SMAKY6

M A T P A C : appels arithmétiques et de gestion de fichiers

Février 1982

EPSITEC-system sa

M O D E D ' E M P L O I D E M A T P A C

GENERALITES

Ce *package* est construit sous forme D'EXTENSION DES APPELS DU SYSTEME. Il réside en mémoire DEPUIS L'ADRESSE 25000 et OCCUPE 11400 bytes.

Il comprend deux parties:

- APPELS MATHEMATIQUES
- APPELS DE GESTION DE FICHIER A STRUCTURE DE RECORD

COMMENT UTILISER MATPAC

Pour pouvoir utiliser les appels de ce *package*, il faut évidemment que celui-ci soit chargé en mémoire. Pour cela il suffit de mettre dans son programme les instructions suivantes:

```
.REF    MAT
```

Le .REF donne accès aux symboles du *package*.

Pour charger le *package* l'utilisateur adjoindra AU DEBUT DE SON PROGRAMME les instructions suivantes qui se chargent d'initialiser l'adresse de l'extension de la table des numéros d'appel TABEX et de charger le *package*

```
;       ERNSA   =       34      ; message no starting adress

        LOAD    HL,#VATABEX     ; initialise l'adresse de
        LOAD    TABEX,HL        ; l'extension de la table
        LOAD    DE,#TMATH       ; pointe le nom du package
        .W      ?LGO            ; charge le package
        COMP    A,#ERNSA        ; teste si le chargement est
        JUMP,NE ERROR           ; correct, sinon traitement de l'erreur
                                ; au gré de l'utilisateur

TMATH:  .ASCIZ  /MATPAC.SY/
```

Cette méthode est la plus sûre, puisqu'elle assure un chargement au début de chaque programme qui le nécessite.

Mais un utilisateur qui employerait systématiquement les appels du package ET QUI EST SUR DE NE JAMAIS DETRUIRE LA PARTIE MEMOIRE OCCUPEE PAR LE *PACKAGE* OU DE MODIFIER TABEX, peut utiliser la méthode suivante qui charge AUTOMATIQUEMENT LE *PACKAGE* AU MOMENT DU RESET.

Il suffit d'assembler le petit programme suivant et de l'APPELER ST.SY ainsi il sera exécuté à chaque reset

```
        .TITLE  MATPAC LOADER
        .PROC   Z80
        .REF    FLO

        .LOC    53000

;       ERNSA   =       34      ; message no starting address

START:
        LOAD    HL,#VATABEX     ; initialise l'adresse de
        LOAD    TABEX,HL        ; l'extension de la table
START1:
        LOAD    DE,#TMATH       ; pointe le nom du package
        .W      ?LGO            ; charge le package
        COMP    A,#ERNSA        ; teste si le chargement est
        JUMP,NE ERROR           ; correct, sinon erreur
        .W      ?TEXTIM
.ASCIZ  /<CR>MATPAC LOADED/
        .W      ?RTN            ; retour au CLI

ERROR:
        .W      ?BUZZ,?BUZZ,?BUZZ
        .W      ?TEXTIM
.ASCIZ  /<CR>MATPAC LOAD ERROR: /
        .W      ?ERROR
        .W      ?TEXTIM
.ASCIZ  /<CR>TRY AGAIN ? /
        .W      ?GETCAR,?DICAR
        CLR     A:5
        COMP    A,#'Y
        JUMP,EQ START1
        .W      ?RTN

TMATH:  .ASCIZ  /MATPAC.SY/

        .END    START
```

Pour le reste, l'utilisation des appels est exactement semblable aux appels du système.

