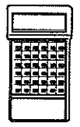
PSION PC

Bulletin Technique

Octobre - Novembre - Décembre 1990 — La revue du support technique d'Aware

Mobile Computer

- Import/Export de fichier OPL



Organiseur II

- La roue de la fortune
- Du beep à Bach
- Gestion de temps
- Redéfinir le clavier

PC4 & Xchange

- Une procédure d'entrée

Edité par : Aware
21 Rue Olivier Métra
75020 Paris
Tél. : (1) 46 36 46 47
Télex : 216 244 F
Fax : (1) 46 36 82 54

Apple Expo 91 : un rendez-vous à ne pas manquer !

Alors que commence une nouvelle année de travail pleine de promesses, j'espère que, comme nous, vous avez fait le plein d'énergie pendant vos vacances !

En effet, c'est sur les "chapeaux de roue" que nous démarrons avec, dès le 18 septembre, une Apple Expo où nous vous réservons une surprise de taille que vous ne devez manquer sous aucun prétexte !

Motus... pour le moment ! Mais surtout, surtout, n'oubliez pas de venir nous rendre visite sur notre stand (N°2C20) du 18 au 21 septembre 1991 au CNIT à La Défense.

Hormis cet événement que nous attendons tous avec impatience, les autres produits Psion continuent à intéresser toujours plus d'utilisateurs et à générer toujours plus d'applications : l'Organiseur II, déjà, populaire pour la lecture de codes à barres ou les relevés de compteurs, est utilisé pour de nouvelles applications à la pointe de la technologie ; plus récent, le HC, avec son écran graphique, son système multi-tâches et sa puissance a déjà une place de choix sur le marché des terminaux de saisie professionnels.

Et si vous regardez bien, la gamme Psion est à ce point complète et variée, qu'il n'est pas une seule application de saisie de données sur le terrain qui ne trouve sa solution avec un de ces produits.

Et dans ce numéro, vous trouverez une foule d'articles qui vous permettront d'aller encore plus loin dans leur utilisation.

Bonne lecture et surtout, venez nous voir au CNIT !

Martine Garrigues

Import/Export de fichiers OPL avec le MC 400

Par
Alexandre
Bouillot

Avec le MC 400, lorsque l'on veut utiliser un fichier en provenance d'un autre système (Mac, PC, Organiseur...) dans un programme OPL, plusieurs opérations sont à réaliser avec différents programmes : Emulation de terminal pour le transfert du fichier, Base de données pour la conversion du fichier en format OPL et enfin programme OPL pour l'utilisation du fichier. Mêmes choses dans l'autre sens. De plus, cela suppose que l'on n'utilise pas de zones numériques. Les procédures qui sont décrites ici permettent de transformer un fichier texte en un fichier de données pour l'OPL du MC. Ces procédures utilisent des fonctions non documentées dans les manuels de la version 1.29 mais fonctionnent correctement.

La procédure Import lit un fichier texte et l'importe dans un fichier MC. Cette procédure est à adapter pour vos propres besoins.

```
proc Import:
rem procédure permettant d'importer un fichier texte dans
rem un fichier OPL. Le fichier d'import est du format
rem séparateur tab comme ceux générés par l'Organiseur II
rem le fichier OPL est constitué de deux rubriques de
rem type caractère
rem V 1.00 Aß Aware 10'90
   local ret% : rem Valeur de retour des IO commands
   local fName$(128) : rem nom du fichier d'import
   local txt$(255) : rem var. tampon d'import des données
   local address% : rem adresse de la variable txt$
   local handle% : rem canal du fichier d'import
   local mode% : rem mode d'ouverture du fichier d'import
   screen 80,25
rem préparation de l'ouverture du fichier d'import
   fName$="import.imp" : rem nom du fichier d'import
   cls
   mode%=$0400 or $0020 : rem fichier texte partagé
   ret%=ioopen(handle%,fName$,mode%)
rem ouverture du fichier import
   if ret%<0
      showErr:(ret%)
      return
   endif
rem ouverture du fichier OPL
   trap create "test.odb",a,a$,b$
   trap open "test.odb",a,a$,b$
   address%=addr(txt$)
rem
rem Boucle d'importation
rem
   while 1
      ret%=ioread(handle%,address%+1,255)
      if ret%<0
         if ret%=-36 : rem fin de fichier
            break
         else
            showErr:(ret%)
            break
         endif
      else
         pokeb address%,ret%
         rem transfert de la longueur de la chaîne lue
         rem import des données
         print txt$
         if loc(txt$,chr$(9))>0
rem vérification de la présence de données à importer
            a.a$=left$(txt$,loc(txt$,chr$(9))-1)
            txt$=right$(txt$,len(txt$)-loc(txt$,chr$(9)))
            a.b$=txt$
            append
         endif
      endif
   endwh
   ret%=ioclose(handle%)
   if ret%
      showErr:(ret%)
   endif
endp

proc showErr:(val%)
   print "Erreur",val%,err$(val%)
   get
endp
```

