La configuration idéale

SVM PRATIQUE

Par Julien POMMIER

Puissance, vitesse, couleur, capacité de mémoire gigantesque... Tout serait parfait sans la désastreuse gestion de la mémoire de MS-DOS. Rien n'est plus énervant que de lancer un programme banal, avec 8 Mo de mémoire interne, pour s'entendre dire "mémoire insuffisante". Cela parce que les programmes résidents réduisent à moins de 500 Ko la mémoire vive réellement disponible. D'où l'intérêt d'Autoconf, proposé par Julien Pommier. Il permet de choisir, lors du démarrage, la configuration sans disquette de lancement séparée, sans changer les fichiers Config.sys et sans réinitialiser la machine.

Notre gagnant du mois : Julien Pommier, pour son programme Autoconf.

VOUS AUSSI, VOUS POUVEZ PARTICIPER À NOTRE CONCOURS PERMANENT DE PROGRAMMATION. IL VOUS SUFFIT D'ADRESSER UN LISTING COMPLET DE VOTRE PROGRAMME, UNE DISQUETTE, UN TEXTE DESCRIPTIF ET UNE PHOTO D'IDENTITÉ À : SVM, 1, RUE DU COLONEL-PIERRE-AVIA, 75503 PARIS CEDEX 15. CHAQUE MOIS, UNE PRIME DE 2 000 F RÉCOMPENSERA LE GAGNANT.

AUTOCONF.ASM. En Assembleur. Pour IBM et compatibles.

```
CODE SEGMENT
ASSUME CS:CODE
ORG 0000h

; Données du header

driver_suiv       dw       -1
                  dw       -1
attribut dw       8004h
req               dw       offset sys_request
run               dw       offset init
nom_device        db       'NUL     '

new_device proc far
   nreq:   or        es:word ptr [bx+3],0100h
   nrun:   ret
new_device endp

fin_residant label near                        ; on perd 32 octets
                                               ; avec un dos < 3.3
req_ofs  dw ?
req_seg  dw ?

sys_request proc far
        mov       cs:[req_ofs],bx    ; mémorise l'adresse
        mov       cs:[req_seg],es    ; request header
        ret
sys_request endp

init proc far
        cld
        push      ax                           ; sauve les registres
        push      bx
        push      cx
        push      dx
        push      si
        push      di
        push      es
        push      ds
        push      cs
        pop       ds
        mov       dx,offset msg
        mov       ah,9
        int       21h
        call      init_driver
        lds       bx,cs:[dword ptr req_ofs]
        lds       si,[bx+18]                   ; adresse des paramètres
```

La version 5 de MS-DOS devrait, paraît-il, résoudre le délicat problème de la gestion de la mémoire vive au-delà de la limite actuelle des 640 Ko. En attendant, les concepteurs de cartes à mémoire et les programmeurs s'en sont donné à cœur joie pour trouver des solutions, hélas souvent incompatibles. Tel programme préfère la mémoire étendue librement accessible, tel autre la préfère dans un disque virtuel, tel autre adore la mémoire paginée. C'est le fichier *Config.sys* qui se charge d'installer la configuration de démarrage. Le hic est que celle-ci ne peut être testée ; pour en changer, il faut changer de *Config.sys*. Des programmes permettent de choisir entre différents fichiers de ce type, mais ils obligent à réinitialiser l'ordinateur. Autrement dit, au démarrage, la configuration ne peut pas être déterminée une fois pour toutes. On commence avec l'ancien fichier *Config.sys*, on lance le programme choisi, on fixe un autre *Config.sys* et on redémarre. Pas très pratique ! Une autre solution – souvent plus simple – consiste à créer plusieurs disquettes de démarrage, chacune permettant le lancement sous une configuration différente. Là, ce n'est pas très logique si l'on dispose d'un disque dur.

Autoconf évite tous ces détours : juste après les tests de démarrage (vérification de la mémoire, de la présence des disques, du clavier, etc.), il suffit d'appuyer sur la lettre attribuée à une configuration prédéfinie pour la lancer. On aura ainsi, au choix, une configuration lourde avec une multitude de programmes résidents, une configuration moyenne avec, par exemple, 512 Ko de mémoire et de bonnes possibilités d'accès aux différents programmes, une configuration "spécial Windows" avec un programme pilote spécifique pour la souris et Smartdrive, une configuration réduite libérant plus de 560 Ko, idéale pour les jeux notamment. On pourra donc adapter la configuration pour s'assurer un maximum d'efficacité.

