
C T E I BP 41 83220 LE PRADET
RC Toulon 85 A 745 RM Toulon 332550060830 N° Siret 332 550 060 00013
Code APE 2914 Banque Populaire Le Pradet 88 21 026 958

MODE D'EMPLOI DE PASCAL09

INTRODUCTION

Comme expliqué dans la documentation de présentation du compilateur PASCAL09, la notice de ce compilateur est constituée par ce document d'une part mais aussi et surtout par l'ouvrage "PASCAL: manuel de l'utilisateur" par Jensen et Wirth publié chez Eyrolles.
Le compilateur PASCAL09 est un compilateur complet c'est à dire qu'il traduit directement et une fois pour toutes un programme PASCAL en code machine 6809. Pour accroitre sa souplesse d'emploi, cette traduction n'est pas directe puisque le compilateur ne délivre pas un fichier binaire mais au contraire un fichier source d'assemblage qu'il vous faut ensuite assembler grace à la commande ASMB du DOS. Cette façon de faire présente deux avantages notables: elle est très pédagogique car elle vous permet de voir comment les instructions PASCAL sont traduites en langage machine 6809; elle vous permet aussi d'améliorer un programme une fois celui ci compilé, soit par optimisation des codes générés, soit par adjonction de vos propres sous programmes.
Ce compilateur respecte d'aussi près que possible les directives du manuel précité, manuel qui a été écrit par les deux personnes à l'origne de la création du langage PASCAL. La présente notice va donc surtout etre consacrée à la présentation de l'utilisation des disquettes fournies et à la mise en évidence des différences qui existent entre PASCALO9 et le PASCAL décrit dans l'ouvrage de Jensen et Wirth.

LES DISQUETTES FOURNIES

Compte tenu de la taille et du nombre de fichiers utilisés par PASCAL09, deux disquettes vous sont fournies en 40 pistes et une seule en 80 pistes. Dans le premier cas, si vous disposez de lecteurs 40 pistes double face ou double densité ou si vous avez par ailleurs un lecteur 80 pistes, nous vous recommandons de copier les deux disquettes 40 pistes simple face fournies sur une seule et même disquette; cela vous facilitera la mise en oeuvre de PASCALO9. Si vous n'avez que des lecteurs simple face, simple densité, 40 pistes, vous serez obligés de manipuler les disquettes.
DANS TOUS LES CAS, FAITES AU MOINS UNE COPIE DE SAUVEGARDE DE LA OU DES DISQUETTES CI JOINTES POUR EVITER TOUT PROBLÈME EN CAS DE FAUSSE MANOEUVRE DE VOTRE PART.

Le langage PASCAL utilise de façon intensive un certain nombre

de caractères ASCII particuliers tels que les accolades {} et

les crochets []. Ces caractères ne sont pas disponibles sur votre système TAV09 si vous utilisez les caractères accentués français; en effet, la norme en ce domaine veut que les accolades soient remplacées respectivement par ° et 5 tandis que les crochets sont remplacés par é et è. Vous avez donc deux possibilités: soit supporter ce remplacement qui rend les listings PASCAL à peine moins lisibles, soit remplacer votre générateur de caractères accentués GCGGA09 par un GCGG09. Si vous avez une imprimante, nous vous conseillons de conserver votre générateur de caractères accentués, vos listings seront ainsi un peu moins lisibles à l'écran mais, en contrepartie, vous pourrez continuer à faire du traitement de texte avec minuscules accentuées. Lorsque vous voudrez un listing correct, vous pourrez le faire sortir sur votre imprimante en configurant celle ci en caractères USA.

Si votre imprimante est une EPSON configurée en caractères accentués français, nous vous rappelons qu'il est inutile de modifier la position de ses micro_ interrupteurs pour passer en mode USA. Il vous suffit de faire, avant de faire un listing PASCAL, un P HECHO lB,52,00 et elle passera alors en mode USA. Pour revenir au mode accentué, il vous suffira de faire un P HECHO lB,52,0l.