La procédure Export lit un fichier OPL et l'exporte dans un fichier texte Cette procédure est à adapter pour vos propres besoins.

```
proc Export:
rem procédure permettant d'exporter un fichier OPL dans
rem un fichier texte. Le fichier d'export est du format
rem séparateur tab comme ceux utilisé par l'Organiseur
rem le fichier OPL est constitué de deux rubriques de
rem type caractère
rem V 1.00 Aß Aware 10'90
   local ret% : rem Valeur de retour des IO commands
   local fName$(128) : rem nom du fichier d'export
   local txt$(255) : rem var. tampon d'export des données
   local address% : rem adresse de la variable txt$
   local handle% : rem canal du fichier d'export
   local mode% : rem mode d'ouverture du fichier d'export
   screen 80,25
rem préparation de l'ouverture du fichier d'export
   fName$="export.exp" : rem nom du fichier d'export
   cls
   mode%=$0100 or $0020 or $0002
   rem fichier texte en lec/ecr créé ou remis à zéro
   ret%=ioopen(handle%,fName$,mode%)
rem ouverture du fichier export
   if ret%<0
      showErr:(ret%)
      return
   endif
rem ouverture du fichier OPL
   trap open "test.odb",a,a$,b$
rem
rem Boucle d'exportation
rem
   address%=addr(txt$)
   do
      txt$=a.a$+chr$(9)+a.b$
      ret%=iowrite(handle%,address%,len(txt$))
      next
   until eof
   ret%=ioclose(handle%)
   if ret%
      showErr:(ret%)
   endif
endp

proc showErr:(val%)
   print "Erreur",val%,err$(val%)
   get
endp
```

Voilà les deux procédures à adapter vous permettant d'importer et d'exporter des données à partir ou vers un fichier texte. Il reste encore à l'importer ou à l'exporter du MC en utilisant MCLINK ou l'Emulation de terminal. Attention, lors des transferts XMODEM, le fichier est complété avec des caractères de valeur ASCII égale à 0 en fin de fichier.

Par
Gilles Jean

La roue de la fortune

Il est assez rare que nous diffusions dans nos colonnes des jeux pour Organiseur II. Néanmoins, la routine listée ci-dessous présente à juste titre beaucoup d'intérêt. Mais c'est avant tout un excellent module de gestion de chaînes de caractères et c'est, d'ailleurs, le point que nous vous engageons à approfondir.

Attention : Ce programme ne fonctionne que sur Organiseur II LZ et LZ64. L'adaptabilité est toutefois sans grosse difficulté puis qu'il s'agit simplement de gérer l'affichage afin que les informations tiennent sur un écran de deux lignes de seize caractères.

