ATARI ST+STE

MISE À JOUR

GFA BASIC 3.5E

DU GFA BASIC ET GFA COMPILATEUR 3.0/3.5 VERS LA VERSION 3.5E

UN LOGICIEL GFA

EDITIONS MICRO APPLICATION

Frank OSTROWSKI
Gottfried P. ENGELS

MISE A JOUR

GFA BASIC

INTERPRETEUR COMPILATEUR 3.5E

EDITIONS MICRO APPLICATION

Auteurs Frank Ostrowski & Gottfried P. Engels

Collection dirigée par Philippe Olivier
Edition réalisée par Frédérique Beaudonnet

Introduction

Vous venez d'acquérir la version 3.5E des nouveaux interpréteurs et compilateurs GFA Basic. Cet ouvrage décrit toutes les nouvelles possibilités de ces versions. Vous pouvez donc en prendre note après avoir lu les manuels de référence de l'interpréteur et du compilateur.

Lorsque vous découvrirez ce langage, votre première réaction pourrait être de penser : pourquoi une version 3.5E et non pas 3.8 ou 4.0 ?

Disons que les numéros de version des logiciels sont à l'image des gammes que proposent les constructeurs d'automobiles. Un changement d'unité correspond à un modèle radicalement différent de ses prédécesseurs : nouveau moteur, nouvelle carrosserie, nouveaux équipements.

Dans le cas du GFA Basic, nous avons ajouté des équipements de choix, un éditeur encore plus confortable, des fonctions d'édition plus puissantes, mais aussi gonflé sensiblement son moteur pour des performances plus marquées en matière de programmation : près de 40 instructions dédiées aux manipulations des matrices, de nouvelles fonctions statistiques mathématiques. Enfin, plusieurs fonctions ont été spécifiquement dédiées à la programmation du STE pour tirer pleinement parti des capacités sonores et graphiques du STE.

Ces outils peuvent jouer un rôle fondamental dans de nombreux domaines, de la gestion aux maths/physique, du graphisme au traitement du signal.

Le programmeur en verra immédiatement les avantages : matrices pour le calcul vectoriel, matrices représentant les coordonnées d'un objet 3D, gestion du son DMA sur STE, détection d'une machine STE, etc.

Dans le domaine du calcul, les étudiants, mathématiciens, physiciens, enseignants et bien d'autres, découvriront que le GFA Basic peut résoudre très rapidement des problèmes ardus, en simplifiant considérablement les tâches de programmation.

Dernière parenthèse, le GFA Compilateur 3.5E que vous possédez dorénavant grâce à cette mise à jour est bien entendu destiné à prendre en charge les nouvelles instructions du GFA Basic, qu'il optimisera aux mêmes titres que les 400 autres commandes du langage.

Et maintenant, hâtons-nous d'en découvrir les trésors !

Les auteurs.

Sommaire

1. Les nouvelles possibilités de l'éditeur

L'éditeur utilise 2 octets de plus pour chaque ligne de programme, ce qui rend possible le pliage des fonctions et augmente la vélocité du scrolling arrière.

La recherche d'une séquence de caractères se fait maintenant même dans les procédures et fonctions repliées.

Lors du listage, les labels sont décalés de 2 cm vers la gauche.

TAB :	Le curseur saute à la prochaine tabulation vers la droite.
Ctrl + TAB :	Le curseur revient à la tabulation précédente vers la gauche.
Shift-gauche + TAB :	Insère des espaces jusqu'à la prochaine tabulation.
Shift-droit + TAB :	Supprime les espaces entre le curseur et la précédente tabulation.

2. Opérations linéaires sur des vecteurs et des matrices

Toutes les fonctions définies dans ce chapitre font références à des tableaux de valeurs réelles ayant 1 ou 2 dimensions.

2.1. Initialisation

MAT BASE 0
MAT BASE 1

L'instruction MAT BASE n'a de sens que si OPTION BASE 0 figure dans le programme. On peut alors définir le début de la numérotation des lignes et colonnes de matrices ou vecteurs à 1 pour MAT BASE 1 et à 0 pour MAT BASE 0.

MAT BASE n influe sur les commandes suivantes

- MAT READ
- MAT PRINT
- MAT CPY
- MAT XCPY
- MAT ADD
- MAT SUB
- MAT MUL

La valeur prise par défaut est MAT BASE 1

Exemple

```
OPTION BASE 0
MAT BASE 1
DATA 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16
DIM a(3,3)
MAT READ a()
```

```
PRINT a(1,1)

      Renvoie la valeur 1
```

2.2. Opérations unaires (sur une matrice)

MAT CLR a()
MAT SET a()=x
MAT ONE a()

a : nom du tableau de valeurs numériques
x : expression réelle

MAT CLR a() correspond à ARRAYFILL a(), 0, ce qui initialise à la valeur 0 tous les éléments de a (matrice ou vecteur).

➥ Abréviation : m cl a()

MAT SET a()=x correspond à ARRAYFILL a(), x, ce qui initialise à la valeur x tous les éléments de a (matrice ou vecteur).

➥ Abréviation : m se a()=x

MAT ONE a() crée, à partir d'une matrice carrée, une matrice unité dont les éléments a(1,1), a(2,2), ..., a(n,n) sont égaux à 1, les autres étant nuls.

➥ Abréviation : m o a()

Exemple

```
DATA 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16
DIM a(3,3)
MAT READ a()
PRINT a(1,1)
MAT CLR a()
PRINT a(1,1)
```



```
      Renvoie la valeur 1, puis la valeur 0

DIM a(5,7)
FOR i%=1 TO 5
  FOR j%=1 TO 7
    a(i%,j%)=RAND(10)
  NEXT j%
NEXT i%
MAT SET a(),5.3
FOR i%=1 TO 5
  FOR j%=1 TO 7
    PRINT a(i%,j%)
  NEXT j%
NEXT i%

Renvoie 35 fois la valeur 5.3

DIM a(3,3)
MAT ONE a()
MAT PRINT a()

Renvoie          1,0,0

                 0,1,0
                 0,0,1
```

2.3. Opérations d'entrée-sortie

MAT READ a()
MAT PRINT [#i,]a()[,g,n]
MAT INPUT #i,a()

i, g, n : expression entière
a : nom du tableau de valeurs numériques

MAT READ a() lit une matrice ou un vecteur dans une zone de DATA.

➥ Abréviation : m r a()

Exemple

```
DATA 1,2,3,4,5,6,7,8,9,10
DIM a(2,5)
```

```
MAT READ a()
PRINT a(2,4)

            Renvoie la valeur 9
```

MAT PRINT [#i,]a()[,g,n] affiche une matrice (sous forme de tableau) ou un vecteur (sur une ligne) formatés comme avec STR$(x,g,n). De plus, la sortie peut être redirigée vers un fichier ou périphérique avec #i.

➥ Abréviation : m p [#i,]a()[,g,n] ou m ? [#i,]a()[,g,n]

Exemple

```
DATA 1,2.33333,3
DATA 7,5.25873,9.376
DATA 3.23,7.2,8.999
DIM a(3,3)
MAT READ a()
MAT PRINT a()
PRINT"---------------"
MAT PRINT a(),7,3

      Renvoie     1,2.33333,3
                  7,5.25873,9.376
                  3.23,7.2,8.999
                  ---------------
                  1.000,2.333,3.000
                  7.000,5.259,9.376
                  3.230,7.200,8.999
```

MAT INPUT #1,a() lit une matrice ou un vecteur dans un fichier ASCII. Les séparateurs et retours à la lignes peuvent être redéfinis, comme pour INPUT #.

