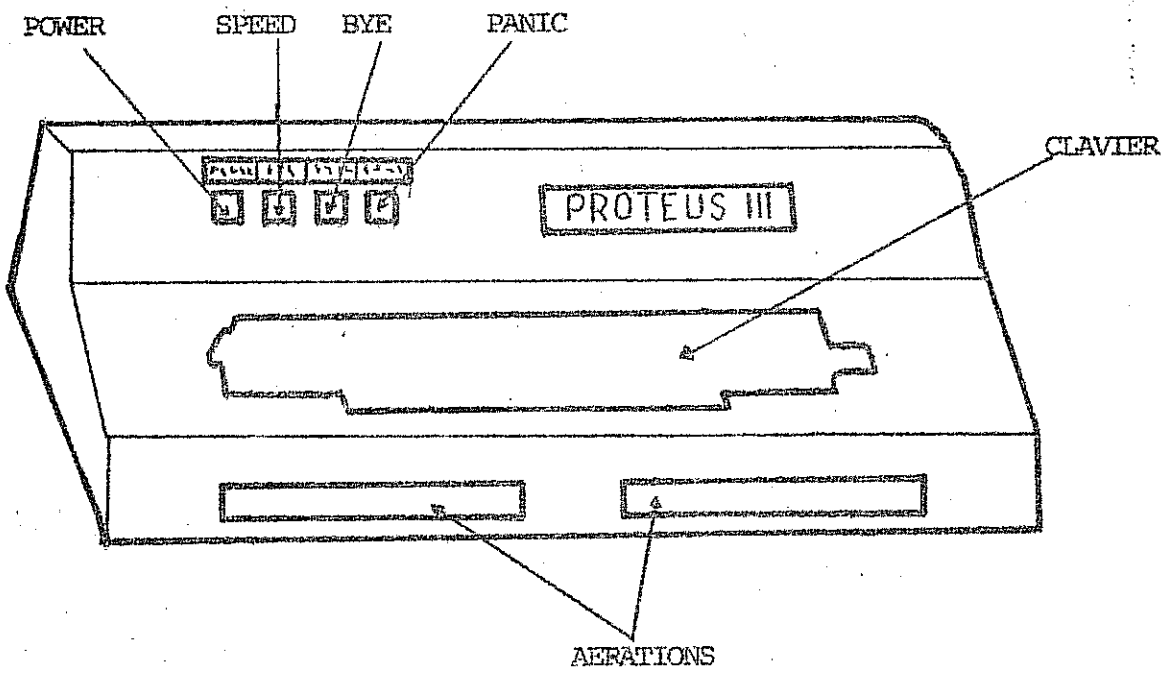
SOMMAIRE

DESCRIPTION PHYSIQUE du PROTEUS III



Sur la face avant, sont regroupées les diverses commandes permettant à l'utilisateur de contrôler le déroulement du programme.

LE CLAVIER:

Ce dernier sert soit à l'écriture du programme soit à la saisie de données en cours de programme.

Il est possible, à partir de ce clavier, d'obtenir tous les caractères alphanumériques classiques.

En frappant normalement sur le clavier, vous obtiendrez des caractères majuscules.

Les minuscules peuvent s'obtenir en appuyant simultanément sur la touche HERE IS et sur la touche correspondant au caractère désiré. A l'aide de cette touche ou de son emploi simultané avec la touche SHIFT, il est également possible d'obtenir des caractères spéciaux permettant à l'utilisateur par exemple d'encadrer du texte.

Les caractères de contrôle (effacement écran, effacement d'un caractère, effacement d'une ligne, etc) s'obtiennent en appuyant simultanément sur la touche CTRL et sur une des touches du clavier.

En appuyant simultanément sur la touche SHIFT et sur une des touches du clavier, vous obtiendrez les caractères de "HAUT de TOUCHE" du clavier.

Vous trouverez ci-après un tableau regroupant tous les caractères pouvant s'obtenir à partir du clavier ainsi que la procédure d'obtention du dit caractère.

