaxe : LET "DELETE ⌊‹option›⌋,‹identificateur›⌊,‹cle›⌋"
‹option› peut etre /F ou /L

LET "ENTER..." : Relecture du dictionnaire d'un fichier.
Syntaxe : LET "ENTER,‹indentificateur›"

LET "NEXT..." : Lecture sequentielle d'un article dans un fichier MEMFILE dans le sens croissant ou decroissant.
Syntaxe : LET "NEXT ⌊‹option›⌋,‹identificateur›⌊,‹cle›⌋"
‹option› peut etre /K, /R, /A ou /P

LET "READ..." : Lecture d'un article dans un fichier MEMFILE suivant une cle.
Syntaxe : LET "READ ⌊‹option›⌋,‹identificateur›⌊,‹cle›⌋"
‹option› peut etre /K, /R, /A, /F ou /L

LET "UPDATE..." : Met a jour un enregistrement d'un fichier MEMFILE.
Syntaxe : LET "UPDATE ⌊‹option›⌋,‹indificateur›⌊,‹cle›⌋"
‹option› peut etre /F ou /L

LET "VISUALISE" : Visualise le tampon contenant les dictionnaires de fichiers

LET "WRITE..." : Ecriture d'un nouvel artcle dans un fichier MEMFILE sans possibilite d'homonymie.
Syntaxe : LET "WRITE,‹identificateur›"

LET "XINDEX..." : Donne la valeur de la prochaine cle relative qui sera creee par defaut dans un fichier relatif. Si la valeur de la cle relative n'est pas specifiee lors d'une ecriture dans un fichier relatif, l'article sera cree avec le premier numero libre. L'instruction XINDEX permet de conaitre cette valeur.
Syntaxe : LET "XINDEX,‹indentificateur›"

LET "ZERO..." : Positionnement sur la borne inferieure ou superieure d'un fichier MEMFILE.
Syntaxe : LET "ZERO [<option>],<identificateur>[,<cle>]"
<option> peut etre /F ou /L

3.7 ANNEXE : LES OPTIONS

3.7.1 : L'option /K/A/R

Cette option est utilisee dans les instructions LET "READ...", LET"NEXT..."
L'option /K permet de ne lire que la cle specifiee.
L'option /A permet de lire toutes les cles.
L'option /R permet de ne lire que l'enrigistrement.
L'option implicite est /A/R.

3.7.2 : L'option /F/L

Cette option peut avoir differents sens selon l'instruction dans laquelle elle est utilisee.
* Dans les instructions : LET "READ..."
LET "UPDATE..."
LET "ADD..."
LET "DELETE..." elle permet de dire comment doit se faire la recherche ou l'insertion en cas d'homonymies.
/F La recherche ou l'insertion se fait en debut de liste d'homonymes.
/L La recherche ou l'insertion se fait en fin de liste d'homonymes.
L'option implicite est /F dans tous ces cas.
* Dans l'instruction : LET "BORNE..."
/F permet de designer la borne inferieure.
/L permet de designer la borne superieure.
* Dans l'instruction : LET "ZERO..."
/F Positionnement sur la borne inferieure du fichier ou en debut de fichier s'il n'y a pas de borne.
/L Positionnement sur la borne superieure du fichier ou en fin de fichier s'il n'y a pas de borne.
L'option implicite est /F.