Installer Autoconf

Autoconf se présente comme un programme pilote pour MS-DOS (c'est donc un fichier de type *.sys*), qui sera chargé *via* le fichier *Config.sys*. Evidemment, *Autoconf* doit être installé en premier afin d'être chargé en priorité. Mais on pourra placer, avant lui, et après quelques essais, d'autres programmes ou des commandes (telles que FILES = ...), qui doivent toujours être exécutées, dans toutes les configurations prévues.

Autoconf accepte trois paramètres, optionnels, pouvant être introduits dans n'importe quel ordre. Le plus utile est le délai d'attente (exprimé en secondes) entre le chargement du programme et le lancement de la configuration par défaut (si aucune n'a été retenue en tapant la lettre correspondante). Par défaut, il n'y a pas d'attente, mais il reste possible de choisir la configuration en tapant une lettre peu de temps après la mise en route. On peut choisir un délai entre zéro et neuf secondes avec le paramètre /TN, N étant un entier entre zéro et neuf. AUTOCONFIG.SYS/T2 impose ainsi une attente de deux secondes.

Le deuxième paramètre concerne l'utilisation éventuelle d'un clavier qwerty. Le programme effectue par défaut la conversion des caractères qwerty en azerty (A devient Q, Z devient W, etc.), les claviers étant normalement en qwerty au démarrage. Attention, il ne s'agit pas ici de remplacer KEYB FR, mais seulement de permettre le choix de la configuration à l'aide du clavier. En effet, *Config.sys*. étant chargé au démarrage, *Keyb.com* (appelé dans l'*Autoexec.bat*) n'a pas encore été chargé. Et comme la lettre servant à déterminer la configuration sera tapée au clavier, il faut faire en sorte que la touche A corresponde bien au choix A. Si l'on veut toutefois garder un clavier qwerty, on ajoutera le paramètre /Q. Exemple : AUTOCONFIG.SYS/T1/Q donne un délai d'attente d'une seconde et un clavier qwerty.

Le troisième paramètre, /S, correspond à une option de commande qui impose, sauf indication contraire, le redémarrage dans la dernière configuration employée. Par défaut, c'est-à-dire si ce paramètre est absent et si aucune configuration n'est choisie, le programme est lancé dans la configuration A. Si l'on se sert de /S, *Autoconf.dat,* un tout petit fichier, est créé sur le disque de démarrage (le disque n'a donc pas à être protégé).