APPELS MATHEMATIQUES

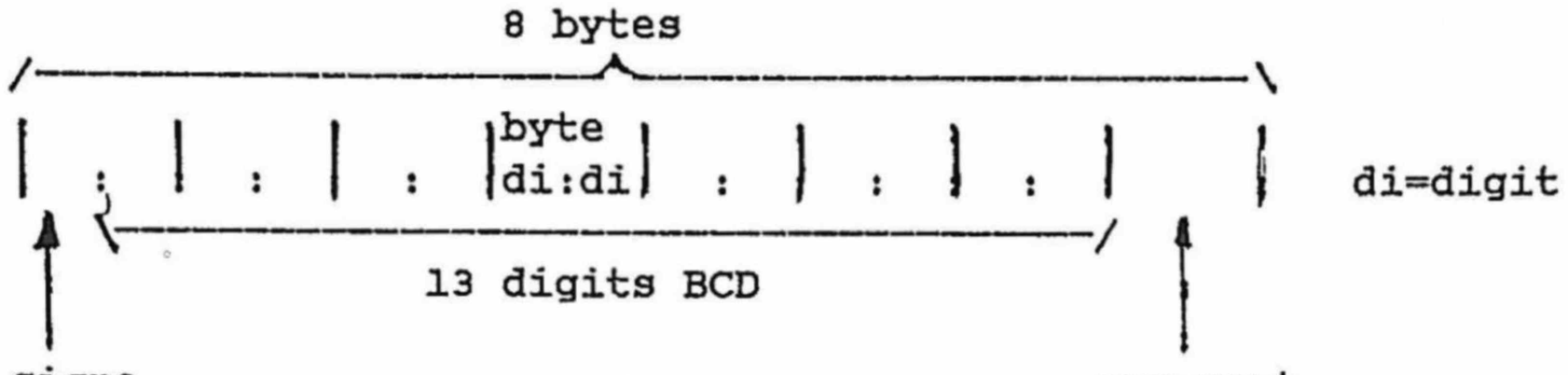
Ces appels sont réalisés autour de routines de calcul ZILOG écrites pour le processeur Z80.
Ces appels permettent de traiter des nombres réels de 13 chiffres significatifs allant de 10 puissance -127 à 10 puissance +126.
L'affichage des nombres peut se faire en trois modes différents:

. virgule flottante
. virgule fixe
. notation scientifique

CARACTERISTIQUES DES VARIABLES BINAIRES ET ASCII

Description des variables binaires codées décimales.

Les variables binaires sont en BINAIRE CODE DECIMAL et elles NECESSITENT 8 BYTES. Voici le format de ces variables:



177 = 10 puissance -2
200 = 10 puissance -1
201 = 10 puissance 0
202 = 10 puissance 1

S = bit de signe 0 = pos
X = bits non significatifs

Description des variables ascii en entrée

Les variables ascii peuvent avoir plus de 13 chiffres significatifs. Cependant seuls les 13 premiers chiffres seront pris en compte. PAR CONTRE L'EXPOSANT TIENDRA COMPTE DE TOUS LES CHIFFRES PRESENTS. Les caractères espace sont ignorés.
Les nombres peuvent être exprimés en notation scientifique. On utilise à cet effet la lettre E majuscule. La fin de la variable ascii n'est pas marquée par un signe particulier mais par le premier caractère hors syntaxe.

Voici cette syntaxe:

- Le premier caractère peut être un signe moins, un signe plus ou un chiffre.
- Les caractères suivants des chiffres, un point ou la lettre E majuscule pour signifier l'entrée d'un exposant.
- On ne peut avoir qu'un seul point et qu'une lettre E.
- Après la lettre E on peut avoir soit un chiffre ou le signe moins, le signe plus puis d'autres chiffres.

Exemples de chaîne ascii:

```
|       1234567890987654321.123456*
\------/\----------/              ^ pris comme terminateur
 ignoré  pris en compte
         \----------------/
          pris en compte pour l'exposant
```

```
|        +1223.234 E-12-+234.689E-13x
           ignoré ^      ^ début 2ème chaîne
                         ^ pris comme terminateur
```

Quelques autres chaînes acceptées:

```
        .ASCIZ  /-.345E-2/
        .ASCIZ  /00000.0000001/
        .ASCIZ  /12.96E-38 = coefficient de correction/
```

Quelques chaînes inacceptées:

```
        .ASCIZ  /E-14/                  exposant seul
        .ASCIZ  /12.1234E-A1/           lettre dans l'exposant
        .ASCIZ  /12346.1234 EABC/       idem
```

Description des variables asciz en sortie

Les appels ?BINDEC, ?DIBIDE, ?WRBIDE transforment une variable binaire en une variable ASCIZ.
Le format de cette variable est soit une notation en virgule fixe ou une notation en virgule flottante ou une notation scientifique.
Pour la notation en virgule fixe, le nombre de digits (chiffres avant la virgule) ainsi que le nombre de décimales (chiffres après la virgule) peut être sélectionné par l'appel ?DECFOR. Lorsqu'il n'est pas possible d'afficher la variable selon le format sélectionné, ces appels passent automatiquement en notation scientifique avec le maximun de chiffres significatifs. Ce passage ce fait soit si l'ordre de grandeur du nombre à afficher est supérieur au nombre de digits sélectionné ou pour les nombres plus petits que 1, si le premier chiffre significatif n'est pas affichable avec le nombre de décimales sélectionné.