Les deux colonnes supplémentaires référencées: Code Héxadécimal et Code Décimal servent à connaître la valeur numérique assimilée au caractère demandé.

Dans le cas où l'on travaille en Basic, il faut considérer la valeur décimale des caractères alphanumériques; leur valeur héxadécimale ne servant qu'en travail de langage machine.

VOIR TABLEAU PAGE SUIVANTE

La touche BREAK sert à interrompre le programme utilisateur en cours; ce dernier n'est bien sûr pas altéré.

Cette procédure est différente d'un CTRL C, en effet le programme ne peut être relancé à l'aide de la commande CONT, il faudra donc utiliser la commande RUN.

Les quatre touches situées en haut à gauche de l'appareil servent respectivement à :

POWER: Cet interrupteur provoque la mise en marche ou l'arrêt de l'appareil ainsi que son initialisation.

SPEED: Il est possible de connecter à votre PROTEUS III plusieurs types d'imprimantes, ces dernières se caractérisent principalement par leur vitesse de transmission.
Deux vitesses sont sélectionnables à l'aide de cet interrupteur.
D'origine, ces deux vitesses sont:
30 Caractères par seconde
120 Caractères par seconde

Ces deux vitesses sont modifiables au gré de l'utilisateur par déplacement d'une connection à l'intérieur de l'appareil.

Pour procéder à cette modification des deux vitesses de transmission, suivez la procédure ci-jointe:

1°) Assurez-vous que toutes les prises de l'appareil sont bien débranchées (secteur, magnétophone, etc...)
2°) Retournez l'appareil de manière à avoir les vis de fermeture de l'appareil vers vous, et dévissez ces dernières.
3°) Retournez de nouveau votre PROTEUS III et basculez avec précaution l'ensemble carrosserie et clavier vers l'arrière
4°) Sur la partie avant droite du circuit imprimé vous trouverez un composant portant la référence MC 14411 avec, à coté, une série de pastilles servant à la sélection de la vitesse de transmission de l'imprimante.

TABLEAU des CARACTERES AFFICHABLES

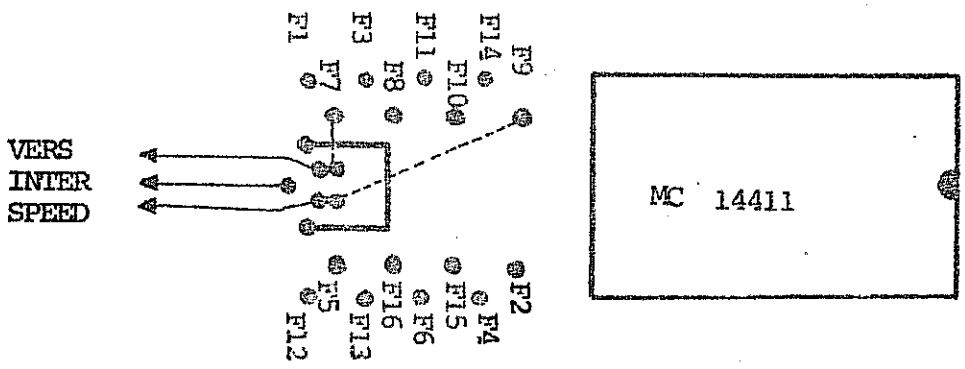
CARACTERE	HEX	DEC	OBTENTION	COMMENTAIRE
@	40	64	SHIFT,P	Caractère normal.
A	41	65	A	Alphabet majuscule.
B	42	66	B	Alphabet majuscule.
C	43	67	C	Alphabet majuscule.
D	44	68	D	Alphabet majuscule.
E	45	69	E	Alphabet majuscule.
F	46	70	F	Alphabet majuscule.
G	47	71	G	Alphabet majuscule.
H	48	72	H	Alphabet majuscule.
I	49	73	I	Alphabet majuscule.
J	4A	74	J	ALphabet majuscule.
K	4B	75	K	Alphabet majuscule.
L	4C	76	L	ALphabet majuscule.
M	4D	77	M	Alphabet majuscule.
N	4E	78	N	Alphabet majuscule.
O	4F	79	O	Alphabet majuscule.
P	50	80	P	ALphabet majuscule.
Q	51	81	Q	Alphabet majuscule.
R	52	82	R	Alphabet majuscule.
S	53	83	S	Alphabet majuscule.
T	54	84	T	Alphabet majuscule.
U	55	85	U	Alphabet majuscule.
V	56	86	V	Alphabet majuscule.
W	57	87	W	Alphabet majuscule.
X	58	88	X	Alphabet majuscule.
Y	59	89	Y	Alphabet majuscule.
Z	5A	90	Z	Alphabet majuscule.
[	5B	91	SHIFT,K	Caractère normal.
\	5C	92	SHIFT,L	Caractère normal.
]	5D	93	SHIFT,M	Caractère normal.
↑	5E	94	SHIFT,N	Caractère normal.
—	5F	95	SHIFT,O	Caractère normal.
	20	32		Alphabet majuscule.
!	21	33	SHIFT,1	Alphabet majuscule.
"	22	34	SHIFT,2	Alphabet majuscule.