Définir ses configurations

Les différentes configurations sont écrites à la suite les unes des autres dans le fichier *Config.sys*. Pour signaler le début d'une nouvelle configuration, on place en tête la commande DEVICE = *X, X représentant une lettre de l'alphabet (A, B, C...). La fin de la configuration (et le début de la nouvelle) sera automatiquement délimitée par la prochaine commande DEVICE = *.

```
        call    analyse_param           ; analyse les paramètres
        call    cherche_fin_config ; recherche la fin du config
        call    lit_configuration  ; lit la configuration
        call    compacte_config    ; trie les commandes
        call    clean_and_move     ; nettoie toute la zone de config.
        mov     al,cs:[config]     ; inscrit le numéro de configuration
        sub     al,'A'-1           ; dans le vecteur d'interruption 0fah
        sub     bx,bx
        mov     ds,bx
        mov     ds:[0FAh*4],al
        pop     ds
        pop     es
        pop     di
        pop     si
        pop     dx
        pop     cx
        pop     bx
        pop     ax
        ret
init endp


init_driver proc                    ; seulement si DOS < 3.3
        lds     bx,dword ptr cs:[req_ofs]
        mov     word ptr [bx+14],0 ; pour ne pas laisser le
        mov     word ptr [bx+16],cs         ; driver résident
        push    bx
        mov     ah,30h                      ; lit la version dos
        int     21h
        pop     bx
        cmp     al,3
        ja      no_tsr
        cmp     ah,21
        ja      no_tsr                      ; teste si dos supérieur à 3.21
        mov     word ptr [bx+14],offset fin_residant ; sinon, le driver
        mov     cs:[run],offset nrun        ; remplacera l'ancien driver NUL
        mov     cs:[req],offset nreq
no_tsr:
        ret
init_driver endp

ligne_suiv proc
nextc:  lodsb                               ; lit un caractère
        cmp     al,10                       ; saut de ligne ?
        jne     nextc
nextc2: lodsb                               ; boucle tant qu'il y a des sauts
        cmp     al,10
        je      nextc2
        dec     si
        ret
ligne_suiv endp

verifie_config proc
        push    si
        mov     bl,al                       ; sauve le caractère de laconfig.
v_next:
        call    ligne_suiv                  ; cherche device=*
        lodsb
        cmp     al,'D'
        jne     v_next
        lodsb
        cmp     al,'$'
        je      v_config_nok
        cmp     al,'*'
        jne     v_next
        lodsb
        cmp     al,bl
        je      v_config_ok
        jmp     v_next                      ; la config. n'existe pas
v_config_nok:
        stc
        pop     si
        ret
v_config_ok:
        clc
        pop     si
        ret
verifie_config endp

affiche_config_name proc
```

```
        mov     dx,si
acn_suiv:                           ; cherche la fin de ligne
        lodsb
        cmp     al,10
        jne     acn_suiv
        mov     ax,'$'*256 +13      ; écrit le retour à la ligne et
        xchg    [si],ax             ; le caractère $ (qui signale au dos
        push    ax                  ; de la chaîne de caractères)
        mov     ah,9
        int     21h
        pop     ax
        mov     [si],ax
        mov     si,dx
        ret
affiche_config_name endp

conv_maj proc
        cmp     al,'a'
        jb      cm_no_conv
        cmp     al,'z'
        ja      cm_no_conv
        sub     al,'a'-'A'
cm_no_conv:
        ret
conv_maj endp

find_switch proc
fs_car_suiv:
        lodsb
        cmp     al,'/'
        je      fs_switch
        cmp     al,' '
        jae     fs_car_suiv
        stc                                 ; stc si fin de la ligne
        ret
fs_switch:
        clc
        ret
find_switch endp

analyse_param proc
        push    si
ap_switch:
        call    find_switch
        jc      ap_fin
        lodsb
        call    conv_maj
        cmp     al,'T'
        jne     no_delai_sw
        lodsb
        sub     al,'0'
        cmp     al,9
        ja      ap_err
        mov     ah,18                       ; car il y a 18,2 tics d'horloge
                                    /sec.
        mul     ah
        mov     cs:[key_delai],ax
        jmp     ap_switch
no_delai_sw:
        cmp     al,'Q'
        jne     no_querty_sw
        mov     cs:[querty_clav],1
        jmp     ap_switch
no_querty_sw:
        cmp     al,'S'
        jne     ap_err
        mov     cs:[sauve_cfg],1
        jmp     ap_switch
ap_fin:
        pop     si
        ret
ap_err: mov     dx,offset sw_err_msg
        push    ds
        push    cs
        pop     ds
        mov     ah,9
        int     21h
        pop     ds
        pop     si
        ret
```

A la suite de la lettre d'appel, il est possible, sinon conseillé, de faire figurer un commentaire, qui apparaîtra au démarrage (si l'on demande à voir les configurations en tapant sur la barre d'espace). Exemple : DEVICE = *B *Configuration pour Windows* affichera B-*Configuration pour Windows*. La fin de *Config.sys* sera reconnue par le programme à la condition de placer une commande DEVICE = $ en fin de fichier. Bien sûr, *Autoexec.bat* devra, lui aussi, être lié à la configuration choisie. Pour ce faire, on a recours à *Readconf.com,* qui renvoie dans la variable *Errorlevel* un numéro correspondant à la lettre choisie au démarrage (1 pour A, 2 pour B, 3 pour C...). Il suffit alors de tester *Errorlevel* pour déterminer la configuration active et prévoir le contenu de *Autoexec.bat* en conséquence.