Avant de lancer le programme, vous devez saisir un fichier contenant les informations à découvrir. Ce fichier doit être nommé ROUE et doit figurer sur le même volume que le programme. Si cela n'était pas le cas, vous devriez modifier l'instruction permettant d'ouvrir le fichier de référence (open "roue",a,a$,b$). La première variable (a.a$) correspond au champ recherché alors que le deuxième (a.b$) correspond à la famille d'appartenance.

```
roue:
local a%,c%,e%,r%,g%,d%,k%,l%,m%,f%(9)
local n(8),t(8)
local n$(8,8),m$(20),l$(20),v$(20),g$(1),a$(1),d$(255),el$(1)
```

```
rem ****************************************
rem a%=saisie du nombre de joueur          *
rem c%=compteur de defilement dans la chain*
rem e%=enjeu pour la recherche en cours    *
rem r%=compteur de tour                    *
rem g%=pointeur de table des joueurs       *
rem d%=drapeaux joueur suivant             *
rem l%=compteur                            *
rem m%=compteur                            *
rem f%(9)=enjeu en francs                  *
rem n(4)=cumul complet par joueur          *
rem t(4)=cumul pour le mot en cours        *
rem n$(4,6)=nom des joueurs                *
rem m$(20)=chaine initiale                 *
rem l$(20)=chaine contenant les consonnes  *
rem v$(20)=chaine contenant les voyelles   *
rem g$(1)=buffer clavier                   *
rem a$(1)=car d'affichage initiale         *
rem d$(255)=chaine d'affichage du resultat *
rem el$(1)=effacement de fin de ligne      *
rem ****************************************
udg 0,1,1,1,1,1,1,1,1
udg 1,31,0,0,0,0,0,0,0
udg 2,0,0,0,0,0,0,0,31
udg 3,31,16,16,16,16,16,16,16
udg 4,31,1,1,1,1,1,1,1
udg 5,16,16,16,16,16,16,16,31
udg 6,1,1,1,1,1,1,1,31
udg 7,16,16,16,16,16,16,16,16
rem ****************************************
rem Initialisation du jeu                  *
rem ****************************************
print chr$(3);rept$(chr$(1),18);chr$(4);
print chr$(7);"Roue de la fortune";chr$(0);
print chr$(7);"TopGil 11/90 - 2.0";chr$(0);
print chr$(5);rept$(chr$(2),18);chr$(6);
el$=chr$(26)
f%(1)=100
f%(2)=200
f%(3)=500
f%(4)=1000
f%(5)=2000
f%(6)=2500
f%(7)=5000
f%(8)=7500
f%(9)=0
pause -50
rem ****************************************
rem Chargement des joueurs (1 à 8)         *
rem ****************************************
cls
udg 0,31,31,25,25,31,31,0,31
print chr$(0);rept$(chr$(2),14)
clock(1)
do
at 1,2
print el$;"NB de joueurs ?",
trap input a%
if err
stop
endif
until a%>0 and a%<9
g%=1
while G%<=a%
at 1,3
```



```
at 1,3
print el$;"Joueur (";G%;")",
trap input n$(G%)
if err or len(n$(g%))=0
stop
else
n(g%)=0
g%=g%+1
endif
endwh
rem ****************************************
rem Ouverture du fichier                   *
rem ****************************************
open"roue",a,a$,b$
c%=1
do
r%=0
do
cursor off
rem ****************************************
rem Recherche de la chaîne à decouvrir     *
rem ****************************************
position int(rnd*count+1)
rem ****************************************
rem Chargement des variables M$,L$ & v$    *
rem ****************************************
d%=1
while d%<=len(a.a$)
if mid$(a.a$,d%,1)<>chr$(32)
a$="."
else a$=" "
endif
m$=m$+a$
d%=d%+1
endwh
d%=1
while d%<=len(a.a$)
a$=mid$(a.a$,d%,1)
if loc("AEIOU ",a$)=0
l$=l$+a$
else
if a$<>chr$(32) and loc(l$,a$)=0
v$=v$+a$
endif
endif
d%=d%+1
endwh
rem ****************************************
rem Affichage initial                      *
rem ****************************************
do
at 1,2
print el$;m$
print el$;a.b$
rem ****************************************
rem calcul de l'enjeu                      *
rem ****************************************
e%=0
do
e%=e%+10
beep 20,e%
until e%=300
e%=int(rnd*9)
if e%=0
beep 500,2500
```

```
rem ****************************************
rem Banqueroute hélas...                    *
rem ****************************************
at 1,4
print el$;n$(c%);"=Banqueroute"