COMILATION ET EXECUTION DES PROGRAMMES

GENERALLTES:

La ou les disquettes ci jointes supportent un certain nombre de fichiers dont voici les fonctions.
- PASCAL.CMD est le programme chargé de lancer la compilation par appels successifs des différents morceaux du compilateur NPASCAL.BIN, NPASS1;BIN à NP6809.BIN.
- NPASCAL.BIN, NPASS1.BIN, NPASS2.BIN...NP6809.BIN sont les différents fichiers qui composent le compilateur‘ proprement dit. Ils sont appelés lorsque c'est nécessaire par PASCAL.CMD et effectuent la compilation de votre fichier source PASCAL.
- PRUN.CMD est le programme qui permet de lancer l'exécution de n'importe quel programme PASCAL une fois que celui ci a été compilé et assemblé (voir ci après). Il contient tous les sous programmes nécessaires pour que votre programme PASCAL compilé puisse tourner avec TAVDOS09 sans problème.
- PREFIX.TXT est un fichier texte qui contient toutes les déclarations de procédures PASCAL de l'environnement standard d'un programme. Ce fichier est automatiquement ajouté au début de tout programme PASCAL lors de la compilation. Des précisions à son sujet vous sont données par ailleurs dans ce manuel.
- PASCDEF.TXT est un fichier source d'assemblage qui est appeléautomatiquement par un LIB lors de l'assemblage du résultat de la compilation d'un programme PASCAL. Ce fichier contient des définitions nécessaires à l'interfacage entre votre programme PASCAL, PRUN.CMD et le DOS.
- Un certain nombre d'autres fichiers sources PASCAL sont

également présents sur la ou les disquettes ci jointes. Ces fichiers sont décrits dans le chapitre "Quelques exemples de programmes".

COMMENT COMPILER ET EXECUTER UN PROGRAMME PASCAL ?

Pour faire exécuter un programme PASCAL il faut:
- L'écrire avec, par exemple, l'éditeur EDIT du DOS
- Le compiler avec la commande PASCAL.CMD
- Assembler le fichier ainsi obtenu avec la commande ASMB du DOS
- Faire exécuter le programme avec la commande PRUN.

Lorsque la procédure compilation, assemblage a été exécutée une fois, le fichier binaire obtenu peut ensuite etre exécuté autant de fois que vous le souhaitez par simple utilisation de PRUN.

La syntaxe de la commande de compilation est la suivante:

+++PASCAL Fichier source,Fichier d'assemblage,+options

Fichier source est le nom que vous avez donné au fichier contenant votre programme source PASCAL. Fichier d'assemblage est le nom du fichier qui sera créé par le compilateur pour recevoir le résultat de sa compilation. Si aucun nom n'est donné, le même nom que celui de Fichier source sera utilisé mais avec l'extension ASM (pour rappeler que c'est un fichier à assembler). Si aucun numéro de lecteur n'est précisé, le lecteur de travail est utilisé par défaut. Les options sont facultatives; elles sont matérialisées par, une ou plusieurs lettres placées les unes à la suite des autres, dans n'importe quel ordre, sans espace ou symbole d'aucune sorte entre elles. Leurs significations sont les suivantes:

- B Permet d'interdir au compilateur de générer du code machine
- L Fait afficher un listing source de compilation sur le terminal du système
- Y Efface automatiquement tout fichier d'assemblage de même nom que celui demandé
- C Supprime un certain nombre de controles automatiques tels que les valeurs extrèmes des scalaires ou des pointeurs
- N Fait générer au compilateur les informations nécessaires pour qu'en cas d'erreur lors de l'exécution du programme, le numéro exact de la ligne qui en est responsable puisse etre indiqué.
- S Fait afficher certaines informations relatives à la compilation en fin de listing
- Q Termine la compilation dès qu'une erreur est détectée.

Pendant la phase de compilation, deux fichiers temporaires sont créés puis automatiquement effacés ensuite. Ces deux fichiers sont placés sur le disque de travail qui doit donc avoir de la place disponible. La place nécessaire est à peu près égale à deux fois la taille du fichier source à compiler exprimée en nombre de secteurs.

Après la phase de compilation, le fichier appelé ci avant fichier d'assemblage doit etre assemblé grace à ASMB. Nous vous renvoyons si nécessaire à la notice de l'assembleur pour cela car cette phase des opérations ne présente aucune difficulté. Rappelons seulement que le fichier PASCDEF.TXT, fourni sur la ou les disquettes ci jointes, doit résider sur la disquette de travail puisque l'assembleur y fera appel grace a un LIB inclus dans le fichier d'assemblage.
Si ces opérations se sont déroulées sans erreur, vous devez alors etre en possession d'un fichier muni d'une extension BIN, fichier qui pourra maintenant etre exécuté toutes les fois que vous le souhaiterez grace à la commande PRUN en utilisant la syntaxe suivante:

+++PRUN Fichier.BIN,Paramètres,<Fichier d'entrée,>Fichier de sortie.

Fichier.BIN est le nom de votre programme PASCAL après compilation et assemblage tandis que Paramètres, Fichier d'entrée et Fichier de sortie sont des informations facultatives qui peuvent etre exploitées par votre programme comme expliqué ci après.