Exemple de lancement de fichiers à partir du disque dur C :

```
COUNTRY=033,,C:\DOS\COUNTRY.SYS¶
DEVICE=C:\LANGAGE\PGM\AUTOCONF\AUTO-
CONF.SYS /S/T1¶

DEVICE=*A Configuration par défaut¶
FILES=30¶
BUFFERS=20¶
DEVICE=C:\HIMEM.SYS¶
SHELL=C:\DOS\COMMAND.COM /P /E:200¶
DEVICE=C:\WINDOWS\MOUSE.SYS /Y¶
DEVICE=C:\DOS\RAMDRIVE.SYS 360 /e¶
INSTALL=C:\DOS\KEYB.COMFR,437,C:\DOS\KEY-
BOARD.SYS¶

DEVICE=*B Configuration pour Windows¶
FILES=40¶
BUFFERS=20¶
DEVICE=C:\HIMEM.SYS¶
SHELL=C:\DOS\COMMAND.COM /P /E:200¶
DEVICE=C:\WINDOWS\MOUSE.SYS /Y¶
DEVICE=C:\WINDOWS\SMARTDRV.SYS 300¶

DEVICE=$
```

Et le fichier *Autoexec.bat* correspondant :

```
ECHO OFF
CLS
PROMPT $p$g

C:\LANGAGE\PGM\AUTOCONF\READCONF
IF ERRORLEVEL 2 goto test2
C:\DOS\GRAFTABL
C:\UTIL\DOSEDIT
PATH=D:\;C:\DOS;C:\BAT;C:\UTIL
C:\PCTOOLS\PC-CACHE /sizext=300K
goto fin
:test2
IF ERRORLEVEL 3 GOTO test3
C:\DOS\KEYB FR,437,,C:\DOS\KEYBOARD.SYS
```

```
C:\DOS\GRAFTABL
C:\UTIL\DOSEDIT
PATH=C:\DOS;C:\BAT;C:\UTIL;C:\PCTOOLS;C:\WINDOWS
SET TEMP=C:\WINDOWS\TEMP
CD\WINDOWS
WIN\S
:fin
```

A propos du fichier *Autoexec,* rappelons que la condition est réalisée si le code *Errorlevel* est supérieur ou égal à la valeur spécifiée.
Au démarrage, on peut frapper la touche correspondant à la configuration souhaitée avant l'affichage d'un message proposant de choisir la configuration. Ce message restant à l'écran durant le laps de temps indiqué lors du lancement d'*Autoconf*, on peut alors frapper soit une autre touche (si on a changé d'avis, par exemple), soit la barre d'espace pour obtenir la liste des configurations disponibles. La frappe d'une lettre entraîne le lancement de la configuration correspondante.
Il arrive que l'ordinateur se "plante" : le programme travaillant sur les données de MS-DOS, il ne fonctionne peut-être pas avec des versions du système d'exploitation non signées de Microsoft. Il a été testé avec succès sur MS-DOS 4.0, 3.3, 3.2 et devrait fonctionner sur toutes les versions de MS-DOS. Mais une erreur peut aussi se produire dans des cas particuliers, par exemple si la configuration est vide (aucune directive FILE, aucun DEVICE) ou encore sur des appels erronés effectués dans *Config.sys* ou dans *Autoexec.bat*.