➥ Abréviation : m i #i,a()

Exemple

```
OPEN "o",#1,"Test.DAT"
DIM a(3,3)
MAT ONE a()
MAT PRINT #1,a()
CLOSE #1
MAT CLR a()
OPEN "i",#1,"Test.DAT"
MAT INPUT #1,a()
CLOSE #1
MAT PRINT a()
      Renvoie     1,0,0
                  0,1,0
                  0,0,1
```

2.4. Opérations binaires (sur deux matrices)

MAT CPY a([i,j])=b([k,l])[,h,w]
MAT XCPY a([i,j])=b([k,l])[,h,w]
MAT TRANS a()[=b()]

a, b : nom des tableaux de valeurs numériques
i, j, k, l, h, w : expression entière

MAT CPY a([i,j])=b([k,l])[,h,w]

copie dans la matrice a, à partir de la ligne i et de la colonne j, h lignes de la matrice b à partir de la ligne k et w colonnes à partir de la colonne l.

➥ Abréviation : m c a(i,j)=b(k,l),h,w
m x a(i,j)=b(k,l),h,w
m t a()=b()

Exemple

```
DIM a(5,5),b(4,4)
MAT SET a()=1
FOR i%=1 TO 4
  FOR j%=1 TO 4
```

```
    b(i%,j%)=SUCC(i%)
  NEXT j%
NEXT i%
'
MAT PRINT a()
PRINT "-------------"
MAT PRINT b()
PRINT "-------------"
MAT CPY a(2,2)=b(2,2),3,3
MAT PRINT a()

      Renvoie 1,1,1,1,1
              1,1,1,1,1
              1,1,1,1,1
              1,1,1,1,1
              1,1,1,1,1
              -------------
              2,2,2,2
              3,3,3,3
              4,4,4,4
              5,5,5,5
              -------------
              1,1,1,1,1
              1,3,3,3,1
              1,4,4,4,1
              1,5,5,5,1
              1,1,1,1,1
```

Cas particuliers

MAT CPY a()=b() copie dans la matrice a, la matrice b, complètement si elles sont de même ordre, les éléments d'indices équivalents sinon.

➥ Abréviation : m c a()=b()

Exemple

```
DIM a(5,3),b(4,4)
MAT SET a()=-1
FOR i%=-1 TO 4
  FOR j%=-1 TO 4
    b(i%,j%)=SUCC(i%)
  NEXT j%
```



```
NEXT i%
'
MAT PRINT a()
PRINT "-------------"
MAT PRINT b()
PRINT "-------------"
MAT CPY a()=b(),3,3
MAT PRINT a()

      Renvoie 1,1,1
              1,1,1
              1,1,1
              1,1,1
              1,1,1
              -------------
              2,2,2,2
              3,3,3,3
              4,4,4,4
              5,5,5,5
              -------------
              2,2,2
              3,3,3
              4,4,4
              1,1,1
              1,1,1
```

MAT CPY a(i,j)=b() copie dans la matrice a à partir de la ligne i et de la colonne j, la matrice b, complète si possible, les éléments d'indices équivalents sinon. Ne pas oublier le rôle de MAT BASE.

➥ Abréviation : m c a(i,j)=b()

Exemple

```
DIM a(5,3),b(4,4)
MAT SET a()=-1
FOR i%=-1 TO 4
  FOR j%=-1 TO 4
    b(i%,j%)=SUCC(i%)
  NEXT j%
NEXT i%
'
MAT PRINT a()
PRINT "-------------"
MAT PRINT b()
PRINT "-------------"
```

```
MAT CPY a(2,2)=b(2,2),3,3
MAT PRINT a()
      Renvoie 1,1,1
              1,1,1
              1,1,1
              1,1,1
              1,1,1
              -------------
              2,2,2,2
              3,3,3,3
              4,4,4,4
              5,5,5,5
              -------------
              1,1,1
              1,3,3
              1,4,4
              1,5,5
              1,1,1
```

MAT CPY a()=b(i,j) copie dans la matrice a, la matrice b à partir de la ligne i et de la colonne j, complète si possible, les éléments d'indices équivalents sinon. Ne pas oublier le rôle de MAT BASE.

➥ Abréviation : m c a()=b(i,j)

Exemple

```
DIM a(5,3),b(4,4)
MAT SET a()=1
FOR i%=1 TO 4
  FOR j%=1 TO 4
    b(i%,j%)=SUCC(i%)
  NEXT j%
NEXT i%
'
MAT PRINT a()
PRINT "-------------"
MAT PRINT b()
PRINT "-------------"
MAT CPY a()=b(2,2),3,3
MAT PRINT a()

      Renvoie 1,1,1
              1,1,1
              1,1,1
```



```
              1,1,1
              1,1,1
              -------------
              2,2,2,2
              3,3,3,3
              4,4,4,4
              5,5,5,5
              -------------
              3,3,3
              4,4,4
              5,5,5
              1,1,1
              1,1,1
```

MAT CPY a(i,j)=b(k,l) copie dans la matrice a à partir de la ligne i et de la colonne j, la matrice b à partir de la ligne k et de la colonne l, complète si possible, les éléments d'indices équivalents sinon.

➥ Abréviation : m c a(i,j)=b(k,l)

Exemple

```
DIM a(5,3),b(4,4)
MAT SET a()=1
FOR i%=1 TO 4
  FOR j%=1 TO 4
    b(i%,j%)=SUCC(j%)
  NEXT j%
NEXT i%
'
MAT PRINT a()
PRINT "-------------"
MAT PRINT b()
PRINT "-------------"
MAT CPY a(2,2)=b(2,2)
MAT PRINT a()

      Renvoie 1,1,1
              1,1,1
              1,1,1
              1,1,1
              1,1,1
              -------------
              2,3,4,5
              2,3,4,5
```

```
            2,3,4,5
            2,3,4,5
            -------------
            1,1,1
            1,3,4
            1,3,4
            1,3,4
            1,1,1
```

MAT CPY a()=b(),h,w copie dans la matrice a, h lignes et w colonnes de la matrice b, les éléments d'indices équivalents s'il y a dépassement. Ne pas oublier le rôle de MAT BASE.

➥ Abréviation : m c a()=b(),h,w

Exemple

```
DIM a(5,3),b(4,4)
MAT SET a()=1
FOR i%=1 TO 4
  FOR j%=1 TO 4
    b(i%,j%)=SUCC(j%)
  NEXT j%
NEXT i%
'
MAT PRINT a()
PRINT "-------------"
MAT PRINT b()
PRINT "-------------"
MAT CPY a()=b()
MAT PRINT a()
```

```
      Renvoie 1,1,1
              1,1,1
              1,1,1
              1,1,1
              1,1,1
              -------------
              2,3,4,5
              2,3,4,5
              2,3,4,5
              2,3,4,5
              -------------
              2,3,4
              2,3,4
```



```
            2,3,4
            2,3,4
            1,1,1
```

MAT XCPY a([i,j])=b([k,l]),[h,w]

fonctionne selon le même principe que MAT CPY a([i,j])=b([k,l]),[h,w], à cela près qu'une transposée de la matrice b est faite pour l'opération, c'est à dire que les lignes et colonnes de b deviennent respectivement les colonnes et les lignes de la matrice servant au calcul. La matrice b reste inchangée aprés l'exécution de la fonction.