On peut également forcer dans tous les cas la notation scientifique ou sélectionner la notation virgule flottante à l'aide de cet appel.

?BINDEC et ?DIBIDE cadrent le nombre. C'est à dire que le nombre de positions qui sépare le pointeur d'origine de la position des unités EST CONSTANT ET CORRESPOND AU NOMBRE DE DIGITS SELECTIONNES PLUS UNE POSITION POUR LE SIGNE. Les positions inutilisées sont remplies avec des espaces. Par contre si l'on veut par exemple afficher un nombre directement après un texte, les positions inutilisées remplies d'espaces peuvent être gênantes. C'est la raison pour laquelle l'appel ?WRBIDE affiche depuis le premier chiffre ou signe.

nombres cadrés	nombres non cadrés
12.2350	12.2350
-123452.2340	-123452.2340
0.0012	0.0012
1235445.5482	1235445.5482
192.2390	192.2390
-1.0000	-1.0000

LISTE DES ERREURS

```
ERLLI   =       5       ; line too long
ERZDI   =      37       ; divide by zero
EROVE   =      40       ; overflow
ERUND   =      41       ; underflow
ERILN   =      42       ; illegal number
ERSQR   =      43       ; negative square-root
ERLOG   =      44       ; negative logarithm
```

DESCRIPTION DES APPELS

****
* ?BINDEC *
****

Cet appel permet la conversion de la variable binaire pointée par HL en une chaîne ASCII mise en mémoire depuis DE. Les digits sélectionnés par ?DECFOR qui ne sont pas utilisés par le nombre sont remplacés par des espaces.

Modifie: rien

****
* ?WRBIDE *
****

Cet appel affiche la variable binaire pointée par HL sur l'écran. Cet appel NE CADRE PAS L'AFFICHAGE, c'est à dire que le premier caractère affiché est le premier caractère du nombre à afficher. Cet appel utilise ?DICAR.

Modifie: rien

****
* ?DIBIDE *
****

Cet appel affiche la variable binaire pointée par HL sur l'écran. Cet appel CADRE L'AFFICHAGE. Les digits sélectionnés par ?DECFOR qui ne sont pas utilisés par le nombre sont remplacés par des espaces.

Modifie: rien

****
* ?DECBIN *
****

Cet appel convertit la variable ASCII pointée par HL en une variable binaire mise en mémoire depuis DE. En cas d'erreur on revient CARRY SET et A contient alors le numéro de l'erreur.

Erreurs possibles:

40	=	dépassement de capacité supérieur
41	=	dépassement de capacité inférieur
42	=	nombre ascii illégal

Modifie: AF

****
* ?HLTBIN * HL -----> (DE)
****

Convertit la valeur binaire signée 16 bits de HL en une variable binaire BCD pointée par DE

Modifie: rien

****
* ?BINTHL * (DE) ----> HL
****

Convertit la variable binaire BCD pointée par DE en une valeur binaire BCD pointée par DE.

Erreurs possible:

U0	=	dépassement de capacité supérieur
U1	=	dépassement de capacité inférieur

Modifie: AF

****
* ?GEDEBI *
****

Cet appel permet la saisie d'une variable binaire depuis le clavier avec écho sur l'écran. La variable binaire est mise en mémoire depuis DE. La sortie de cet appel est provoquée par le premier caractère hors syntaxe. La touche BS permet de revenir en arrière jusqu'au début de la saisie, la touche DEL annule tous les caractères déjà saisis.

Syntaxe du fonctionnement de l'appel:
- Le premier caractère peut être un signe moins ou un chiffre.
- Les caractères suivants des chiffres, un point ou la lettre E majuscule pour signifier l'entrée d'un exposant.
- On ne peut taper qu'un seul point et qu'une lettre E.
- Après la lettre E on peut taper soit un chiffre ou le signe moins, puis d'autres chiffres.
- Tout caractère hors syntaxe provoque la fin de la saisie au clavier et, pour autant qu'il ne s'agisse pas du premier caractère, la conversion en binaire avant le retour de l'appel.
- Ne sont considérés comme tapés que les caractères visibles sur l'écran. On peut donc par exemple retaper la lettre E si la précédente lettre E a été effacée par BS ou DEL.