Les programmes pilotes

Les drivers permettent de piloter toute une série de périphériques (imprimantes, claviers, écrans, disques...). Et d'autres, que l'on ajoute en écrivant les programmes pilotes correspondants (disques virtuels, par exemple). Il s'agit en fait d'extensions de MS-DOS et du BIOS. MS-DOS contient des drivers utilisés par défaut (disques, écran...).
Ils se divisent en deux catégories : périphérique "caractère" (imprimante...) ou "bloc" (disques). Chacun recèle un *Device header,* au début du fichier, qui indique le type du driver au système d'exploitation, avec le format :
offset 0 : pointeur long vers le prochain driver
offset 4 : attribut du driver
offset 6 : pointeur sur la routine de requête
offset 8 : pointeur sur la routine d'exécution
offset 10 : nom du driver (8 octets)
L'attribut décrit les caractéristiques du programme pilote. Celui utilisé ici n'étant qu'un artifice permettant le chargement d'un programme dans le fichier *Config.sys*, on peut remplacer le driver NUL de MS-DOS (chargé par défaut). La routine de requête sera appelée avant la routine d'exécution à

```
analyse_param endp

cherche_fin_config proc
        push    si
        call    ligne_suiv
cfc_suiv:
        call    ligne_suiv
        cmp     word ptr [si],'$D'
        jne     cfc_suiv
        inc     si
        inc     si
        mov     cs:[fin_config],si
        pop     si
        ret
cherche_fin_config endp

liste_configuration proc
        push    si
list_suiv:
        call    ligne_suiv
        lodsb
        cmp     al,'D'                  ; => device
        jne     list_suiv
        lodsb
        cmp     al,'$'                  ; fin ?
        je      fin_list
        cmp     al,'*'
        jne     list_suiv
        lodsb
        mov     dl,al
        mov     ah,2
        int     21h                     ; affiche la lettre
        mov     dl,'-'
        int     21h
        call    affiche_config_name     ; affiche la configuration
        jmp     list_suiv       ; correspondante

fin_list:
        pop     si
        ret
liste_configuration endp

lit_ecrit_cfg proc
        push    ds
        push    cs
        pop     ds
        mov     ah,19h                  ; retourne unité active dans al
        int     21h                     ; (0 -> A,1 -> B)
        add     al,'A'
        mov     bx,offset cfg_file_name
        mov     [bx],al
        mov     dx,offset cfg_file_name
        or      cx,cx
        jne     lit_cfg

        mov     ah,3Ch                  ; création du fichier
        xor     cx,cx                   ; pas d'attribut particulier
        int     21h
        jc      lec_fin
        mov     bx,ax                   ; handle du fichier
        mov     ah,40h                  ; écriture
        jmp     lec_exec

lit_cfg:
        mov     ax,3D00h          ; ouverture
        int     21h
              jc        lec_fin
        mov     bx,ax
        mov     ah,3Fh                  ; lecture

lec_exec:
        mov     dx,offset config        ; depuis la variable config
        mov     cx,1                    ; d'un octet
        int     21h
        mov     ah,3Eh                  ; fermeture
        int     21h
lec_fin:
        pop     ds
        ret
lit_ecrit_cfg endp
```

```
lit_configuration proc
   ;- attente de la pression d'une touche
         mov      ah,0
         int      1Ah                        ; charge dans dx le nombre de tops
attend_touche:                               ; d'horloge (18,2 par seconde)
         mov      ah,1
         int      16h                        ; teste la présence d'un
                                             caractère
         jnz      lit_config_car
         mov      ah,0
         push     dx
         int      1Ah                        ; lit l'heure actuelle
         mov      cx,dx
         pop      dx
         sub      cx,dx
         cmp      cx,cs:[key_delai]          ; le délai s'est-il écoulé ?
         jb       attend_touche
         jmp      config_defaut              ; config par défaut -> 'A' ou
                                             ; dans autoconf.dat
lit_config_car:
         sub      ah,ah                      ; lit la touche
         int      16h
         cmp      al,13                      ; la touche entrée permet de
                                             ; charger la configuration
         jne      lconf_no_entre             ; précédente
config_defaut:
         cmp      cs:[sauve_cfg],0           ; lire dans autoconf.dat
         jz       lconf_fin
         mov      cl,1
         call     lit_ecrit_cfg              ; fonction de lecture
         jmp      lconf_fin

lconf_no_entre:
         cmp      al,' '
         jne      no_config_list             ; affichage de la liste
         call     liste_configuration        ; des configurations
         jmp      lit_config_car

no_config_list:
         cmp      cs:[querty_clav],1 ; utilisation d'un clavier qwerty?
         je       no_convert
         push     ds
         push     si
         call     azerty_convert             ; non : transformation
                                             ; des caractères comme Z,A,M…
         pop      si
         pop      ds
no_convert:
         call     conv_maj                   ; convertit en majuscules
         cmp      al,'A'
         jb       lconf_err          ; ce n'est pas une lettre
         cmp      al,'Z'
         ja       lconf_err

         call     verifie_config             ; la lettre est-elle valide ?
         jnc      config_ok

   ;- affiche le message d'erreur
lconf_err:
         mov      dx,offset err_msg
         push     ds
         push     cs
         pop      ds
         mov      ah,9h
         int      21h
         pop      ds
         jmp      lconf_fin
config_ok:
         mov      cs:[config],al             ; sauve le numéro de
                                             ; configuration
         cmp      cs:[sauve_cfg],0
         je       lconf_fin
         xor      cx,cx                      ; fonction d'écriture
         call     lit_ecrit_cfg              ; inscrit la configuration sur
                                               le disque
lconf_fin:
         ret
lit_configuration endp
```