➥ Abréviation : m x a([i,j])=b([k,l]),[h,w]

Exemple

```
DIM a(5,3),b(4,4)
MAT SET a()=1
FOR i%=1 TO 4
  FOR j%=1 TO 4
    b(i%,j%)=SUCC(j%)
  NEXT j%
NEXT i%
'
MAT PRINT a()
PRINT "-------------"
MAT PRINT b()
PRINT "-------------"
MAT XCPY a(2,2)=b(2,2),3,3
MAT PRINT a()
```

```
      Renvoie 1,1,1
              1,1,1
              1,1,1
              1,1,1
              1,1,1
              -------------
              2,3,4,5
              2,3,4,5
              2,3,4,5
```

```
2,3,4,5
------------
1,1,1
1,3,3
1,4,4
1,5,5
1,1,1
```

Cas particuliers

Semblables à MAT CPY a(i,j)=b(k,l),w,h

Pour MAT CPY comme pour MAT XCPY, i, j et h sont ignorés dans le cas d'utilisation de vecteurs. Après un DIM a(n),b(m), a et b seront définis comme vecteurs ligne, c'est à dire des matrices de types (1,n) et (1,m). Pour travailler avec des vecteurs colonne, il vous faut des matrices de type (n,1) et (m,1), donc faire un DIM a(n,1),b(m,1). Si les deux vecteurs sont de même ordre (vecteurs ligne ou colonne), vous devez utiliser MAT CPY. Indépendamment du type des vecteurs a et b, MAT CPY considère les deux comme des vecteurs colonne. La syntaxe correcte est MAT CPY a(n,1)=b(m,1).

Exemple

```
DIM a(10),b(5)       ' a et b sont des vecteurs ligne
MAT SET a()=1
FOR i%=1 TO 5
  b(i%)=SUCC(i%)
NEXT i%
PRINT "a():   ";
MAT PRINT a()
PRINT "b():   ";
MAT PRINT b()
PRINT STRING$(45,"-")
MAT CPY a(3,1)=b(1,1) !considère a et b comme vecteurs
                      ! colonne
PRINT "MAT CPY a(3,1)=b(1,1):  ";
MAT PRINT a()
```

```
      Renvoie a():  1,1,1,1,1,1,1,1,1,1
             b():  2,3,4,5,6
                   -------------------------------------------
                   MAT CPY a(3,1)=b(1,1):  1,1,2,3,4,5,6,1,1,1
```


L'utilisation de MAT XCPY avec des vecteurs nécessite un vecteur ligne et un vecteur colonne. Le deuxième argument de la fonction est transposé (pour le calcul seulement, il n'est pas modifié) avant d'être copié dans le premier. MAT XCPY avec des vecteurs fonctionnera avec les dimensionnements suivants :

- DIM a(1,n),b(m,1) : a()=Vecteur ligne, b()=Vecteur colonne
- DIM a(n,1),b(1,m) : a()=Vecteur colonne, b()=Vecteur ligne

Exemple

```
DIM a(1,10),b(5,1)
MAT SET a()=1
FOR i%=1 TO 5
  b(i%,1)=SUCC(i%)
NEXT i%
MAT PRINT a()
PRINT
MAT PRINT b()
MAT XCPY a(1,3)=b(1,1)
PRINT
MAT PRINT a()
```

```
      Renvoie     1,1,1,1,1,1,1,1,1,1

                  2
                  3
                  4
                  5
                  6

                  1,1,2,3,4,5,6,1,1,1
```

On peut utiliser des paramètres optionnels h et w lors d'une copie de vecteurs avec MAT CPY et MAT XCPY mais :

- **Pour MAT CPY**
  Si w >= 1, h seul est pris en compte
  Si w = 0, pas de copie

- **Pour MAT XCOPY**
  Si w >= 1, seul h est pris en compte, si b est un vecteur colonne qui est copié après la transposition dans un vecteur ligne

Si w = 0 , pas de copie
Si h >= 1, seul w est pris en compte, si b est un vecteur ligne qui est copié après la transposition dans un vecteur colonne
Si h = 0, pas de copie

MAT TRANS a()=b() copie dans a, la transposée de la matrice b avec le nombre de lignes de a égal au nombre de colonnes de b et le nombre de colonnes de a égal au nombre de lignes de b. La dimension des matrices est du type DIM a(n,m),b(m,n).

Exemple

```
DIM a(3,4),b(4,3)
MAT SET b()=4
MAT SET a()=1
MAT PRINT a()
PRINT STRING$(10,"-")
MAT PRINT b()
PRINT STRING$(10,"-")
MAT TRANS a()=b()
MAT PRINT a()
```

```
      Renvoie     1,1,1,1
                  1,1,1,1
                  1,1,1,1
                  ----------
                  4,4,4
                  4,4,4
                  4,4,4
                  4,4,4
                  ----------
                  4,4,4,4
                  4,4,4,4
                  4,4,4,4
```

Si la matrice est carrée (même nombre de lignes et de colonnes), on peut utiliser MAT TRANS a(). Cette instruction intervertit les lignes et les colonnes de la matrice a. On peut récupérer le contenu initial en faisant un nouveau MAT TRANS a().

➥ Abréviation : m t a()

Exemple

```
DIM a(5,5)
FOR i%=1 TO 5
  FOR j%=1 TO 5
    a(i%,j%)=j%
  NEXT j%
NEXT i%
MAT PRINT a()
PRINT STRING$(10,"-")
MAT TRANS a()
MAT PRINT a()
```

```
      Renvoie     1,2,3,4,5
                  1,2,3,4,5
                  1,2,3,4,5
                  1,2,3,4,5
                  1,2,3,4,5
                  ----------
                  1,1,1,1,1
                  2,2,2,2,2
                  3,3,3,3,3
                  4,4,4,4,4
                  5,5,5,5,5
```

2.5. Instructions de calcul

MAT ADD a()=b()+c()
MAT ADD a(),b()
MAT ADD a(),x

MAT SUB a()=b()-c()
MAT SUB a(),b()
MAT SUB a(),x

MAT MUL a()=b()*c()
MAT MUL x=a()*b()
MAT MUL x=a()*b()*c()
MAT MUL a(),x

MAT NORM a(),0
MAT NORM a(),1

MAT DET x=a([i,j])[,n]
MAT QDET x=a([i,j])[,n]
MAT RANG x=a([i,j])[,n]
MAT INV a()=b()

a, b, c : nom des tableaux de valeurs numériques
x : expression numérique, scalaire
i, j, n : expression numérique

MAT ADD a()=b()+c() est seulement définie pour des matrices du même ordre, par exemple de dimensions DIM a(n,m),b(n,m),c(n,m) ou DIM a(n),b(n),c(n). Le résultat dans a est la somme des deux éléments de mêmes indices de b et c.

➥ Abréviation : m a()=b()+c()

Exemple

```
DIM a(3,5),b(3,5),c(3,5)
MAT SET b()=3
MAT SET c()=4
MAT PRINT b()
PRINT STRING$(10,"-")
MAT PRINT c()
PRINT STRING$(10,"-")
MAT ADD a()=b()+c()
MAT PRINT a()
```

```
      Renvoie     3,3,3,3,3
                  3,3,3,3,3
                  3,3,3,3,3
                  ----------
                  4,4,4,4,4
                  4,4,4,4,4
                  4,4,4,4,4
                  ----------
```



```
                  7,7,7,7,7
                  7,7,7,7,7
                  7,7,7,7,7
```

MAT ADD a(),b() équivaut à MAT ADD a()=a()+b().