Si l'appel s'est effectué correctement, A contient le dernier caractère tapé. En cas d'erreur on revient CARRY SET et A contient alors le numéro de l'erreur.

Erreurs possibles:

40	=	dépassement de capacité supérieur
41	=	dépassement de capacité inférieur
42	=	nombre ascii illégal

Modifie: AF

****
* ?DECFOR *
****

Cet appel initialise le format d'affichage des variables ASCIZ délivrées par les appels ?BINDEC, ?DIBIDE, ?WRBIDE.
B spécifie le nombre maximum de digits affichables et C le nombre de décimales à afficher. On peut sélectionner au maximum 13 digits, 12 décimales. Cependant, la somme de ces deux nombres ne dépassera pas 13. De toute façon, si l'on n'a pas respecté ces règles, l'appel corrige automatiquement. Les décimales sont plafonnées à 12 et les digits sont plafonnés en fonction des décimales. L'appel rend alors les valeurs prises. Si l'on spécifie 0 digit et un nombre de décimales non nul, on force l'affichage permanent en notation scientifique. Si l'on spécifie 0 digit et 0 décimale, on sélectionne l'affichage en virgule flottante.

Modifie: BC

****
* ?FOPER *
****

Cet appel effectue une des quatre opérations, plus, moins, fois, divisé selon le caractère ASCII correspondant dans A.
L'opération par défaut (A diff de '+ '- '* '/) est l'addition. Le comportement de l'appel est identique à celui de l'opération effectuée (voir ci-après).

Modifie: AF

****
* ?FADD * (HL) = (HL) + (DE)
****

Cet appel effectue l'addition de la variable binaire pointée par HL avec la variable binaire pointée par DE. Le résultat est à la place de la variable pointée par HL.
En cas d'erreur on revient CARRY SET avec dans A le numéro de l'erreur.

Erreurs possibles:

40	=	dépassement de capacité supérieur
41	=	dépassement de capacité inférieur

Modifie: AF

****
* ?FSUB * (HL) = (HL) - (DE)
****

Cet appel effectue la soustraction de la variable binaire pointée par HL avec la variable binaire pointée par DE. Le résultat est à la place de la variable pointée par HL.
En cas d'erreur on revient CARRY SET avec dans A le numéro de l'erreur.

Erreurs possibles:

40	=	dépassement de capacité supérieur
41	=	dépassement de capacité inférieur

Modifie: AF

****
* ?FMUL * (HL) = (HL) * (DE)
****

Cet appel effectue la multiplication de la variable binaire pointée par HL avec la variable binaire pointée par DE. Le résultat est à la place de la variable pointée par HL.
En cas d'erreur on revient CARRY SET avec dans A le numéro de l'erreur.

Erreurs possibles:

40	=	dépassement de capacité supérieur
41	=	dépassement de capacité inférieur

Modifie: AF,DE

****
* ?FDIV * (HL) = (HL) / (DE)
****

Cet appel effectue la division de la variable binaire pointée par HL avec la variable binaire pointée par DE. Le résultat est à la place de la variable pointée par HL.
En cas d'erreur on revient CARRY SET avec dans A le numéro de l'erreur.

Erreurs possibles:

37	=	division par zéro
40	=	dépassement de capacité supérieur
41	=	dépassement de capacité inférieur

Modifie: AF

****
* ?FPOW * (HL) = (HL) puissance (DE)
****

Cet appel élève la variable binaire pointée par HL à la puissance: variable binaire pointée par DE. Le résultat est à la place de la variable pointée par HL.
En cas d'erreur on revient CARRY SET avec dans A le numéro de l'erreur.

Erreurs possibles:

40	=	dépassement de capacité supérieur
41	=	dépassement de capacité inférieur

Modifie: AF

****
* ?FSQRT * (HL) = racine carrée (HL)
****

Cet appel extrait la racine carrée de la variable binaire pointée par HL le résultat étant mis à la place de la variable.
En cas d'erreur on revient CARRY SET avec dans A le numéro de l'erreur.