chaque fois que MS-DOS utilise le périphérique. Cette routine sert à mémoriser l'adresse du *Request header*, bloc de données permettant à MS-DOS de dialoguer avec le driver. Ce bloc débute toujours par la même séquence :
offset 0 : longueur du *Request header*
offset 1: longueur de la sous-unité (un driver bloc peut gérer plusieurs périphériques)
offset 2 : numéro de la fonction demandée
offset 3 : code de retour
offset 5 : 8 octets réservés
offset 13 : données variables selon la fonction requise
Les fonctions pouvant être demandées au driver :
0 : init
1 : media check (pour un bloc seulement)
2 : buid BPB (Bios Parameter Bloc, pour un bloc)
3,4,5,6,7 : fonctions de lecture
8,9,10,11,12 : fonctions d'écriture
13 : open device
14 : close device
15 : test disque (bloc)
La fonction utilisée ici est essentiellement *Init* : c'est elle qui permet de prendre le contrôle et de modifier la configuration. Pour le *Request header*, la structure doit être complétée comme suit :
offset 13 : nombre d'unités (pour un bloc)
offset 14 : adresse de fin du driver
offset 18 : pointeur sur le BPB (bloc seulement)
offset 22 : numéro du disque (bloc)
Le programme utilise l'adresse de fin du driver pour signifier au système d'exploitation qu'il ne désire pas rester en mémoire : il suffit d'y inscrire l'adresse de début du driver. Cependant, avec MS-DOS 3.21 et les versions précédentes, le programme pilote ne peut quitter la mémoire : il doit rester résident, et remplit une fonction précise. La solution : le substituer au NUL de MS-DOS pour qu'il n'occupe que 32 octets de mémoire. Pour cela, on l'intitule NUL, on positionne le bit 2 de l'attribut (spécifique à NUL) et on attend l'appel à la fonction *Init*, qui est toujours la première appelée. Il n'y a pas de test et la fonction d'exécution se résume à la seule fonction *Init*. Avec MS-DOS 3.21, ou une version antérieure, la routine d'exécution ne fait rien (c'est un RETF) et la routine de requête se contente de renvoyer une information d'absence d'erreur (MS-DOS fait la même chose).

Structure de Config.sys une fois chargé

Config.sys est entièrement chargé en mémoire avant l'exécution du moindre *Device driver*. Il y subit un semblant de compilation : les commandes sont remplacées par des lettres, suivies des paramètres, et les codes convertis en majuscules. Ainsi, DEVICE=AUTOCONF.SYS devient DAUTOCONF.SYS. De même, BUFFERS=30 devient B30. Le programme saura qu'il a atteint la fin du fichier *Config.sys* lorsqu'il trouvera les caractères D$ (device =$).

La difficulté consiste ici à ne faire exécuter que les "bonnes" commandes du *Config.sys* chargé en mémoire. Avec MS-DOS 4, il est possible de transformer les commandes inutiles en commentaires, en utilisant en début de commande la lettre O qui signale un commentaire. Mais cela est impossible sur les versions antérieures de MS- DOS, qui ignorent les REM. La solution consiste à effacer directement en mémoire les commandes non utilisées. Comme la version ainsi corrigée doit occuper exactement la même place en mémoire, la partie inutilisée est ajoutée dans la ligne d'appel de *Autoconf* (collée aux paramètres, avec tous les caractères de saut de ligne effacés) et les commandes conservées sont déplacées à la fin de *Config.sys*.
Lorsqu'on utilise le programme avec l'option /s, un fichier particulier, *Autoconf.dat*, apparaît dans la racine du disque. Il ne contient qu'un seul caractère : la dernière configuration utilisée.