En temps normal, les informations lues par un programme PASCAL le sont à partir du fichier par défaut INPUT qui correspond en fait au clavier du système. De même, les informations écrites par le programme PASCAL le sont dans le fichier par défaut OUTPUT qui n'est autre que l'écran de votre système. Le fait de spécifier un nom de fichier précédé de < fait lire les informations demandées par le programme PASCAL dans ce fichier plutot qu'à partir du clavier. De même, le fait de spécifier un nom de fichier précédé de > fait écrire les informations fournies par le programme PASCAL dans ce fichier plutot que sur l'écran.

Pour ce qui de "Paramètres", toutes informations à ce sujet vous sont fournies dans le chapitre suivant.

FONCTIONS STANDARDS NON DISPONIBLES SUR PASCAL09

Afin de mettre en évidence de la façon la plus efficace qui soit les différences entre PASCAL09 et la définition de PASCAL faite par Niklauss et Wirth, nous allons, ci après, faire directement référence aux pages du manuel cité en introduction. Si l'édition de ce dernier en votre possession n'est pas la même que la notre, il se peut que ces numéros de pages diffèrent d'une ou deux unités. Cela ne devrait toutefois vous poser aucun problème compte tenu des sujets abordés.

PASCAL09 admet, pour les mots réservés, l'utilisation de majuscules ou minuscules indiféremment. Cependant, pour ce qui est des fonctions, procédures et constantes contenues dans PREFIX.TXT, les noms utilisés doivent etre scrupuleusement respectés (majuscule pour majuscule et minuscule pour minuscule).

Page 7 du manuel, il est question d'identificateurs. PASCAL09 autorise jusqu'à 160 caractères significatifs dans le nom d'un identificateur. Les majuscules et minuscules sont autorisées ainsi que le caractère "souligné" (_) sauf en guise de premier caractère d'un identificateur. '
Page 12 du manuel, il est dit que dès qu'un au moins des opérandes d'une expression contenant +, −, / ou ∗ est réel, l'expression est considérée comme réelle. Ceci n'est pas admis par PASCAL09. Cela signifie que si un entier est utilisé comme réel dans une expression, la fonction CONV(X) doit etre utilisée explicitement pour le convertir en réel. Ainsi, le programme suivant serait faux:

```
VAR x,y: integer;
z: real;
BEGIN
Z:= x/y
END;
```

Il faudrait le modifier de la façon suivante:

```
z:= CONV(x) / CONV(y);
```

Les programmes PASCAL ne nécessitent pas l'en tete PROGRAM car celle ci est incluse dans le fichier PREFIX qui est automatiquement ajouté avant tout programme PASCAL par le compilateur. Le listing de PREFIX vous est par ailleurs fourni en annexe de ce manuel. '
Pages 14 et 29 il est fait allusion à l'instruction GOTO. Cette dernière n'est pas disponible sur PASCAL09 car elle est contraire à la notion de programme structuré à laquelle le langage PASCAL doit conduire. Ceci est d'ailleurs confirmé par le texte du bas de la page 30 qui demande de restreindre l'emploi de GOTO à des ruptures d'algorithmes. Pour cela PASCAL09 vous propose ABORT présenté dans un chapitre suivant.
Pour communiquer avec des fichiers autres que les fichiers par défaut INPUT et OUTPUT, la commande PRUN supporte l'utilisation d'un tableau d'articles constitué de chaines de caractères. Ce tableau est appelé PARAM et la chaine de caractères ID. Le premier paramètre de la ligne de commande peut ainsi etre trouvé dans PARAM[1].ID, le deuxième dans PARAM[2].ID et ainsi de suite. Il est possible d'utiliser jusqu'à 5 paramètres, chacun ayant au maximum 16 caractères de long. Par exemple, si l'on veut fournir

à un programme PASCAL le nom de fichier l.DONNEES.TXT, il faut faire:

```
+++PRUN Programme.BIN,1.DONNEES.TXT
```

Le programme PASCAL (Programme.BIN dans cet exemple) pourra accéder au nom de fichier de la façon suivante:

```
VAR nom_fich: array [1..l6] of char;
fich_ok: boolean; v
BEGIN
```

```
...
IF PARAM[1].ID = nom_fich THEN fich_ok = true;
...
END;
```