Erreur possible:

43	=	racine carrée d'un nombre négatif

Modifie: AF

****
* ?FEXP * (HL) = exponentiel (HL)
****

Cet appel donne l'exponentielle de la variable binaire pointée par HL le résultat étant mis à la place de la variable.
En cas d'erreur on revient CARRY SET avec dans A le numéro de l'erreur.

Erreurs possibles:

40	=	dépassement de capacité supérieur
41	=	dépassement de capacité inférieur

Modifie: AF

****
* ?FLOG * (HL) = logarithme népérien (HL)
****

Cet appel prend le logarithme népérien de la variable binaire pointée par HL le résultat étant mis à la place de la variable.
En cas d'erreur on revient CARRY SET avec dans A le numéro de l'erreur.

Erreur possible:

44 = logarithme d'un nombre négatif

Modifie: AF

****
* ?FSIN * (HL) = sinus (HL)
****

Cet appel prend le sinus de la variable binaire pointée par HL le résultat étant mis à la place de la variable.

Modifie: rien

****
* ?FCOS * (HL) = cosinus (HL)
****

Cet appel prend le cosinus de la variable binaire pointée par HL le résultat étant mis à la place de la variable.

Modifie: rien

****
* ?FTAN * (HL) = arc-tangente (HL)
****

Cet appel prend le tangente de la variable binaire pointée par HL le résultat étant mis à la place de la variable.
En cas d'erreur on revient CARRY SET avec dans A le numéro de l'erreur.

Erreurs possibles:

40 = dépassement de capacité supérieur
41 = dépassement de capacité inférieur

Modifie: AF

****
* ?FATAN * (HL) = arc-tangente (HL)
****

Cet appel prend l'arc-tangente de la variable binaire pointée par HL le résultat étant mis à la place de la variable.

Modifie: rien

****
* ?FINVER * (HL) = 1 / (HL)
****

Cet appel prend l'inverse de la variable binaire pointée par HL le résultat étant mis à la place de la variable.
En cas d'erreur on revient CARRY SET avec dans A le numéro de l'erreur.

Erreur possible:

37 = division par zéro

Modifie: AF

****
* ?RANDOM * (HL) = nombre aléatoire entre 0 et 1
****

Cet appel génère un nombre binaire compris entre 0 et 1 de façon aléatoire le résultat étant mis à la place pointée par HL.

Modifie: rien

****
* ?FINT * (HL) = partie entière (HL)
****

Cet appel prend la partie entière de la variable binaire pointée par HL le résultat étant mis à la place de la variable.

Modifie: rien

****
* ?FFRAC * (HL) = partie fractionnaire (HL)
****

Cet appel prend la partie fractionnaire de la variable binaire pointée par HL le résultat étant mis à la place de la variable.

Modifie: rien

****
* ?FCLR * (HL) = 0
****

Cet appel met à zéro la variable binaire pointée par HL.

Modifie: rien

****
* ?FPI * (HL) = PI
****

Cet appel met la valeur PI dans la variable binaire pointée par HL.

Modifie: rien

****
* ?FABS * (HL) = valeur absolue (HL)
****

Cet appel prend la valeur absolue de la variable binaire pointée par HL.

Modifie: rien

****
* ?FCHSIG *   (HL) = signe opposé (HL)
****

Cet appel change le signe de la variable pointée par HL.

Modifie: rien

****
* ?FRND   *   (HL) = RND (HL)
****

Cet appel arrondit la variable binaire pointée par HL. L'arrondi est fait sur la dernière décimale. Le nombre de décimales est sélectionné par l'appel ?DECFOR.

Modifie: rien

****
* ?FRND5 *   (HL) = RND5 (HL)
****

Cet appel arrondit à 5 la variable binaire pointée par HL. L'arrondi est fait sur la dernière décimale. Le nombre de décimales est sélectionné par l'appel ?DECFOR.

Exemple: arrondi effectué pour 2 décimales:

1.92	=>	1.90
1.93	=>	1.95
1.97	=>	1.95
1.98	=>	2.00

Modifie: rien

****
* ?FCOMP *   F = COMP (HL),(DE)
****

Cet appel effectue la comparaison entre la variable binaire pointée par HL et la variable binaire pointée par DE. Les bits du registre F sont mis à jour.