t(c%)=0
d%=1
pause 25
at 1,4
print el$
else
label::
d%=1
at 1,4
kstat 1
rem ****************************************
rem Recherche et validiter du caractère     *
rem ****************************************
pokeb $74,0
g$=chr$(view(4,n$(c%)+"(="+fix$(n(c%),0,12)+"+"+fix$(t(c%),0,12)+
"F) pour "+fix$(f%(e%),0,5)+"F"))
at 1,4
print el$
if g$=chr$(2) and t(c%)>=2000
rem ****************************************
rem Acces aux voyelles                      *
rem ****************************************
at 1,4
print el$;n$(c%),"Voyelle ?",
rem ****************************************
rem Saisie d'une voyelle                    *
rem ****************************************
cursor on
pokeb $74,0
g$=get$
cursor off
if g$=chr$(1)
goto label::
endif
l%=1
while l%<=len(v$)
if mid$(v$,l%,1)=g$
v$=left$(v$,l%-1)+mid$(v$,l%+1,20)
if d%=1
d%=0
endif
endif
l%=l%+1
endwh
l%=1
if d%=0
while l%<=len(a.a$)
if mid$(a.a$,l%,1)=g$
m$=left$(m$,l%-1)+g$+mid$(m$,l%+1,20)
at 1,2
print m$
beep 50,150
endif
l%=l%+1
endwh
rem ****************************************
rem Retrait du cout d'une voyelle           *
rem ****************************************
t(c%)=t(c%)-2000
endif
```



```
elseif loc("BCDFGHJKLMNPQRSTVWXYZ",G$)>0
rem ****************************************
rem Acces aux consonnes                     *
rem ****************************************
l%=1
while l%<=len(l$)
if mid$(l$,l%,1)=g$
l$=left$(l$,l%-1)+mid$(l$,l%+1,20)
l%=0
rem ****************************************
rem Affectation des mises                   *
rem ****************************************
t(c%)=t(c%)+f%(e%)
if d%=1
d%=0
endif
endif
l%=l%+1
endwh
m%=1
rem ****************************************
rem traitement des chaînes                  *
rem ****************************************
if d%=0
while m%<=len(a.a$)
if mid$(a.a$,m%,1)=g$
m$=left$(m$,m%-1)+g$+mid$(m$,m%+1,20)
beep 50,150
endif
m%=m%+1
endwh
endif
endif
endif
if d%=1
c%=c%+1
endif
if c%>a%
c%=1
endif
until len(l$)=0
rem ****************************************
rem Gestion de la solution                  *
rem ****************************************
l%=loc(m$,".")
if l%
do
if c%>a%
c%=1
endif
l%=loc(m$,".")
cursor off
kstat 1
at 1,2
print el$;m$
print el$;a.b$
print el$;n$(c%),"Solution..."
rem ****************************************
rem Saisie de la solution                   *
rem ****************************************
at l%,2
cursor on
pokeb $74,0
g$=get$
cursor off
```

```
if mid$(v$,1,1)=g$
v$=right$(v$,len(v$)-1)
m$=left$(m$,l%-1)+g$+mid$(m$,l%+1,20)
beep 50,150
else
beep 500,2500
c%=c%+1
endif
until loc(m$,".")=0
endif
cursor off
at 1,2
print el$;m$
print el$;"BRAVO",n$(c%)
rem ***************************************
rem Recherche du bonus                      *
rem ***************************************
do
e%=int(rnd*9)
until e%>=1
print el$;"Bonus=",f%(e%);"F"
t(c%)=t(c%)+f%(e%)
pause 25
rem ***************************************
rem Initialisation des variables            *
rem ***************************************
l%=1
d$=""
m$=""
l$=""
v$=""
rem ***************************************
rem Calcul de la chaîne a afficher          *
rem ***************************************
while l%<a%+1
t(l%)=-t(l%)*(l%=c%)
n(l%)=n(l%)+t(l%)
t(l%)=0
d$=d$+n$(l%)+rept$(" ",8-len(n$(l%)))+gen$(n(l%),-10)+"F"+chr$(9)
l%=l%+1
endwh
rem ***************************************
rem Vider le buffer clavier                 *
rem ***************************************
pokeb $74,0
disp(1,mid$(d$,1,len(d$)-1))
r%=r%+1
cls
print chr$(0);rept$(chr$(2),14)
clock(1)
until r%=3
rem ***************************************
rem Uniquement tout les 3 mots              *
rem ***************************************
at 1,4
print el$;"Rejouer O/N"
kstat 1
pokeb $74,0
until get=%N
rem ***************************************
rem fin du programme                        *
rem ***************************************
```