De plus amples précisions à propos de ce type d'utilisation peuvent etre obtenues en examinant le contenu du fichier PREFIX dont le listing vous est fourni en annexe ainsi que les programmes de démonstration décrits au chapitre "Quelques exemples de programmes".
Page 48 est présenté le type ensemble (SET). PASCAL09 peut admettre jusqu'à 128 éléments dans un ensemble. Les positions de ces éléments sont numérotées de 0 à 127 ce qui signifie qu'un ensemble d'entiers ne peut contenir de nombre supérieur à 127 ou inférieur à 0. Pour cette même raison, un ensemble de réels n'est pas admis.
Le PASCAL standard autorise l'imbrication des déclarations de procédures et de fonctions. PASCAL09 ne supporte pas une telle imbrication ce qui interdit de déclarer une procédure dans une autre procédure. Cette limitation n'est pas vraiment contraignante et permet de réaliser une compilation beaucoup plus efficace.
Un tableau de caractères doit toujours contenir un nombre pair d'éléments (toujours afin d'améliorer l'efficacité de la compilation).
La procédure standard DISPOSE(p) n'est pas implémentée.

Les procédures PACK et UNPACK ne sont pas supportées mais les tableaux compactés sont autorisés.
Les procédures standards EOF et EOLN nécessitent l'indication explicite du nom de fichier concerné, même si ce dernier est le fichier INPUT. On écrira ainsi EOF(INPUT) par exemple.

POSSIBILITÉS PARTICULIÈRES

Un certain nombre de possibilités offertes par PASCAL09 ne figurent pas dans le manuel PASCAL de Jensen et Wirth ou permettent de réaliser des fonctions plus étendues que celles initialement prévues. C'est ce que nous allons vous présenter ci après.
Les commentaires, dans le programme source PASCAL, peuvent etre inclus entre les deux accolades classiques {} ,entre les couples parenthèse-étoile (* *) ou entre des guillemets " ". Des commentaires ne peuvent etre imbriqués les uns dans les autres (ce qui est tout de même assez rare) en utilisant les mêmes délimiteurs.

Les chaines de caractères qui sont écrites explicitement en PASCAL sont toujours terminées par un caractère blanc (ou espace). Ainsi si vous définissez la chaine 'ABC', ce sera un groupe de 4 caractères (A, B, C et l'espace de fin "ajouté" par PASCAL09). Ce caractère blanc doit etre pris en compte lors des comparaisons de chaines de caractères. Ainsi:
```
BEGIN
```

....
IF PARAM[1].ID = ’1.PASCALPG.TXT ’ THEN

est un exemple qui fonctionne grace au blanc qui suit PASCALPG.TXT. En effet, rappelons que PARAM[X].ID est une chaine de 16 caractères. Il faut donc, compte tenu de l’espace ajouté par PASCAL09, faire suivre 1.PASCALPG.TXT par un espace pour arriver à une chaine de la même taille. Si la longueur de la chaine n’était pas correcte, le compilateur génèrerait un message d’erreur.
Les directives RESET et REWRITE ont été améliorées pour pouvoir supporter un nom de fichier. Ainsi peut on écrire:

```
VAR f:file of integer;
BEGIN
REWRITE (f,PARAM[2].ID);
ou
RESET (f,’INDATA.TXT’);
```

Pour les fichiers texte, le fichier utilisé peut s’appeler ’ME.’. Dans ces conditions, toutes les entrées/sorties faites sur .ce fichier sont en fait envoyées sur l’ensemble écran-clavier du système. Dans ce cas, vous pouvez faire un RESET avant de faire des entrées/sorties de la façon suivante:

```
VAR text_file TEXT;
BEGIN
RESET (text_file, ’ME’);
```

Cela aura pour effet d’associer le fichier texte au terminal du système.
Tous les fichiers doivent etre déclarés dans le bloc global. Les fichiers locaux à une procédure ne sont pas admis.
La plage de variation des entiers va de -32768 à +32787 en utilisant 16 bits. Les nombres réels quant à eux vont de 1.0E-38 à 1.0E+38 (ou des mêmes nombre précédés du signe moins pour les nombres négatifs). Tous les caractères ASCII de code compris entre 0 et 127 inclus peuvent etre utilisés avec PASCAL09.
Les nombre hexadécimaux sont admis à condition d’etre précédés du symbole classique dollar ($). Ces nombres doivent etre des entiers dont la valeur maximale est FFFF (codage sur 16 bits).