➥ Abréviation : m a a(),b()

Exemple

```
DIM a(3,5),b(3,5)
MAT SET a()=1
MAT SET b()=3
MAT PRINT a()
PRINT STRING$(10,"-")
MAT PRINT b()
PRINT STRING$(10,"-")
MAT ADD a(),b()
MAT PRINT a()
```

```
      Renvoie     1,1,1,1,1
                  1,1,1,1,1
                  1,1,1,1,1
                  ----------
                  3,3,3,3,3
                  3,3,3,3,3
                  3,3,3,3,3
                  ----------
                  4,4,4,4,4
                  4,4,4,4,4
                  4,4,4,4,4
```

MAT ADD a(),x est définie pour toutes matrices et vecteurs. Le scalaire x est ajouté élément par élément à ceux de a. Le résultat est disponible dans a.

➥ Abréviation : m a a(),x

Exemple

```
DIM a(3,5)
MAT SET a()=1
MAT PRINT a()
PRINT STRING$(10,"-")
```

```
MAT ADD a(),5
MAT PRINT a()

      Renvoie     1,1,1,1,1
                  1,1,1,1,1
                  1,1,1,1,1
                  ----------
                  6,6,6,6,6
                  6,6,6,6,6
                  6,6,6,6,6
```

MAT SUB a()=b()-c() est seulement définie pour des matrices du même ordre, par exemple de dimensions DIM a(n,m),b(n,m),c(n,m) ou DIM a(n),b(n),c(n). Le résultat dans a est la différence des deux éléments de mêmes indices de b et c.

➥ Abréviation : m a()=b()-c()

Exemple

```
DIM a(3,5),b(3,5),c(3,5)
MAT SET b()=5
MAT SET c()=3
MAT PRINT b()
PRINT STRING$(10," ")
MAT PRINT c()
PRINT STRING$(10,"-")
MAT SUB a()=b()-c()
MAT PRINT a()

      Renvoie     5,5,5,5,5
                  5,5,5,5,5
                  5,5,5,5,5
                  ----------
                  3,3,3,3,3
                  3,3,3,3,3
                  3,3,3,3,3
                  ----------
                  2,2,2,2,2
                  2,2,2,2,2
                  2,2,2,2,2
```


MAT SUB a(),b() équivaut à MAT SUB a(),b().

➥ Abréviation : m a a(),b()

Exemple

```
DIM a(3,5),b(3,5)
MAT SET a()=3
MAT SET b()=1
MAT PRINT a()
PRINT STRING$(10,"-")
MAT PRINT b()
PRINT STRING$(10,"-")
MAT SUB a(),b()
MAT PRINT a()

      Renvoie     3,3,3,3,3
                  3,3,3,3,3
                  3,3,3,3,3
                  ----------
                  1,1,1,1,1
                  1,1,1,1,1
                  1,1,1,1,1
                  ----------
                  2,2,2,2,2
                  2,2,2,2,2
                  2,2,2,2,2
```

MAT SUB a(),x est définie pour toutes matrices et vecteurs. Le scalaire x est soustrait élément par élément à ceux de a. Le résultat est disponible dans a.

➥ Abréviation : m s a(),x

Exemple

```
DIM a(3,5)
MAT SET a()=6
MAT PRINT a()
PRINT STRING$(10,"-")
MAT SUB a(),5
MAT PRINT a()

      Renvoie     6,6,6,6,6
```

```
6,6,6,6,6
6,6,6,6,6
----------
5,5,5,5,5
5,5,5,5,5
5,5,5,5,5
----------
1,1,1,1,1
1,1,1,1,1
1,1,1,1,1
```

MAT MUL a()=b()*c() met dans a le produit des matrices b et c. Pour que le résultat soit défini, il faut que le nombre de lignes de b soit égal au nombre de colonnes de c et que le nombre de colonnes de b soit égal au nombre de lignes de c. La matrice résultat a doit avoir un nombre de lignes équivalent à celui de b et le même nombre de colonnes que c, par exemple comme le dimensionnement DIM a(2,2),b(2,3),c(3,2). On obtient le calcul d'un élément de position i,j par a(i,j)=SOMME(n=0;n=x)[b(i,x)*c(x,j)].

➥ Abréviation : m a()=b()*c()

Exemple

```
DIM a(2,2),b(2,3),c(3,2)
MAT SET b()=1
DATA 1,2,-3,4,5,-1
MAT READ c()
MAT PRINT b(),5,1
PRINT STRING$(18,"-")
MAT PRINT c(),5,1
PRINT STRING$(18,"-")
MAT MUL a()=b()*c()
MAT PRINT a(),5,1
```

```
Renvoie     1.0,  1.0,  1.0
            1.0,  1.0,  1.0
            ---------------
             1.0,  2.0
            -3.0,  4.0
```



```
            5.0, -1.0
            ---------------
            3.0,  5.0
            3.0,  5.0
```

Si, au lieu de matrices, on a des vecteurs, alors on obtient un produit matriciel de vecteurs.

Exemple

```
DIM a(3,3),b(3),c(3)
DATA 1,2,-3,4,5,-1
MAT READ b()
MAT READ c()
MAT PRINT b(),5,1
PRINT STRING$(18,"-")
MAT PRINT c(),5,1
PRINT STRING$(18,"-")
MAT MUL a()=b()*c()
MAT PRINT a(),5,1
```

```
Renvoie     1.0,  2.0, -3.0
            ----------------
             4.0,  5.0, -1.0
            ----------------
             4.0,  5.0, -1.0
             8.0, 10.0, -2.0
           -12.0,-15.0,  3.0
```

MAT MUL x=a()*b() est défini avec b et c vecteurs. Le résultat du produit scalaire de a et de b est un scalaire x. On le trouve en faisant le calcul x=SOMME((n=0;n=i)[a(n)*b(n)].

➥ Abréviation : m x=a()*b()

Exemple

```
DIM b(3),c(3)
DATA 1,2,-3,4,5,-1
MAT READ b()
MAT READ c()
MAT PRINT b(),5,1
PRINT STRING$(18,"-")
```

```
MAT PRINT c(),5,1
PRINT STRING$(18,"-")
MAT MUL x=b()*c()
PRINT x
```

```
   Renvoie     1.0,  2.0, -3.0
              ------------------
                4.0,  5.0, -1.0
              ------------------
               17.0
```

MAT MUL x=a()*b()*c()

est défini pour des vecteurs a, c et une matrice b particuliers. Le résultat est un scalaire x qui se calcule en multipliant le vecteur a à la matrice b puis ce produit au vecteur c. Le vecteur a doit avoir le même nombre d'éléments que la matrice b de lignes et le vecteur b le même nombre d'éléments que la matrice b de colonnes, comme le dimensionnement DIM a(5),b(5,3),c(3).