SOUS-PROGRAMMES EMPLOYÉS DANS LE LISTING

INIT est le programme principal. Il sauve les registres, appelle d'autres sous-programmes et écrit dans le vecteur d'interruption 0FAh le numéro de la configuration demandée.
INIT_DRIVER teste la version de MS-DOS pour savoir si une partie du programme pilote doit ou non résider en mémoire.
LIGNE_SUIV a pour paramètre DS:SI, lequel pointe sur une ligne de *Config.sys*. Au retour, DS:SI pointe sur le début de la ligne suivante.
VERIFIE_CONFIG a pour paramètres DS:SI, qui pointe sur une ligne de *Config.sys*, et AL, qui contient la lettre de la configuration demandée. Le sous-programme vérifie l'existence de ce caractère.
AFFICHE_CONFIG_NAME affiche les commentaires qui suivent les directives DEVICE=*.
CONV_MAJ convertit en majuscules le caractère contenu dans AL.
FIND_SWITCH cherche le prochain caractère "/" dans DS:SI.
ANALYSE_PARAM analyse les paramètres /T, /Q, /Q
CHERCHE_FIN_CONFIG recherche la fin de la configuration. Si la ligne DEVICE=$ est absente, la machine se "plantera" très certainement.
LISTE_CONFIGURATION fournit la liste des configurations disponibles.
LIT_DISQUE_ACTIF cherche le disque de démarrage et initialise CFG_FILE_NAME en conséquence.
SAUVE_DERNIERE_CFG sauve la dernière configuration active dans un fichier *Autoconf.dat*.
CHARGE_DERNIERE_CFG va lire la référence de la dernière configuration dans *Autoconf.dat*.
LIT_CONFIGURATION lit le caractère désignant la configuration. Il teste d'abord la disponibilité d'un caractère pendant KEY_DELAYS tops d'horloge puis charge, selon le cas, la configuration par défaut ou celle demandée. Il affiche la liste des configura-

```
compacte_config proc
        mov     cs:[dans_config],1 ; tout ce qui suit autoconf est
                                   ; exécuté jusqu'au
        call    ligne_suiv                 ; premier device=*
        mov     cs:[debut_ligne],si
        mov     cs:[prem_ligne],si
compacte_lsuiv:
        lodsb                              ; charge le premier octet de la
                                             liste :
        cmp     al,'D'                   ; le type de fonction (D = device)
        jne     no_special_cmd
        lodsb
        cmp     al,'$'
        jz      fin_compacte               ; on atteint la fin du config.sys
        cmp     al,'*'
        jne     no_special_cmd
        lodsb                              ; c'est donc le début
                                           ; d'un nouveau bloc
        mov     cs:[dans_config],0 ; de configuration
        cmp     al,cs:[config]
        jne     no_special_cmd

        mov     cs:[dans_config],1 ; config = 1 si c'est le bon
        call    affiche_config_name        ; affichage du nom de la
                                             configuration
        jmp     copie_ligne                ; on efface cependant la
                                           ; directive device=*
no_special_cmd:
        cmp     cs:[dans_config],0 ; est-ce que l'on garde
                                   ; les commandes
        jne     nomodif                    ; pointées par si ?

copie_ligne:                               ; non :
        mov     di,cs:[debut_ligne]        ; on recopie la ligne suivante
        call    ligne_suiv                 ; sur la zone pointée par
                                           ; debut_ligne
        mov     cx,cs:[fin_config]
        sub     cx,si
        push    ds
        pop     es
        rep     movsb
        mov     si,cs:[debut_ligne]        ; réinitialise si au début de la
                                             ligne
        jmp     compacte_lsuiv

nomodif:                           ; on est dans la bonne configuration
        call    ligne_suiv
        mov     cs:[debut_ligne],si        ; réactualise debut_ligne
        jmp     compacte_lsuiv
fin_compacte:
        ret
compacte_config endp

clean_and_move proc
        mov     si,cs:[debut_ligne]        ; pointe sur la fin des données
raccorde:                          ; conservées
        dec     si                         ; cherche la fin de la ligne
                                             précédente
        cmp     byte ptr [si],10   ; pour supprimer les sauts de ligne
        je      raccorde
        cmp     byte ptr [si],13
        je      raccorde
        mov     di,cs:[fin_config]
        dec     di
        mov     cx,si
        sub     cx,cs:[prem_ligne] ; déplacement des données
                                   ; conservées du début du
        add     cx,3                       ; config vers la fin
        std                                ; en inversant la direction
                                           ; (si et di
        rep     movsb                      ; diminuent)
        cld
        mov     cx,cs:[prem_ligne]
        sub     cx,3
cleanall:                          ; remplissage de la zone
                                   ; inutilisée par des espaces
        mov     byte ptr [di],' '          ; (c'est plus propre
        dec     di                         ; et supprime les sauts de ligne)
        cmp     di,cx
        ja      cleanall
```

```
        ret
clean_and_move endp

azerty_convert proc
        push    ax
        push    cs
        pop     ds
        mov     si,offset convert_tab
test_suiv:
        lodsb                           ; charge un scan code
        inc     si
        or      al,al                   ; teste la fin du tableau
        jz      fin_convert
        cmp     ah,al                   ; compare les scan codes
        jne     test_suiv
        dec     si
        lodsb                           ; charge le code ascii
        inc     sp                      ; pas de pop ax
        inc     sp
        ret
fin_convert:
        pop     ax
        ret
azerty_convert endp

config          db 'A'  ; mémorise la configuration demandée (A par défaut)
key_delai dw 0    ; attente d'un caractère en 18e de sec.
dans_config     db ?
debut_ligne     dw ?
prem_ligne      dw ?
fin_config      dw ?                    ; pointe sur la fin de la config
querty_clav     db 0                    ; 0 => azerty par défaut
sauve_cfg db 0                  ; 0 => pas de sauvergarde par
                                        défaut
convert_tab     db 010h,'A'             ; conversions azerty vers qwerty
                db 011h,'Z'
                db 01Eh,'Q'
                db 02Ch,'W'
                db 033h,'M'
                db 0      ; fin du tableau

cfg_file_name   db 'A:\AUTOCONF.DAT',0

err_msg  db  7,'Erreur : configuration inexistante',13,10,'$'
sw_err_msg      db 7,'autoconf.sys : mauvaises options',13,10,'$'
msg     db
'_____________________________________________________________________',
13,10
        db      '_                AUTOCONF.SYS : configurations multiples
_',13,10
       db      '_                                    1991   S.V.M.
_',13,10
        db      '_        Appuyez sur espace pour avoir la liste des
          configurations        _',13,10
        db
'_____________________________________________________________________',
13,10,'$'

CODE ENDS
END
```