Par
Alexandre
Bouillot

Du beep à Bach

L'Organiseur II a une fonction, beep pour ne pas la nommer, qui permet de générer une fréquence pendant une durée. Bien souvent, les fréquences sont définies comme ça sans chercher à faire une note précise. Pourtant, il est tout à fait possible de faire jouer de la musique à votre petit ordinateur. Pour ce faire, il faut d'abord connaître les fréquences des différentes notes de la gamme. Voici un tableau reprenant les paramètres du beep pour 8 octaves.

Attention les valeurs centrales sont les plus précises.

Note	Valeur	Note	Valeur
do	28144	do#	26566
ré	25073	ré#	23653
mi	22330	fa	21070
fa#	19892	sol	18769
sol#	17711	la	16717
la#	15774	si	14888
do	14053	do#	13260
ré	12513	ré#	11810
mi	11145	fa	10518
fa#	9924	sol	9365
sol#	8838	la	8339
la#	7869	si	7425
do	7006	do#	6610
ré	6237	ré#	5885

Suite du tableau

Valeur	Note	Valeur	Note
mi	5553	fa	5239
fa#	4943	sol	4663
sol#	4399	la	4150
la#	3915	si	3693
do	3484	do#	3286
ré	3099	ré#	2923
mi	2757	fa	2600
fa#	2452	sol	2312
sol#	2180	la	2056
la#	1938	si	1827
do	1722	do#	1623
ré	1530	ré#	1442
mi	1359	fa	1280
fa#	1206	sol	1137
sol#	1071	la	1008
la#	950	si	894
do	842	do#	792
ré	746	ré#	702
mi	660	fa	621
fa#	584	sol	549
sol#	516	la	485
la#	455	si	428
do	401	do#	377
ré	353	ré#	331
mi	308	fa	291
fa#	272	sol	255
sol#	238	la	223
la#	208	si	194
do	181	do#	169
ré	157	ré#	146
mi	136	fa	126
fa#	117	sol	108
sol#	100	la	92
la#	85	si	78

Voici deux petits programmes qui vous permettent de saisir des valeurs puis de le faire rejouer ensuite.

```
record:
local music$(10)
print "Fichier :",
input music$
create music$,a,x
print "Tempo :",
input a.x
append
do
```



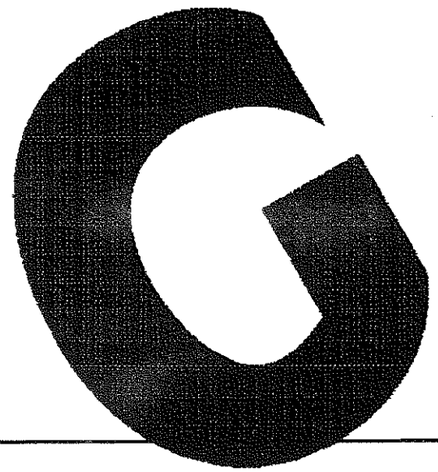
```
print "dur (0=fin) :",
      input a.x
      if a.x=0
            close
            return
      endif
      append
      print "valeur :",
      input a.x
      append
until 0<>0

play:
local music$(10),tempo,dur%,b%
print "Fichier :",
input music$
open music$,a,x
tempo=a.x
next
do
      dur%=a.x
      next
      b%=a.x
      if b%
            beep dur%*tempo,b%
      else
            pause dur%*tempo/50
      endif
until eof
```

Enfin, voici un exemple de musique pour l'Organiseur.

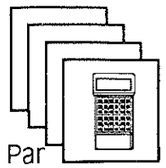
Joyeux anniversaire - tempo 39

Durée	Valeur	Durée	Valeur
4	1137	4	1137
8	1008	8	1137
8	842	16	894
4	1137	4	1137
8	1008	8	1137
8	746	16	842
4	1137	4	1137
8	549	8	660
8	842	8	894
8	1008	4	621
4	621	8	660

Ces programmes sont loin d'être la panacée mais libre à vous de les modifier, de les enrichir ou d'en faire quelque chose de complètement différent.