➥ Abréviation : m x=a()*b()*c()

Exemple

```
DIM a(2),b(2,3),c(3)
DATA 1,2,-3,4,5
MAT READ a()
MAT READ c()
MAT SET b()=1
MAT PRINT a(),5,1
PRINT STRING$(18,"-")
MAT PRINT b(),5,1
PRINT STRING$(18,"-")
MAT PRINT c(),5,1
PRINT STRING$(18,"-")
MAT MUL x=a()*b()*c()
PRINT x
```

```
      Renvoie     1.0,  2.0
                 ------------------
                 1.0,  1.0,  1.0
```



```
                 1.0,  1.0,  1.0
                 1.0,  1.0,  1.0
                ------------------
                -3.0,  4.0,  5.0
                ------------------
                18.0
```

MAT NORM a(),0 ou MAT NORM a(),1

sont définies pour des matrices ou des vecteurs. MAT NORM a(),0 normalise une matrice ou un vecteur par ligne, MAT NORM a(),1 le fait par colonne. Une matrice normalisée par ligne (par colonne) doit avoir la somme des carrés des éléments d'une ligne (d'une colonne) égale à 1.

➥ Abréviation : m no a(),0 ou m no a(),1

Exemple

```
DIM a(10,10),b(10,10),v(10)
DATA 1,2,3,4,5,6,7,8,9,-1
DATA 3.2,4,-5,2.4,5.1,6.2,7.2,8.1,6,-5
DATA -2,-5,-6,-1.2,-1.5,-6.7,4.5,8.1,3.4,10
DATA 5,-2.3,4,5.6,12.2,18.2,14.1,16,-21,-13
DATA 4.1,5.2,16.7,18.4,19.1,20.2,13.6,14.8,19.4,18.6
DATA 15.2,-1.8,13.6,-4.9,5.4,19.8,16.4,-20.9,21.4,13.8
DATA -3.6,6,-8.2,-9.1,4,-2.5,2.3,4.6,7,8.4
DATA 4.7,8.3,9.4,10.5,11,19,15.4,18.9,-20,12.6
DATA 5.3,-4.7,6.1,6.5,6.9,-9.2,-10.8,4.3,5.6,9.1
DATA 21.4,19.5,28.4,19.3,24.6,14.9,71.3,23.5,14.5,-12.3
'
CLS
MAT READ a()
MAT CPY b()=a()          ! Sauvegarder matrice originale
PRINT "Matrice originale"
PRINT
MAT PRINT a(),7,2
INP(2)
'
' Normalisation par ligne
'
CLS
MAT NORM a(),0
```

```
PRINT
PRINT "Normaliser par ligne : "
PRINT

MAT PRINT a(),7,2
INP(2)
'
' Tester la normalisation
'
PRINT
PRINT "Vérification : "
PRINT
FOR i%=1 TO 10
  MAT XCPY v()=a(i%,1)  ! Copier les lignes dans des vecteurs
  MAT MUL x=v()*v()     ! Calcule le produit scalaire entre v
                                                    ! et v
  PRINT x'
NEXT i%
PRINT
INP(2)
'
' Normaliser par colonne
'
CLS
MAT CPY a()=b()           ! Restituer matrice originale
MAT NORM a(),1
PRINT "Normalise par colonne : "
PRINT
MAT PRINT a(),7,2
INP(2)
'
' Tester la normalisation
'
PRINT
PRINT "Vérification : "
PRINT
FOR i%=1 TO 10
  MAT CPY v()=a(1,i%)  ! Copier les colonnes dans des vecteurs
  MAT MUL x=v()*v()    ! Calcule le produit scalaire entre v
                                                    ! et v
  PRINT x'
NEXT i%
INP(2)
```

```
      Renvoie

      Matrice originale

   1.00,   2.00,   3.00,   4.00,   5.00,   6.00,   7.00,
8.00,   9.00,
  -1.00
   3.20,   4.00,  -5.00,   2.40,   5.10,   6.20,   7.20,
```



```
8.10,   6.00,
  -5.00
  -2.00,  -5.00,  -6.00,  -1.20,  -1.50,  -6.70,   4.50,
8.10,   3.40,
  10.00
   5.00,  -2.30,   4.00,   5.60,  12.20,  18.20,  14.10,
16.00, -21.00,
 -13.00
   4.10,   5.20,  16.70,  18.40,  19.10,  20.20,  13.60,
14.80,  19.40,
  18.60
  15.20,  -1.80,  13.60,  -4.90,   5.40,  19.80,  16.40,
-20.90,  21.40,
  13.80
  -3.60,   6.00,  -8.20,  -9.10,   4.00,  -2.50,   2.00,
3.40,   6.70,
   8.40
   4.70,   8.30,   9.40,  10.50,  11.00,  19.00,  15.40,
18.90, -20.00,
  12.60
   5.30,  -4.70,   6.10,   6.50,   6.90,  -9.20, -10.80,
4.30,   5.60,
   9.10
  21.40,  19.50,  28.40,  19.30,  24.60,  14.90,  71.30,
23.50,  14.50,
 -12.30

Normalisation par ligne

   0.06,   0.12,   0.18,   0.24,   0.30,   0.35,   0.41,
0.47,   0.53,
  -0.06
   0.18,   0.23,  -0.29,   0.14,   0.29,   0.36,   0.42,
0.47,   0.35,
  -0.29
  -0.11,  -0.28,  -0.34,  -0.07,  -0.09,  -0.38,   0.26,
0.46,   0.19,
   0.57
   0.12,  -0.06,   0.10,   0.14,   0.30,   0.45,   0.35,
0.40,  -0.52,
  -0.32
   0.08,   0.10,   0.33,   0.36,   0.38,   0.40,   0.27,
0.29,   0.38,
   0.37
   0.32,  -0.04,   0.29,  -0.10,   0.11,   0.42,   0.35,
-0.44,   0.45,
   0.29
  -0.19,   0.32,  -0.44,  -0.48,   0.21,  -0.13,   0.11,
0.18,   0.36,
   0.45
   0.11,   0.19,   0.21,   0.24,   0.25,   0.43,   0.35,
0.43,  -0.46,
```

```
   0.29
   0.23,  -0.21,   0.27,   0.29,   0.31,  -0.41,  -0.48,
0.19,   0.25,
   0.40
   0.23,   0.21,   0.30,   0.21,   0.26,   0.16,   0.76,
0.25,   0.15,
  -0.13

Vérification :

1 1 1 1 1 1 1 1 1 1

Normalisation par colonne :

   0.04,   0.08,   0.08,   0.12,   0.13,   0.14,   0.09,
0.18,   0.20,
  -0.03
   0.11,   0.16,  -0.13,   0.07,   0.14,   0.14,   0.09,
0.18,   0.13,
  -0.14
  -0.07,  -0.21,  -0.15,  -0.04,  -0.04,  -0.15,   0.06,
0.18,   0.07,
   0.28
   0.18,  -0.09,   0.10,   0.17,   0.33,   0.41,   0.18,
0.35,  -0.46,
  -0.36
   0.14,   0.21,   0.42,   0.57,   0.51,   0.46,   0.17,
0.33,   0.42,
   0.52
   0.53,  -0.07,   0.35,  -0.15,   0.15,   0.45,   0.21,
-0.46,   0.47,
   0.38
  -0.13,   0.25,  -0.21,  -0.28,   0.11,  -0.06,   0.03,
0.08,   0.15,
   0.23
   0.17,   0.34,   0.24,   0.33,   0.30,   0.43,   0.20,
0.42,  -0.44,
   0.35
   0.19,  -0.19,   0.15,   0.20,   0.19,  -0.21,  -0.14,
0.10,   0.12,
   0.25
   0.75,   0.80,   0.72,   0.60,   0.66,   0.34,   0.90,
0.52,   0.32,
  -0.34
vérification :

1 1 1 1 1 1 1 1 1 1
```


MAT DET x=a([i,j])[,n]

calcule le déterminant d'une matrice carrée de type (n,n). Les indices commencent à 0 ou 1 en fonction de MAT BASE. Il est aussi possible de calculer le déterminant d'une partie carrée d'une matrice quelconque. Pour cela, indiquer pour i et j le début de la partie de matrice dont vous voulez calculer le déterminant et pour n le nombre de lignes et colonnes. Le calcul se fera de façon interne sans affecter la matrice originale.