READCONF. ASM. Routine de récupération du numéro de configuration

```
code segment
     assume cs:code
org  100h

start:
     sub   ax,ax
     mov   ds,ax
     mov   al,ds:[byte ptr 0FAh*4]
     mov   ah,4Ch
     int   21h
code ends
end  start
```


tions si l'on a appuyé sur la barre d'espace.
COMPACTE_CONFIG est au cœur du programme. Son rôle : regrouper toutes les lignes de la configuration demandée vers le début de la zone mémoire réservée au stockage de *Config.sys*.
CLEAN_AND_MOVE complète la procédure précédente. Elle déplace toutes les informations regroupées vers la fin de la zone de *Config.sys* et fait passer la zone restant libre pour des paramètres (non pris en compte) du programme *Autoconf*.
AZERTY_CONVERT convertit les caractères qwerty en azerty grâce au tableau CONVERT_TAB.

DONNÉES ET MISE EN ŒUVRE DU PROGRAMME

CONFIG contient la lettre de la configuration.
KEY_DELAY contient le délai d'attente maximal du caractère de configuration.
DANS_CONFIG : utilisée par COMPACTE_CONFIG pour savoir si DS:SI pointe sur la configuration demandée ou sur une autre.
DÉBUT_LIGNE : utilisée par COMPACTE_CONFIG pour mémoriser l'endroit où sera recopiée la prochaine ligne appartenant à la configuration demandée.
PREM_LIGNE pointe sur le début de la zone du *config.sys* conservée.
FIN_CONFIG pointe sur la fin de *Config.sys* (avant le compactage).
QWERTY_CLAV : 0 si azerty, 1 si qwerty (paramètre /Q).
SAUVE_CFG : 1 si paramètre /S.
CONVERT_TAB : tableau formé des scan codes des touches à convertir en azerty, suivis des codes ASCII correspondant (scan code de "Q" avec code ASCII de "A"...).
On entre les deux programmes sous les noms *Autoconf.asm* et *Readconf.asm* à l'aide de tout éditeur capable de sauvegarder en ASCII. On devra ensuite les compiler avec MASM (version 4.0 ou supérieure) ou avec TASM 1.0.
Pour compiler *Autoconf.asm*, inscrire :
MASM AUTOCONF¶
LINK AUTOCONF¶
EXE2BIN AUTOCONF AUTOCONF.SYS¶
Et pour compiler *Readconf.asm*, inscrire :
MASM READCONF¶
LINK READCONF¶
EXE2BIN READCONF READCONF.COM¶

Le signe ¶ indique un retour à la ligne.

Ce programme est disponible en téléchargement sur notre serveur **36 15 SVM**. *Il est accompagné d'un exemple comprenant un fichier Autoexec.bat et Config.sys pour démarrer sur une disquette système placée dans le lecteur A.*