Cette comparaison permet d'utiliser les conditions:

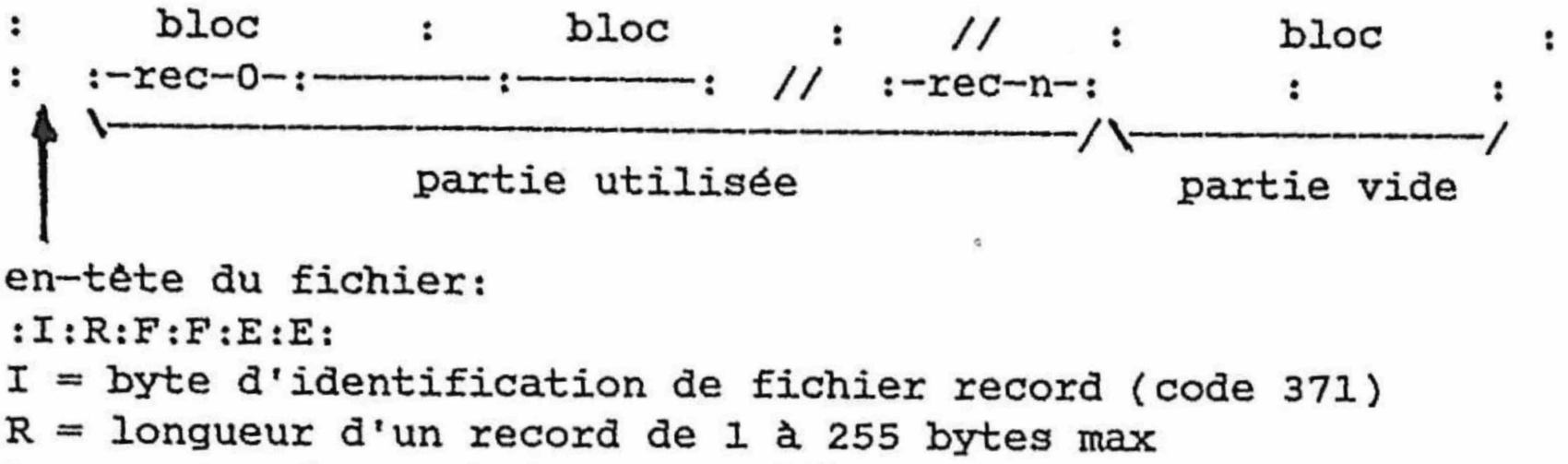
higher or same	=	carry clear
lower	=	carry set
non equal	=	zero clear
equal	=	zero set

Modifie: F

APPELS DE GESTION DE FICHIER A STRUCTURE DE RECORD

DEFINITIONS

L'utilisation de ces appels permet la création et la gestion de fichier disque ayant la structure suivante:



I = byte d'identification de fichier record (code 371)
R = longueur d'un record de 1 à 255 bytes max
F = numéro du prochain record libre
E = nombre de records dans le fichier

Comme le montre le schéma ci-dessus, le fichier contient UNE EN-TETE qui décrit le fichier, UNE PARTIE UTILISEE, et UNE PARTIE VIDE. Ces deux parties sont subdivisées en RECORD de longueur égale et définie à la céation du fichier. La frontière entre ces deux parties représente LA FIN DU FICHIER. Chacune de ces parties peut évidemment être de longueur nulle. Ainsi, immédiatement après sa création, le fichier ne contiendra qu'une partie vide. L'appel ?ADDREC permet de créer, puis d'agrandir la partie utilisée. Les autres appels traitant des records se rapportent à la partie utilisée exclusivement.

GENERALITES ET RECOMMENDATIONS

Ces appels sont constuits à l'aide des appels de base de SAMOS. Il est donc nécessaire de bien avoir présent à l'esprit le fonctionnement des appels de SAMOS et de se réferer si besoin est à la notice SAMOS.
Les fichiers à structure de records seront exclusivement traités par les appels de ce *package*. On ne peut pas, par exemple, ouvrir le fichier avec un appel SAMOS puis le traiter avec des appels de ce *package*.
Il est par contre possible d'utiliser séparément mais simultanément des fichiers simples avec directement les appels SAMOS, et des fichiers à structure de records avec les appels de ce *package*. La seule restriction importante est DE NE PAS UTILISER L'APPEL SAMOS ?RESET TANT QUE DES FICHIERS A STRUCTURE DE RECORDS SONT OUVERTS.