➥ Abréviation : m d x=a([i,j])[,n]

Exemple

```
DIM a(10,10),b(4,4)
DATA 1,2,3,4,5,6,7,8,9,-1
DATA 3,2,4,-5,2,4,5,1,6,2,7,2,8,1,6,-5
DATA -2,-5,-6,-1.2,-1.5,-6.7,4.5,8.1,3.4,10
DATA 5,-2.3,4.5,6,12.2,18.2,14.1,16,-21,-13,3.8
DATA 4.1,5.2,16.7,18.4,19.1,20.2,13.6,14.8,19.4,18.6
DATA 15.2,-1.8,13.6,-4.9,5.4,19.8,16.4,-20.9,21.4,13.8
DATA -3.6,6,-8.2,-9.1.4,-2.5,2,3.4,6.7,8.4,10.9
DATA 4.7,8.3,9.4,10.5,11,19,15.4,18.9,-20,12.6
DATA 5.3,-4.7,6.1,6.5,6.9,-9.2,-10.8,4.3,5.6,9.1
DATA 21.4,19.5,28.4,19.3,24.6,14.9,71.3,23.5,14.5,-12.3
'
CLS
MAT READ a()
PRINT "Matrice originale"
PRINT
MAT PRINT a(),7,2
PRINT
PRINT "Déterminant : ";
MAT DET x=a()           ! Calculer le déterminant de a
PRINT x;
MAT DET y=a(1,4),4      ! Calculer le déterminant d'une partie
                            ! de la matrice
PRINT
PRINT "Déterminant de a(1,4),4 : ";
PRINT y
PRINT
```

```
PRINT "Vérification :"
PRINT
MAT CPY b()=a(1,4),4,4 ! Copie la partie de matrice dans b
MAT PRINT b(),7,2
MAT DET z=b()          ! Calculer le déterminant de b
PRINT
PRINT z

      Renvoie

Matrice originale

   1.00,   2.00,   3.00,   4.00,   5.00,   6.00,   7.00,
8.00,   9.00,
  -1.00
   3.20,   4.00,  -5.00,   2.40,   5.10,   6.20,   7.20,
8.10,   6.00,
  -5.00
  -2.00,  -5.00,  -6.00,  -1.20,  -1.50,  -6.70,   4.50,
8.10,   3.40,
  10.00
   5.00,  -2.30,   4.00,   5.60,  12.20,  18.20,  14.10,
16.00, -21.00,
 -13.00
   3.80,   4.10,   5.20,  16.70,  18.40,  19.10,  20.20,
13.60,  14.80,
  19.40
  18.60,  15.20,  -1.80,  13.60,  -4.90,   5.40,  19.80,
16.40, -20.90,
  21.40
  13.80,  -3.60,   6.00,  -8.20,  -9.10,   4.00,  -2.50,
2.00,   3.40,
   6.70
   8.40,  10.90,   4.70,   8.30,   9.40,  10.50,  11.00,
19.00,  15.40,
  18.90
 -20.00,  12.60,   5.30,   4.70,   6.10,   6.50,   6.90,
-9.20, -10.80,
   4.30
   5.60,   9.10,  21.40,  19.50,  28.40,  19.30,  24.60,
14.90,  71.30,
  23.50

Déterminant : -2549840202186
Déterminante de a(1,4),4 : -57.61200000001

Vérification :

   4.00,   5.00,   6.00,   7.00
   2.40,   5.10,   6.20,   7.20
  -1.20,  -1.50,  -6.70,   4.50
   5.60,  12.20,  18.20,  14.10
```



```
-57.612D0000001
```

MAT QDET x=a([i,j])[,n]

fonctionne de la même manière que MAT DET mais est optimisée pour la vitesse. En général, les deux procédé donnent le même résultat mais pour des matrices dont le déterminant est proche de 0, il vaut mieux utiliser la fonction MAT DET.

➥ Abréviation : m qd x=a([i,j])[,n].

Exemple

```
DIM a(10,10)
DATA 1,2,3,4,5,6,7,8,9,-1
DATA 3.2,4,-5,2.4,5.1,6.2,7.2,8.1,6,-5
DATA -2,-5,-6,-1.2,-1.5, 6.7,4.5,8.1,3.4,10
DATA 5,-2.3,4.5,6.12.2,18.2,14.1,16,-21,-13,3.8
DATA 4.1,5.2,16.7,18.4,19.1,20.2,13.6,14.8,19.4,18.6
DATA 15.2,-1.8,13.6,-4.9,5.4,19.8,16.4,-20.9,21.4,13.8
DATA -3.6,6,-8.2,-9.1,4,-2.5,2.3,4,6.7,8.4,10.9
DATA 4.7,8.3,9.4,10.5,11,19.15.4,18.9,-20,12.6
DATA 5.3,-4.7,6.1,6.5,6.9,-9.2,-10.8,4.3,5.6,9.1
DATA 21.4,19.5,28.4,19.3,24.6,14.9,71.3,23.5,14.5,-12.3
'
CLS
MAT READ a()
PRINT "Matrice originale"
PRINT
MAT PRINT a(),7,2
PRINT
PRINT "Déterminant avec MAT DET : ";
MAT DET x=a()
PRINT x;
PRINT
PRINT "Déterminant avec MAT QDET : ";
MAT DET y=a()
PRINT y;
PRINT
PRINT "Ecart : ";x-y

      Renvoie

Matrice originale
```

```
   1.00,   2.00,   3.00,   4.00,   5.00,   6.00,   7.00,
8.00,   9.00,
  -1.00
   3.20,   4.00,  -5.00,   2.40,   5.10,   6.20,   7.20,
8.10,   6.00,
  -5.00
  -2.00,  -6.00,  -6.00,  -1.20,  -1.60,  -6.70,   4.50,
8.10,   3.40,
  10.00
   5.00,  -2.30,   4.00,   5.60,  12.20,  18.20,  14.10,
16.00, -21.00,
 -13.00
   3.80,   4.10,   5.20,  16.70,  18.40,  19.10,  20.20,
13.60,  14.80,
  19.40
  18.60,  15.20,  -1.80,  13.60,  -4.90,   5.40,  19.80,
16.40, -20.90,
  21.40
  13.80,  -3.60,   6.00,  -8.20,  -9.10,   4.00,  -2.50,
2.00,   3.40,
   6.70
   8.40,  10.90,   4.70,   8.30,   9.40,  10.50,  11.00,
19.00,  15.40,
  18.90
 -20.00,  12.60,   5.30,  -4.70,   6.10,   6.50,   6.90,
-9.20, -10.80,
   4.30
   5.60,   9.10,  21.40,  19.50,  28.40,  19.30,  24.60,
14.90,  71.30,
  23.50

Déterminant avec MAT DET : -2549840202186
Déterminant avec MAT QDET : -2549840202186
Ecart : 0
```

MAT RANG x=a([i,j])[,n] donne le rang d'une matrice carrée. On peut, de la même manière que pour MAT QDET, calculer le rang d'une partie de matrice.