Le code ASCII d’un caractère peut etre placé au sein d’une chaine de caractères de la façon suivante (:XX:) où XX est le code. Ainsi:
’HELLO (:33:)’ est équivalent a ’HELLO !’ puisque 33 est le code ASCII exprimé en décimal de !.
Les tableaux et les articles compactés sont supportés cependant, compte tenu de l'utilisation en interne de mots de 16 bits pour coder les données, PACKED ne change pas la représentation de ces derniers. Cela justifie l’absence des procédures PACK et UNPACK;
Le type ALFA a été inclus dans les types standardisés. Sa description est un tableau de l0 caractères comme ceci:

```
TYPE ALFA = array [1..10] of char; :
```

Les procédures WRITE et WRITELN ne peuvent utiliser les fonctions ABS et SQR directement comme arguments car elles doivent connaitre par avance le type du résultat, ce qui est impossible avec SQR et ABS puisque cela dépend de l'argument de ces fonctions; ainsi:
WRITE (SQR(X)) est il incorrect.

PROCÉDURES ET FONCTIONS SPECIALES

Un certain nombre de procédures et fonctions ont été ajoutées au PASCAL initial. Leurs descriptions peuvent etre trouvées dans le listing de PREFIX fourni en annexe mais voici quelques informations quant à leur utilisation.

Les routines SYSTEM_DRIVE et WORK_DRIVE fournissent tout simplement les numéros de lecteurs système et travail de façon à pouvoir manipuler facilement des noms de fichiers complets.

La routine ABORT permet à un programme PASCAL09 de se terminer de façon anormale. Le numéro de la dernière ligne exécutée préalablement à ABORT et une information d'état sont fournis au programme demandeur ou au terminal du système si c'est le programme principal qui s'est ainsi terminé.

Les procédures PEEK, POKE, PEEKW et POKEW permettent d'accéder à des adresses mémoires absolues comme en Basic. PEEK fournit le contenu de l'octet adressé tandis que POKE écrit un octet à l'adresse spécifiée. PEEKW et POKEW font de même mais avec un mot de 16 bits.
Les routines __GET à _SREWRITE (voir listing de PREFIX en annexe) sont utilisées par le sytème pour les entrées/sorties de fichiers. Toute routine de PREFIX commençant par _ ou __ ne doit pas etre utilisée par vos programmes.
Les procédures BUFFER et UNBUFFER sont utilisées pour les entrées/sorties via le terminal du système. UNBUFFER arrete le mécanisme de "bufférisation" pour tout fichier associé au terminal du système. Si le fichier n'est pas associé au terminal, UNBUFFER est sans effet. UNBUFFER est pratique lors des entrées de caractères, lorsque vous devez répondre par un caractère seulement à une question par exemple. BUFFER rétablit le fonctionnement normal suite à UNBUFFER.
La fonction RND est un générateur de nombres aléatoires compris entre 0 et 1. Elle permet de générer un nombre aléatoire dans n'importe quelle plage avec la formule suivante:
NOMBRE_ALEATOIRE:: (MAX-MIN) * RND(0.0) + MIN; MAX et MIN étant les valeurs extrèmes désirées.
L'argument utilisé par RND fonctionne comme en Basic, à savoir:
- S'il est inférieur à 0, une nouvelle série de nombres aléatoires est lancée. Pour toutes les valeurs différentes de X, une séquence différente sera initialisée mais chaque fois que X prendra une valeur déjà utilisée, la même séquence que celle précédemment lancée sera reproduite.
- S'il est nul, un nouveau nombre aléatoire est généré à chaque

appel de la fonction (utilisation normale).
- S'il est positif, la fonction fournit le dernier nombre aléatoire précédemment généré.

PROGRAMATION AVANCÉE

La commande PRUN, utilisée pour faire exécuter un programme PASCAL rappelons le, supporte l'appel d'autres programmes PASCAL ou de programmes en assembleur considérés comme sous programmes du programme PASCAL principal à exécuter.

Le principe général de fonctionnement est le suivant: lorsque le programme PASCAL appelé en tant que sous programme est terminé, une information relative à la dernière ligne exécutée et au déroulement de l'exécution est fournie au programme principal. Comme cela, ce dernier peut savoir exactement ce qui s'est passé dans le programme appelé. Voici un exemple d'utilisation:

```
VAR prgm, data_file: IDENTIFIER;
parms=ARGLIST; (*les types IDENTIFIER et ARGLIST sont
définis dans PREFIX*)
nuligne:INTEGER; «
motif: PROGRESULT; (Xdéfini aussi dans PREFIX*)

BEGIN

prgm:= 'QSORT '; (*appel du programme QSORT*)
data_file:= '1.DATA.TXT '; {*nom du fichier de données*)

parms[l].ID:= data_file; (*liste de paramètres*)
RUN (prgm,parms,nuligne,motif);
(*test de présence d'erreurs*)
IF motif=TERMINATED THEN _(*fin normale du programme,
voir PREFIX*)
```