Gestion de temps



Par
Gilles Jean

Voici un programme qui va maintenant vous permettre de gérer avec encore plus d'efficacité votre emploi du temps. Ce logiciel permet de saisir une heure de début d'activité accompagnée d'un libellé texte (par exemple pour identifier l'activité) puis de saisir l'heure de fin de l'activité et ainsi pouvoir déterminer le temps passé. Ces informations sont stockées dans le fichier du calepin électronique (MAIN) de la mémoire interne de l'Organiseur (A:). Vous pouvez modifier le programme afin de lui affecter un programme particulier.

```
time:
local m%,t%,m$(15)
rem ******************************************
rem Titre    : TIME
rem Type     : OPL 2 lignes (CM/XP)
rem Date     : 10/02/90
rem ******************************************
rem m$=libellé aphanumérique du menu
rem m%=Valeur retournée par la fonction menu
rem t%=Buffer numérique du calcul de temps passé
rem ******************************************
open "a:main",a,hd$,hf$,lib$,ht$
rem ******************************************
rem a.hd$=heure alphanumérique de début
rem a.hf$=heure alphanumérique de fin
rem a.lib$=libellé alphanumérique
rem a.ht$=ecard alphanumérique
```



```
rem ***********************
rem initialisation du menu
rem ***********************
m$="Debut"
do
m%=menu(m$)
if m%=1
a.hd$=mid$(gen$(100+hour,3),2,2)+":"+mid$(gen$(100+minute,
3)-,2,2)
cls
print"Heure de debut"
trap edit a.hd$
rem ******************************************
rem Il est possible de quitter avec la touche ON
rem ******************************************
if err=0
rem ******************
rem Saisie du libellé
rem ******************
cls
print"Texte"
edit a.lib$
m$="Debut,Fin"
endif
elseif m%=2
cls
print"Heure de fin"
a.hf$=mid$(gen$(100+hour,3),2,2)+":"+mid$(gen$(100+minu-
te,3)-,2,2)
trap edit a.hf$
if err=0
m$=m$+",Temps"
rem ***************************
rem Calcul du temps passé
rem ***************************
t%=val(mid$(a.hf$,1,2))-val(mid$(a.hd$,1,2))
t%=t%*60
t%=abs(t%-abs(val(mid$(a.hf$,4,2))-val(mid$(a.hd$,4,2))))
a.ht$=gen$(t%,9)
rem ***************************
rem Insertion dans le fichier
rem ***************************
append
endif
elseif m%=3
cls
print"Difference temps";a.ht$,"Minutes"
get
endif
until m%=0
rem ******************************
rem Fin du programme
rem ******************************
rem A vous maintenant de
rem personnaliser ce petit
rem programme qui nous l'espéront
rem vous rendra bien des services
rem ******************************
```

Redéfinir le clavier de l'Organiseur

Par Alexandre Bouillot

Il ne faut pas beaucoup de temps à un utilisateur pour s'apercevoir qu'il lui est impossible d'écrire cette phrase :

```
Comment puis-je écrire ça avec l'Organiseur ?
```

Les accents peuvent êtres atteints sur les modèles LZ, en minuscules, en appuyant sur les touches [SHIFT] [→] simultanément puis sur la lettre à accentuer. Ainsi sont disponibles les caractères suivants :

Caractères : `àâçéèêëîïôùûü`
Touches : `abcefghijouvw.`

Malgré tout il manque quand même les apostrophes et points d'interrogation pour notre exemple. Pour ce faire, nous allons faire un tour dans la mémoire de l'Organiseur, dans la zone du clavier.