LES LIMITES

Les limites globales sont naturellement celle de SAMOS. Cela vaut surtout si l'on traite simultanément les deux types de fichier. Voici cependant les limites spécifiques du *package* valable si l'on ne traite que des fichiers à structure de records.

- au maximum 4 fichiers ouverts simultanément.
- longueur du record comprise entre 1 et 255 caractères.
- longueur minimum d'un buffer de travail égale à 2 blocs.

Les appels du *package* n'utilisent l'ouverture en écriture en accès par blocs que durant l'execution de l'appel ?RCREAT. On dispose donc toujours avec SAMOS, de 8 canaux en écriture. Le *package* n'utilise pas les buffers de SAMOS.

DESCRIPTION DES APPELS

****
* ?RCREAT *
****

Cet appel crée une fichier à structure de records. Cette création s'effectue de la manière suivante:
Création d'un fichier avec réservation de la bonne taille calculée à partir de la longueur du record et du nombre de records, écriture de tous les blocs avec dans le bloc 0 l'en-tête du fichier, fermeture du fichier.

Paramètres d'entrée: DE = pointeur au nom
BC = nombre de records
A = longueur du record

Paramètres de sortie: CC = exécution correcte de l'appel
CS = erreur, numéro d'erreur dans A

Modifie: AF

Liste des erreurs possibles:

11 = file already exist
13 = illegal filename
21 = unknown device
24 = all channel in use
25 = directory full
26 = disk full
30 = device timeout
31 = write protect tab set
32 = write error
33 = read error
45 = record length nul
46 = zero record

****
* ?ROPEN *
****

Cet appel ouvre un fichier à structure de records pour lecture, écriture ou addition de records.
L'ouverture du fichier est faite en accès par bloc. En effet, toutes les opérations qui seront effectuées seront émulées à l'aide de l'appel SAMOS ?UPDATE.
Le buffer de travail est attribué par l'utilisateur, ce qui permet d'optimiser au mieux les temps de réponse. Il faut évidemment prendre garde de ne pas oublier d'attribuer ce buffer et que celui-ci soit situé dans une zone absolument libre. Attention au stack, au soft système, à son propre programme. Le buffer doit avoir une taille de 2 blocs minimum. Juste après l'ouverture, le pointeur du fichier (position dans le fichier) est sur le record 0.

Paramètres d'entrée: DE = pointeur au nom
HL = pointeur début du buffer
A = nombre de blocs pour le buffer

Paramètres de sortie: CC = correct, no de canal dans A
CS = erreur, numéro d'erreur dans A

Modifie: AF

Liste des erreurs possibles:

2 = read protect file
12 = file does not exist
13 = illegal filename
20 = file in use for reading
21 = unknown device
24 = all channel in use
30 = device timeout
32 = write error
33 = read error
50 = buffer to small

****
* ?RCLOSE *
****

Cet appel ferme un fichier à structure de records. S'il y a lieu la dernière tranche traitée dans le buffer est écrite sur le fichier (écriture de records) et l'en-tête du fichier est mise à jour (addition de records). Finalement le fichier est fermé.

Paramètre d'entrée: A = canal

Paramètres de sortie: CC = exécution correcte de l'appel
CS = erreur, numéro d'erreur dans A

Modifie: AF

Liste des erreurs possibles:

22 = channel error
30 = device timeout
32 = write error
33 = read error

****
* ?RRESET *
****

Cet appel ferme tous les fichiers à structure de records ouverts. Cet appel n'a pas de contrôle d'erreur. Si une erreur survient durant la fermeture d'un canal, celui-ci est purement et simplement supprimé. Cet appel ne doit en principe être utilisé que pour une fermeture d'urgence.

Modifie: rien

****
* ?RDREC *
****

Cet appel lit le record courant.
Transfert du buffer à l'utilisateur. Si le record courant n'est pas dans le buffer, écrit s'il y a lieu le buffer dans le fichier, puis lit la tranche du fichier qui contient le record courant. Le pointeur du record courant n'est pas modifié.

Paramètres d'entrée: A = canal
DE = pointeur en mémoire

Paramètres de sortie: CC = exécution correcte de l'appel
CS = erreur, numéro d'erreur dans A

Modifie: AF

Liste des erreurs possibles:

22 = channel error
30 = device timeout
32 = write error
33 = read error

****
* ?WRREC *
****

Cet appel écrit le record courant.
Transfère de l'utilisateur au buffer. Si le record courant n'est pas dans le buffer, écrit s'il y a lieu le buffer dans le fichier, puis lit la tranche du fichier qui contient le record courant. Mémorise qu'une opération d'écriture a eu lieu et qu'il faudra donc écrire le buffer dans le fichier. Le pointeur du record courant n'est pas modifié.