Dans cet exemple, le programme QSORT.BIN est appelé est exécuté. La variable motif du type PROGRESULT reflète l'état de l'exécution de ce programme comme vous pouvez le constater en examinant le listing de PREFIX fourni en annexe ainsi que les ' exemples décrits dans le chapitre suivant.
Un programme ainsi appelé est toujours chargé depuis le disque de travail avec l'extension BIN. Tous les fichiers éventuellement ouverts par le programme appelant restent ouverts et peuvent etre utilisés par le programme appelé. Ceci est valable même pour INPUT et OUTPUT. Par contre, tous les fichiers utilisés en propre par le programme appelé son fermés lors de la fin de celui ci. Un programme appelé peut lui même appeler d'autres programmes selon le même procédé.
Un programme peut passer à un autre programme des paramètres du type IDENTIFIER, integer ou boolean. Ceci permet de se constituer une bibliothèque des sous programmes les plus fréquemment utilisés et de ne les appeler ensuite que lorsque c'est nécessaire dans un programme déterminé.
Bien qu'un tel mode d'utilisation du PASCAL09 ne présente pas de difficulté majeure, nous vous recommandons d'y faire appel qu'une

fois que vous maitriserez bien le langage PASCAL et son utilisation sur votre système TAVO9.
L'appel de sous programmes en assembleur est également possible et repose sur un principe similaire à celui présenté ci avant cependant, comme il fait appel à des notions de programmation en langage machine assez évoluées, nous ne conseillons son utilisation qu'aux programmeurs expérimentés.
Le programme assembleur appelé doit respecter quatre contraintes. Il doit etre écrit en code translatable et ne faire référence à aucune adresse mémoire absolue; les variables doivent donc toutes etre référencées par utilisation du pointeur de pile. Les registres Y et U ne doivent pas etre modifiés par le programme. Les deux premiers octets du programme doivent contenir la longeur totale du programme exprimée en nombre d'octets, y compris les deux premiers. Enfin, le programme doit restituer une pile "propre" lors de la fin de son exécution.
Les deux premières contraintes sont simples à respecter puisque ce sont des contraintes de programmation faciles à mettre en oeuvre en assembleur 6809. La troisième signifie qu'il faut placer au début du programme un FDB suivi par le nombre total d'octets+2 (ceux du FDB lui même) que comporte le programme. La dernière condition enfin est remplie par l'appel de deux sous programmes contenus dans PRUN selon l'exemple ci après.

Cet exemple peut etre utilisé comme un "squelette" de programme assembleur appelé par un programme PASCAL; il vous suffit de le remplir avec vos propres instructions pour que cela soit directement utilisable. 'Par ailleurs, les programmes DEMOASM et GETDATE décrits dans le paragraphe suivant sont également de bons guides pour ce faire.

```
FDB TAILLE nombre d'octets du programme
START EQU 0003 adresse de START dans PRUN
TERM EQU 0009 adresse de TERM dans PRUN
DEBUT JSR START préparation de la pile
FDB O,l2,0,0 données nécessaires pour START
PSHS U,Y sauvegarde de U et Y
* votre programme peut commencer ici en sachant que
* les FCB de INPUT, OUTPUT et de la liste de paramètres
* peuvent etre référencés via Y grace aux offsets suivants
* l6,Y adresse du FCB de INPUT
* 14,Y adresse du FCB de OUTPUT
* l2,Y adresse du début de la liste de paramètres
PULS U,Y récupération de U et Y
JSR TERM remise en état de la pile
TAILLE EQU *
END
```

Une représentation du contenu de la pile après l'appel d'un programme en PASCAL ou en assembleur vous est fournie en annexe. Remarquez que les adresses des FCB évoquées ci avant sont également disponibles sur la pile de même que celle de début de la liste de paramètres.

QUELQUES EXEMPLES DE PROGRAMMES

Afin de vous faciliter la prise de contact avec PASCAL09, un certain nombre de programmes PASCAL vous sont fournis sous forme de fichiers sources sur la ou les disquettes ci jointes. La majorité d'entre eux sont extraits du manuel de Jensen et Wirth.