Lorsque l'on appuie sur une touche de l'Organiseur, un code, appelé "Scan code", est envoyé. Ce code est utilisé pour regarder dans une table à quel caractère correspond la touche pressée. La routine qui fait cela le fait en fonction d'un état majuscule, minuscule, numérique ainsi qu'en fonction de l'état de la touche Shift. Voici la table permettant de trouver le Scan Code en fonction du clavier :

[ON]	[MODE]	[↑]	[↓]	[←]	[→]
36	31	32	33	34	35
A	B	C	D	E	F
30	25	20	5	15	10
G	H	I	J	K	L
29	24	19	4	14	9
M	N	O	P	Q	R
28	23	18	3	13	8
S	T	U	V	W	X
27	22	17	2	12	7
[SHIFT]	[DEL]	Y	Z	[SPACE]	[EXE]
26	21	16	1	11	6

En fonction de ce code, le système de l'Organiseur va chercher dans la "Keyboard Lookup Table" pour connaître le caractère à renvoyer. En fonction des différents modificateurs, les caractères sont éventuellement convertis en majuscules ou minuscules.

Adresse	Décimales	Caractères
0	122	z
1	118	v
2	112	p
3	106	j
4	100	d
5	13	[EXE]
6	120	x
7	114	r
8	108	l
9	102	f
10	32	[SPACE]
11	119	w
12	113	q
13	107	k
14	101	e
15	121	y
16	117	u
17	111	o
18	105	i
19	99	c
20	8	[DEL]
21	116	t
22	110	n
23	104	h

Adresse	Décimales	Caractères
24	98	b
25	63	[SHIFT] *
26	115	s
27	109	m
28	103	g
29	97	a
30	2	[MODE]
31	3	[↑]
32	4	[↓]
33	5	[←]
34	6	[→]
35	1	[ON]
36	46	.
37	50	2
38	53	5
39	56	8
40	41	)
41	13	[EXE]
42	43	+
43	45	-
44	42	*
45	47	/
46	32	[SHIFT]
47	51	3

Adresse	Décimales	Caractères
48	54	6
49	57	9
50	37	%
51	48	0
52	49	1
53	52	4
54	55	7
55	40	(
56	8	[DEL]
57	58	:
58	36	$
59	34	"

Adresse	Décimales	Caractères
60	62	>
61	63	[SHIFT] *
62	59	;
63	44	,
64	61	=
65	60	<
66	2	[MODE]
67	3	[↑]
68	4	[↓]
69	5	[←]
70	6	[→]
71	1	[ON]

*la touche [SHIFT] ne renvoie jamais de code.

Lorsque l'on voit cette table, on est vite tenté d'intervenir dessus pour reprogrammer certaines touches. Malheureusement, elle est située en ROM donc non modifiable.

La solution consiste à copier la table dans une zone de la RAM, de la modifier et enfin de s'assurer que l'Organiseur va bien utiliser cette table.

En RAM, une adresse appelée BTA_TABL ($205E) contient l'adresse de départ de la table. Changer cette valeur permet d'adresser une autre table en mémoire.

Pour copier la table du clavier, nous avons besoin d'une zone mémoire stable. BTA_SBAS ($2065) contient l'adresse basse de la pile du langage. Si l'on retire à cette valeur le nombre d'octets que l'on souhaite préserver et que l'on remplace BTA_SBAS par cette valeur, cette portion de RAM ne sera pas effacée par le système de l'Organiseur. Changer la valeur de BTA_SBAS dans le cours d'un programme peut entraîner des comportements bizarres de la machine. Pour contourner le problème, la solution consiste à changer la valeur en fin de programme.

Maintenant, nous avons tous les éléments pour faire notre programme de reconfiguration. Avant tout, nous devons décider quelles sont les touches à redéfinir et fournir un moyen pour permettre de revenir au clavier standard. Dans notre exemple, nous changerons les touches < et > dans la table afin d'obtenir l'apostrophe et le point d'interrogation. Ce qui veut dire qu'en mode Alpha, la pression des touches [SHIFT] A renverra une apostrophe et [SHIFT] B renverra un point d'interrogation. A noter que sur les LZ, la touche [→] ne peut être redéfinie. La restauration du clavier original est réalisée en replaçant les valeurs BTA_TABL et BTA_SBAS à leurs valeurs originales.

Les trois programmes qui suivent réalisent les opérations suivantes :

MOVEBASE : Change la base de la pile langage afin de réserver un espace en mémoire pour faire la copie de la "Lookup table" et la valeur standard de BTA_TABL.