Paramètres d'entrée: A = canal
DE = pointeur en mémoire

Paramètres de sortie: CC = exécution correcte de l'appel
CS = erreur, numéro d'erreur dans A

Modifie: AF

Liste des erreurs possibles:

22 = channel error
30 = device timeout
32 = write error
33 = read error

****
* ?NEXREC *
****

Cet appel pointe le record suivant.

Paramètre d'entrée: A = canal

Paramètres de sortie: CC = exécution correcte de l'appel
CS = erreur, numéro d'erreur dans A

Modifie: AF

Liste des erreurs possibles:

6 = end of file
22 = channel error

****
* ?ADDREC *
****

Cet appel ajoute un record au fichier.
Transfert du l'utilisateur au buffer. Si le record en ajout n'est pas dans le buffer, écrit s'il y a lieu le buffer dans le fichier, puis lit la tranche du fichier qui contient le record en ajout. Mémorise qu'une opération d'écriture a eu lieu et quil faudra donc écrire le buffer dans le fichier. Mémorise également qu'un ajout a eu lieu et qu'il faudra à la fermeture du fichier mettre à jour l'en-tête du fichier. Le pointeur du record courant est sur le record ajouté donc sur le nouveau dernier record du fichier.

Paramètres d'entrée: A = canal
DE = pointeur en mémoire

Paramètres de sortie: CC = exécution correcte de l'appel
CS = erreur, numéro d'erreur dans A

Modifie: AF

Liste des erreurs possibles:

 6 = end of file
22 = channel error
30 = device timeout
32 = write error
33 = read error

****
* ?SETREC *
****

Cet appel positionne sur le record spécifié.

Paramètres d'entrée: A = canal
                      HL = numéro du record

Paramètres de sortie: CC = exécution correcte de l'appel
                      CS = erreur, numéro d'erreur dans A

Modifie: AF

Liste des erreurs possibles:

17 = out of file
22 = channel error

****
* ?GETREC *
****

Cet appel donne le numéro du record courant.

Paramètres d'entrée: A = canal

Paramètre de sortie: HL = numéro du record courant
                      CS = erreur, numéro d'erreur dans A

Modifie: AF,HL

Erreur possible: 22 = channel error

****
* ?POSREC *
****

Cet appel recherche depuis le record courant le premier record qui contient la chaîne pointée par DE de longueur B et commençant à C caractères du début du record. Si la recherche aboutit, le record trouvé devient le record courant. Dans le cas contraire on a le message d'erreur *end of file* et le record courant est le dernier record du fichier.
La recherche commence tout d'abord dans le buffer. Si l'on ne trouve pas dans le buffer, celui-ci est écrit dans le fichier s'il y a lieu puis la tranche suivante du fichier est lue dans le buffer et la recherche continue. Ce processus se répétant jusqu'à ce que la recherche aboutisse ou que l'on ait atteint la fin du fichier.

Paramètres d'entrée: A = canal
DE = pointeur argument de recherché
B = longueur argument de recherche
C = offset dans le record

Paramètres de sortie: CC = exécution correcte de l'appel
CS = erreur, numéro d'erreur dans A

Modifie: AF

Liste des erreurs possibles:

6 = end of file
22 = channel error
30 = device timeout
32 = write error
33 = read error
51 = illegal search parameters

****
* ?RARGS *
****

Donne les paramètres du fichier ouvert de canal A

Paramètre de sortie: A = canal
HL = nombre de records dans le fichier
DE = nombre de records utilisés

Modifie: AF, HL, DE

Erreur possible: 22 = channel error

Remarque: La soustraction directe en sortie d'appel HL-DE donne dans HL de nombre de records libres.

****
* ?WARGS *
****

Change le nombre de records utilisés.

Paramètre d'entrée: DE = Nombre de records utilisés

Modifie: AF

Erreurs possibles: 17 = out of file
22 = channel error

Remarque:
Cet appel permet de supprimer physiquement des enregistrements à la fin du fichier.
Il permet également d'en créer avec un contenu quelconque.