Ce sont:

COSINUS.TXT	qui est le programme 4.5 page 22
GRAPH2.TXT	qui est le programme 6.2 page 36
COMPLEX.TXT	qui est le programme 7.1 page 42
SETOP.TXT	qui est le programme 8.1 page 49
PREMIERS.TXT	qui est le programme 8.2 page 52
MINMAX3.TXT	qui est le programme 11.2 page 68
PARCOURS.TXT	qui est le programme 11.5 page 75
EXPON2.TXT	qui est le programme 11.8 page 79
PGCDRECU.TXT	qui est le programme 11.9 page 80

Ces fichiers étant des fichiers source PASCAL, il est évidemment nécessaire de les compiler puis de les assembler pour pouvoir les faire exécuter.
Trois autres programmes, non contenus dans le manuel de Jensen et Wirth, sont également disponibles sur la ou les disquettes ci jointes.
XREF.TXT est un programme PASCAL (à compiler et à assembler donc) qui sert à générer des références croisées (cross reference) d'un programme source PASCAL. C'est à dire qu'il est capable de vous fournir, sous forme de tableau classé par ordre alphabétique, les numéros de lignes d'apparition de tous les mots rencontrés dans un programme PASCAL. Il s'utilise (une fois compilé et assemblé répétons le) de la façon suivante:

+++PRUN XREF,NOM DE FICHIER SOURCE PASCAL,+L

et fait donc lister sur l'écran du système tous les mots avec leurs numéros de ligne d'apparition du programme PASCAL concerné. L'option +L interdit l'affichage du listing du programme PASCAL avant la sortie du tableau. Si vous voulez autoriser cet affichage, il suffit d'enleVer le +L de la ligne de commande ci dessus.
DEMOASM.TXT est un exemple de programme PASCAL qui appelle un programme en assembleur, le programme en assembleur étant fourni dans le fichier GETDATE.TXT. Ces deux programmes sont donc à utiliser comme guides si vous souhaitez appeler des programmes assembleur à partir de programmes PASCAL. L'exemple fourni fait afficher, par le programme assembleur, un message fourni par le programme PASCAL et, réciproquement, le programme assembleur va lire la date du DOS et la fournit au programme PASCAL qui l'affiche en clair sur l'écran.

UTILISATION DE L'IMPRIMANTE

Comme pour toutes les commandes du DOS, PASCAL ou PRUN permettent d'utiliser l'imprimante du système. Il suffit de

précéder PASCAL ou PRUN du P traditionnel.
Il est également possible de faire appel à cette dernière directement à partir d'un programme PASCAL selon la procédure d'appel d'un programme assembleur et en utilisant le fichier PRINT.SYS du DOS qui contient les sous programmes de pilotage de l'imprimante. Cette façon de faire est toutefois assez délicate et nous ne la conseillons qu'aux programmeurs très expérimentés.

```
(*****************************************************************)
(*      S T A N D A R D  P R E F I X                              *)
(*****************************************************************)

CONST LINELENGTH = 132;
TYPE LINE = ARRAY [1..LINELENGTH] OF CHAR;

CDNST IDLENGTH = 16;
TYPE IDENTIFIER ARRAY [1..IDLENGTH] OF CHAR;

TYPE
 TEXT = FILE OF CHAR;
 text = TEXT;

CONST MAXSTR = 10; {Length of Alfa strings }
TYPE
 ALFA = ARRAY [1..MAXSTR] OF CHAR;
 alfa = ALFA;

TYPE PROGRESULT =
 (TERMINATED, OVERFLOW, POINTERERROR, RANGEERROR, VARIANTERROR,
HEAPLIMIT, STACKLIMIT, ABORTED);

TYPE ARGTAG =
(NILTYPE, BOOLTYPE, INTTYPE, IDTYPE);

TYPE ARGTYPE = RECORD
         CASE TAG: ARGTAG OF
          NILTYPE, BOOLTYPE: (BOOL: BOOLEAN);
          INTTYPE: (INT: INTEGER);
          IDTYPE: (ID: IDENTIFIER)
         END;

CONST MAXARG = 5;
TYPE ARGLIST = ARRAY [1..MAXARG] OF ARGTYPE;

PROCEDURE MARK(VAR TOP: INTEGER);
PROCEDURE RELEASE(TOP: INTEGER);

PROCEDURE RUN(ID: IDENTIFIER; VAR PARAM: ARELIST;
                 VAR LINE: INTEBER; VAR RESULT: PRDBRESULT);

PRDCEDURE SYSTEM_DRIVE(VAR C: CHAR);
PROCÉDURE WORK_DRIVE(VAR C: CHAR);

PRDCEDURE ABORT; "TERMINATE PROGRAM IMMEDIATELY"
```

```
(*
PROCEDURES USED TO MANIPULATE ABSOLUTE MEMORY LOCATIONS.
*)

FUNCTIDN PEEK(LOC: INTEBER): INTEGER; (* READ BYTE *)
FUNCTIÜN PEEKW(LOC: INTEBER): INTEGER; (* READ WORD *)

PROCEDURE POKE(LOC, VAL: INTEGER); '(* WRITE BYTE *)
PROCEDURE POKEW(LOC, VAL: INTÉEGER); (* WRITE WORD *)

FUNCTION ODD(X: INTEGER): BOOLEAN;
FUNCTION ROUND(X: REEL): INTEGER;

(*****************************************************************)
(*      FILE I/O DEFINITIONS                                     *)
(*****************************************************************)

PRDCEDURE __GET(VAR F: TEXT);
PROCEDURE __PUT(VAR F: TEXT);

PROCEDURE __RDX(VAR C: CHAR);
PROCEDURE __WRX(C: CHAR);

FUNCTION EOLN(VAR F: UNIV TEXT): BOOLEAN;
FUNCTION ÉOF(VäR F: UNIV TEXT): BOOLEAN;

PROCEDURE __RLN;
PROCEDURE __WLN;

PROCEDURE __RWF(VâR F: TEXT; DUHMY1, DUMMY2: INTEGER);
PROCEDURE __RWFS(VñR F: TEXT); { SHDRT FDRM DF RNF }
PROCEDURE __EIO;

PROCEDURE __RDI(VAR I: INTEGER; WIDTH, DIGITS: INTEGER);
PROCEDURE __RDC(VAR C: CHAR; WIDTH, DIGITS: INTEGER);
PROCEDURE __RDR(VAR R: REAL; WIDTH, DIGITS: INTEGER);

PROCEDURE NRI(I: INTEGER; NIDTH, DUHMY: INTEGER);
PROCEDURE NRC(C: CHAR; HIDTH, DUMMY: INTEBER);
PROCEDURE NRS(S: LINE; NIDTH, DUMMY: INTEGER);
PROCEDURE NRR(R: REAL; WIDTH, DIGITS: INTEGER);

PROCEDURE _FRESET(SIZE: INTEGER; NAME: LINE);
PROCEDURE _FREWRITE(SIZE: INTEGER; NAME: LINE);

PRDCEDURE _SRESET;
PROCÉDURE _SREWRITE;

PROCEDURE BUFFER(VAR F: TEXT); (* TURN BUFFÉRINB ON *)
PROCEDURE UNBUFFER(VAR F: TEXT); (* TURN BUFFERING OFF *)

PROCEDURE PAGE(VAR F: TEXT); (* OUTPUT FORM FEÉD *)
PROCEDURE SETBIN(VAR F: TEXT); {Set binary file mode}
```