NEWKEY : Copie la "Lookup table" et la valeur de BTA_TABL vers la RAM sure, change les valeurs de la table et change BTA_TABL pour pointer sur la nouvelle table.

OLDKEY : Change la valeur de BTA_TABL à sa valeur originale et libère la RAM de l'ancienne table.

Pour redéfinir votre clavier, exécutez MOVEBASE puis NEWKEY. Le clavier est redéfini jusqu'à la réinitialisation de l'Organiseur ou à l'exécution de OLDKEY.

```
MOVEBASE:
POKEW $2065,PEEKW($2065)-74


NEWKEY:
LOCAL K%,TA%,SB%
K%=0
TA%=PEEKW($205E)
SB%=PEEKW($2065)
WHILE K%<72
      POKEB SB%+K%,PEEKB(TA%+K%)
      K%=K%+1
ENDWH
POKEW SB%+K%,TA%
POKEB SB%+65,39
POKEB SB%+60,63
POKEW $205E,SB%


OLDKEY:
POKEW $205E,PEEKW(PEEKW($2065)+72)
POKEW $2065,PEEKW($2065)+74
```

Attention malgré tout, les opérations sur BTA_SBAS peuvent entraîner des problèmes. Aussi, dans un premier temps, pensez à sauvegarder les données présente dans votre Organiseur.

Une procédure d'entrée

Le programme listé ci-dessous vous permet de faire saisir à l'utilisateur un mot de passe (BOSS) et de contrôler la validité de la saisie. Si vous disposez d'un écran couleur, vous pourrez, en mixant les couleurs, masquer l'affichage du code en utilisant les commandes papier et encre.

```
proc start
local boucle,mp$
note ******************************************
note Date    : 14/06/91                        *
note Objet   : Bulletin technique              *
note ce programme fonctionne sur PC4 et XCHANGE *
note ******************************************
note boucle = compteur numérique               *
note mp$ = mot de passe saisie                 *
note ******************************************
mode 0
éponge
écris car(218);repro(car(196),78);car(191);
écris car(179);repro(car(32),78);car(179);
écris car(192);repro(car(196),78);car(217);
écris au 20,0;repro(car(196),80);
écris au 21,0;repro(car(32),20);"Veuillez saisir le mot de passe - Merci"
écris au 8,32;car(218);repro(car(196),14);car(191)
écris au 9,32;car(179);" MOT DE PASSE ";car(179)
écris au 10,32;car(192);repro(car(196),14);car(217)
tantque boucle<3
que boucle=boucle+1
écris au 12,32;car(27)+car(65);"===>";
saisis mp$
note ******************************************
note Dans notre exemple, le mot de passe est   *
note BOSS (en majuscule). Vous pouvez le       *
note modifier par votre propre mot de passe    *
note ******************************************
si majus(mp$)="BOSS"
note ******************************************
note appel de votre procédure                  *
note ******************************************
stoppe
sinon
note ******************************************
note bip sonore                                *
note ******************************************
écris car(7)
fsi
ftantque
éponge
mode 1
fproc
note ******************************************
note Fin de la procédure                       *
note ******************************************
```


Réalisation : Alexandre Bouillot & Gilles Jean

MC

Système	1.29
Mobile TR	1.20
MC-link	1.29
Tableur	1.10

ORGANISEUR

CM	3.3
XP	3.6
POS200	3.6
ALPHAPOS	3.6
POS250	3.6
POS350	3.6
POS296	3.6
LZ	4.6
LZ64	4.6
POS432	4.6
POS464	4.6

Log. 2 lignes

Top Fin.	1.3
Filepak	1.4
Barpak	0.9
Tableur	2.0
Formulat.	1.0
Portfolio	1.0
Travel.	1.5
Maths.	1.0

Log. 4 lignes

Top Fin.	2.2
Filepak	2.0
Tableur	2.0
Formulat.	2.0
Prakpak	1.0

Dév.	3.01
CL PC	2.1
CL MAC	2.1

PC4

Pc4	1.1

XCHANGE

Xchange	1.37