➥ Abréviation : m ra x=a([i,j])[,n].

Exemple

```
DIM a(5,5)
DATA 1,2,3,4,5
```



```
DATA 3.2,4,-5,2.4,5.1
DATA -2,4,-5,2.4,5.1
DATA 5,-2.3,4,5.6,12.2
DATA 4.1,5.2,16.7,18.4,19.1
'
CLS
MAT READ a()
PRINT "Matrice originale"
PRINT
MAT PRINT a(),7,2
PRINT
PRINT "Rang de a: ";
MAT RANG x=a()
PRINT x;
PRINT
PRINT "Rang de a(1,2),3 : ";
MAT RANG y=a(1,2),3
PRINT y;
PRINT

      Renvoie

Matrice originale

   1.00,   2.00,   3.00,   4.00,   5.00
   3.20,   4.00,  -5.00,   2.40,   5.10
  -2.00,   4.00,  -5.00,   2.40,   5.10
   5.00,  -2.30,   4.00,   5.60,  12.20
   4.10,   5.20,  16.70,  18.40,  19.10

Rang de a: 5
Rang de a(1,2),3 : 2
```

MAT INV b()=a() calcule l'inverse d'une matrice carrée. L'inverse de a est placé dans b. La matrice b doit être du même type que a.

➥ Abréviation : m inv b()=a().

Exemple

```
DIM a(5,5),b(5,5),c(5,5)
DATA 1,2,3,4,5
DATA 3.2,4,-5,2.4,5.1
DATA -2,4,-5,2.4,5.1
DATA 5,-2.3,4,5.6,12.2
DATA 4.1,5.2,16.7,18.4,19.1
'
```

```
CLS
MAT READ a()
PRINT "Matrice originale a() : "
PRINT
MAT PRINT a(),7,2
'
MAT INV b()=a()
PRINT
PRINT "Inverse de a() : "
PRINT
MAT PRINT b(),7,2
PRINT
PRINT "Vérification b()*a() = matrice unité ? "
PRINT
MAT MUL c()=b()*a()
MAT PRINT c(),7,2
      Renvoie

Matrice originale a() :

   1.00,    2.00,    3.00,    4.00,    5.00
   3.20,    4.00,   -5.00,    2.40,    5.10
  -2.00,    4.00,   -5.00,    2.40,    5.10
   5.00,   -2.30,    4.00,    5.60,   12.20
   4.10,    5.20,   16.70,   18.40,   19.10

Inverse de a() :

   0.00,    0.19,   -0.19,   -0.00,   -0.00
   0.97,    0.02,   -0.09,   -0.10,   -0.17
   0.71,   -0.10,   -0.10,   -0.01,   -0.12
  -1.65,    0.17,    0.11,   -0.06,    0.39
   0.71,   -0.12,    0.04,    0.09,   -0.17

Vérification b()*a() = matrice unité ?

   1.00,    0.00,    0.00,    0.00,    0.00
   0.00,    1.00,    0.00,    0.00,   -0.00
   0.00,   -0.00,    1.00,    0.00,   -0.00
  -0.00,   -0.00,   -0.00,    1.00,    0.00
  -0.00,    0.00,    0.00,    0.00,    1.00
```


3. Autres fonctions de la version 3.5

Voici, présentées ici, les autres instructions ne traitant pas du calcul matriciel et vectoriel.

3.1. Combinatoire

x=FACT(n)
y=VARIAT(n,k)
z=COMBIN(n,k)

x, y, z: variables
n, k: expressions entières

x=FACT(n) place dans x la factorielle d'un nombre entier n. C'est le produit des n premiers nombres entiers naturels, hormis 0 (0!=1).

y=VARIAT(n,k) met dans y le nombre possible de variations de n éléments de la kième classe sans répétition.

➥ VARIAT(n,k)=FACT(n)/FACT(n-k)

z=COMBIN(n,k) calcule dans z le coefficient binomial de (n,k). Par exemple calculer le nombre de possibilités différentes de construire un sous-ensemble de k éléments d'un ensemble de n éléments.

➥ COMBIN(n,k)=FACT(n)/(FACT(k)*FACT(n-k))

3.2. Pointeurs sur DATA

_DATA
_DATA=

_DATA renvoie un pointeur sur les zones de DATA. Si _DATA est à 0 et que l'on tente un READ, on reçoit le message d'erreur "out of data".

_DATA= fixe le pointeur de la zone de DATA sur une valeur déterminée précedemment avec _DATA.

```
DIM dp%(100)
DATA 1,2,3,4,5,6,7,8,9
DATA 13,24,328,3242,1,0
'
i%=0
DO WHILE _DATA
      dp%(i%)=_DAT

      INC i%
      READ a
LOOP
'
DEC i%
'
FOR j%=i% DOWNTO 0
      _DATA=dp%(j%)
      READ a
      PRINT a'
NEXT j%
INP(2)
```

Renvoie 0 1 3242 328 24 13 9 8 7 6 5 4 3 2 1

4. Les fonctions spécifiques à l'Atari STE

Ce chapitre décrit les fonctions incluses dans GFA Basic 3.5E qui sont destinées à exploiter pleinement les particularités de l'Atari STE.

Ces fonctions ne renvoient aucun effet ou génèrent une erreur lorsqu'elles sont utilisées sur une machine Atari STF ou Mega ST anciennes ROM. Par conséquent, si vous développez un programme en GFA Basic 3.5E, vous devez tenir compte de ces informations. Si votre programme doit aussi être utilisé sur les anciens Atari ST, vous devez soit éviter de faire appel à ces fonctions soit les utiliser en détectant préalablement la présence d'un STE (fonction STE?).

DMACONTROL ctrl

ctrl = 0 désactive le son DMA
ctrl = 1 active le son
ctrl = 3 répète continuellement le son

Si cette commande est utilisée sur Atari STF, le message d'erreur "Opération privilégiée à STE" est générée.

DMASOUND beg, end, rate [,ctrl]

Lance le son DMA

beg = adresse de début de l'échantillon
end = adresse de fin de l'échantillon
rate = fréquence d'exécution (0 = 6,25 KHz, 1 = 12,5 KHz, 2 = 25 KHz, 3 = 50 KHz)
ctrl = 0 pour désactiver le son DMA, ctrl = 1 pour activer le son, ctrl = 3 pour répéter le son

Si cette commande est utilisée sur Atari STF, le message d'erreur "Opération privilégiée à STE" est générée.

LPENX et LPENY

var = LPENX var = LPENY

Renvoient les coordonnées X et Y du pointeur d'un crayon optique éventuellement connecté.

Si cette commande est utilisée sur Atari STF, le message d'erreur "Opération privilégiée à STE" est générée.

MW_OUT Masque,Datas

Gère l'interface interne MICROWIRE du STE. L'utilisation générale est la gestion sonore.