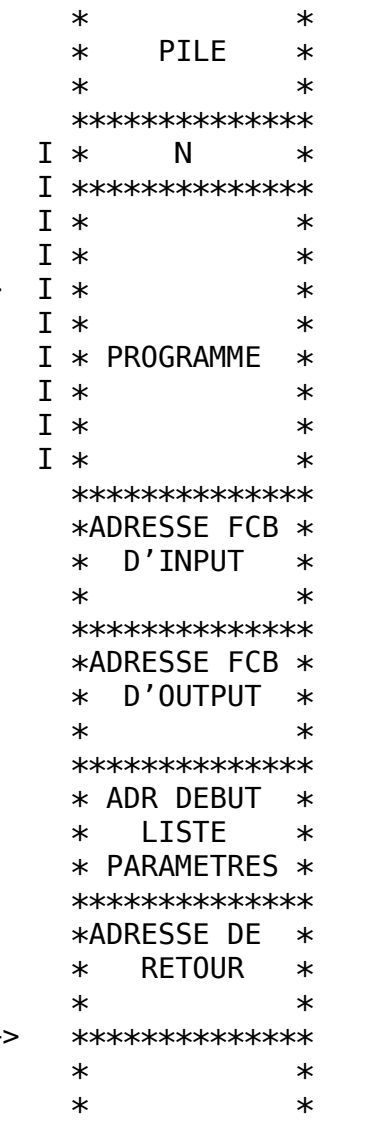
```
(*****************************************************************)
(*      STANDARD FUNCTIDNS (TRIC, ETC)                           *)
(*****************************************************************)

FUNCTION SIN(X: REAL): REAL;
FUNCTION COSCX: REAL): REAL;
FUNCTION ARCTAN(X: REAL): REAL;

FUNCTION EXP(X: REAL): REAL;
FUNCTION LN(X: REAL): REAL;

FUNCTION SQRT(X: REAL): REAL;
FUNCTION RND(X: REAL): REAL;

PROGRAM P(VAR INPUT, OUTPUT: TEXT; VAR PARAM: ARGLIST);
```



ALLURE DE LA PILE APRES UN APPEL DE PROGRAMME