MW_OUT &H7FF,x

```
x = &X10 011 ddd ddd    Règle le volume maître
             000 000    -80 dB
             010 100    -40 dB
             101 xxx      0 dB
```

Les 5 derniers bits doublent la puissance du son en dB.

```
x = &X10 101 xdd ddd    Règle le volume du canal gauche
x = &X10 010 xdd ddd    Règle le volume du canal droit
              00 000    -40 dB
              01 010    -20 dB
              10 1xx      0 dB
```

Les 4 derniers bits doublent la puissance du son en dB.

```
x = &X10 010 xxd ddd    Règle l'aigu
x = &X10 001 xxd ddd    Règle le grave
              0 000    -12 dB
              0 110      0 dB
              1 100    +12 dB
```



```
x = &X10 000 xxx xdd    Règle le mixage
                 00     -12 dB
                 01     Son normal ST
                 10     Pas de mixage
                 11     Réservé
```

Exemple : MW_OUT &H7FF, &X10000000010 désactive le son ST.

Attention : Cette fonction ne doit être utilisée que sur STE.

PADX, PADY et PADT

PADX(i) renvoie la position X d'un des deux paddles (i=0 ou 1)
PADY(i) renvoie la position Y d'un des deux paddles (i=0 ou 1)
PADT(i) renseigne sur le bouton Feu (i=0 ou 1).

STE?

Cette fonction renvoie -1 si l'ordinateur est un Atari STE (ou un TT), 0 sinon.

STICK(i) et STRIG(i)

STICK permet de tester la position du joystick (i=0 à 5).
STRIG permet de tester l'état du bouton de feu.
STICK(0) renvoie la position du joystick sur le port 0.
STICK(1) renvoie la position du joystick sur le port 1.

Ces considérations sont valables sur tous les modèles d'Atari ST. Reportez-vous à la commande STICK dans le manuel de référence de l'interpréteur page 138.

L'Atari STE dispose de ports joystick supplémentaires. STICK(2) à STICK(5) permettent de tester la position du joystick sur l'un des ports supplémentaires.

TT?

Cette fonction renvoie -1 si l'ordinateur est équipé d'un processeur 68020 ou 68030, 0 sinon.

5. Le compilateur GFA BASIC 3.5E

Ce chapitre décrit exclusivement les nouvelles fonctionnalités du compilateur GFA BASIC 3.5E. Nous ne traiterons ici que des nouvelles possibilités de ce compilateur. Pour ce qui est de l'utilisation générale du compilateur ainsi que de tous les menus standard, reportez-vous à la documentation propre au compilateur.

Le compilateur 3.5E comporte une seconde interface fournie sur la disquette sous le nom de MENUX.PRG. Il s'agit d'une variante du programme MENU.PRG des anciennes versions. Il comporte certains nouveaux points de menus que nous allons décrire.

Fichier / RCS **^R**

Ce point de menu permet d'appeler le programme RCS2.PRG. Ce nom peut bien entendu être modifié pour que vous puissiez indiquer votre propre éditeur de ressources. Dans ce cas, modifiez le contenu de la variable gfarcs$ dans MENUX.GFA.

Fichier / Exécuter **^X**

Ce point de menu permet de charger et exécuter n'importe quel programme .PRG. La sélection du PRG se fait par l'intermédiaire du sélecteur de fichier.

Sélection / PRG=GFA **F2**

Habituellement, lorsque vous compilez et linkez un programme .GFA, le résultat est toujours fourni sous la forme TEST.PRG. A charge ensuite pour vous de renommer le fichier. Dans cette version 3.5E, lorsque vous appuyez sur F2 ou vous cliquez sur Sélection/PRG=GFA, le programme .PRG portera le même nom que le programme .GFA. Ainsi la traduction de COMPTA.GFA

ne sera pas TEST.PRG mais COMPTA.PRG. Il vous suffit ainsi d'effectuer F1-Sélecteur fichier-F10 pour compiler un fichier .GFA.

Attention, cette commande est différente de Sélection/G3PRG. Celle-ci vous permet de donner un autre nom que TEST.PRG, par exemple SORTIE.PRG. Mais dans ce cas, tout programme .GFA sera traduit en SORTIE.PRG.

Sélection / C-Objetc C

Ce point de menu permet de spécifier un fichier objet .O destiné à être compilé (DR-C, Assembleur, etc).

Après chaque appel de programme, la valeur de retour du programme correspondant est affichée. 0 signifie OK, des nombres négatifs correspondent à des erreurs (comme fichier non trouvé), des nombres positifs pour d'autres erreurs. Le compilateur renvoie le nombre des instructions non compilées, le linker le nombre des symboles indéfinis et des dépassements offset.

Le linker GL affiche quelques messages d'erreurs :

```
?xxxxxxxx   Symbole inconnu xxxxxxxx
+xxxxxxxx   Redéfinition du symbole
>xxxxxxxx   Offset 16 octets trop grand
```

Ces messages ne doivent jamais intervenir dans les programmes GFA Basic sans $X.

C: CALL

Vous ne pouvez lire malheureusement qu'entre les lignes du manuel du GFA BASIC que ces routines, comme les routines C sur les ordinateurs 68000 habituellement utilisés, ne doivent pas modifier les registres A3 jusqu'à A6 et A7, bien que ceci n'ait aucune conséquence dans l'interpréteur. Pour C:, il n'y a presque

jamais de paramètre dans D0, une particularité fortuite de l'interpréteur.

Pour pouvoir utiliser des routines assembleur qui ne suivent pas cette convention, il y a l'option compilateur $C+ et $C-. $C+ permet de placer les registres A3 A6 sur pile avant l'appel C: ou l'appel Call et ensuite de les restaurer.

UNPACK.GFA

Ce programme convertit TEST.O sauvé en un format standard DR TESTX.O.

Achevé d'imprimer
sur les presses de l'imprimerie IBP
à Rungis (Val-de-Marne 94) (1) 46.86.73.54
Dépôt légal - Octobre 1990
N° d'impression: 5390

ATARI ST+STE

GFA BASIC 3.5E

Possesseurs du GFA Basic 3.01 à 3.5 et GFA Compilateur 3.01 à 3.5, cette mise à jour vous permet d'acquérir la version 3.5E des nouveaux interpréteur et compilateur. C'est pourquoi seules les nouvelles caractéristiques de ces produits sont étudiées dans ce manuel.

L'INTERPRETEUR GFA BASIC 3.5E

C'est toute la puissance du GFA Basic 3.0 plus de nouveaux outils et fonctions pour faciliter la tâche des programmeurs :

- Un éditeur encore plus souple et un plus grand confort d'utilisation : scrolling rapide, repliage des fonctions, recherche étendue aux procédures et fonctions repliées.
- Plus de 50 nouvelles instructions pour simplifier votre travail :

fonctions statistiques : factorielle, calcul des combinaisons et permutations;
fonctions matricielles : initialisation, déterminant, inverse, opérations arithmétiques, transfert et copie de matrices... Redéfinition des pointeurs sur les zones de données (DATA).

- Des fonctions puissantes pour exploiter toutes les extensions du STE : interface MICROWIRE, son stéréo...

Avec la version 3.5E de nouveaux domaines d'application vous sont aujourd'hui immédiatement accessibles : gestion, maths, physique, graphisme, simulation, traitement du signal...

LE COMPILATEUR GFA 3.5E

Le compilateur optimise au même titre que les 400 autres commandes du langage les nouvelles instructions du GFA Basic 3.5E. Il intègre une nouvelle interface MENUX pour appeler directement un éditeur de ressources, un programme externe...

3 325121 030420

REF. ST 042.

EDITIONS MICRO APPLICATION
58 RUE DU FAUBOURG POISSONNIERE
75010 PARIS TEL (1) 47 70 32 44