CARACTERE	HEX	DEC	OBTENTION	COMMENTAIRE
#	23	35	SHIFT,3	Caractère normal.
$	24	36	SHIFT,4	Caractère normal.
%	25	37	SHIFT,5	Caractère normal.
&	26	38	SHIFT,6	Caractère normal.
'	27	39	SHIFT,7	Caractère normal.
(	28	40	SHIFT,8	Caractère normal.
)	29	41	SHIFT,9	Caractère normal.
x	2A	42	SHIFT,:	Caractère normal.
+	2B	43	SHIFT,;	Caractère normal.
,	2C	44	'	Caractère normal.
-	2D	45	-	Caractère normal.
.	2E	46	.	Caractère normal.
/	2F	47	/	Caractère normal.
Ø	30	48	Ø	Caractère numérique.
1	31	49	1	Caractère numérique.
2	32	50	2	Caractère numérique.
3	33	51	3	Caractère numérique.
4	34	52	4	Caractère numérique.
5	35	53	5	Caractère numérique.
6	36	54	6	Caractère numérique.
7	37	55	7	Caractère numérique.
8	38	56	8	Caractère numérique.
9	39	57	9	Caractère numérique.
:	3A	58	:	Caractère normal.
;	3B	59	;	Caractère normal.
<	3C	60	SHIFT,,	Caractère normal.
=	3D	61	SHIFT,-	Caractère normal.
>	3E	62	SHIFT,.	Caractère normal.
?	3F	63	Shift,/	Caractère normal.
	C0	192	HERE IS,	Caractère graphique.
a	C1	193	HERE IS,A	Alphabet minuscule.
b	C2	194	HERE IS,B	Alphabet minuscule.
c	C3	195	HERE IS,C	Alphabet minuscule.
d	C4	196	HERE IS,D	Alphabet minuscule.
e	C5	197	HERE IS,E	Alphabet minuscule.

CARACTERE	HEX	DEC	OBTENTION	COMMENTAIRE
f	C6	198	HERE IS,F	Alphabet minuscule.
g	C7	199	HERE IS,G	Alphabet minuscule.
h	C8	200	HERE IS,H	Alphabet minuscule.
i	C9	201	HERE IS,I	Alphabet minuscule.
j	CA	202	HERE IS,J	Alphabet minuscule.
k	CB	203	HERE IS,K	Alphabet minuscule.
l	CC	204	HERE IS,L	Alphabet minsucule.
m	CD	205	HERE IS,M	Alphabet minuscule.
n	CE	206	HERE IS,N	Alphabet minuscule.
o	CF	207	HERE IS,O	Alphabet minuscule.
p	D0	208	HERE IS,P	Alphabet minsucule.
q	D1	209	HERE IS,Q	Alphabet minuscule.
r	D2	210	HERE IS,R	Alphabet minuscule.
s	D3	211	HERE IS,S	Alphabet minuscule.
t	D4	212	HERE IS,T	Alphabet minuscule.
u	D5	213	HERE IS,U	Alphabet minuscule.
v	D6	214	HERE IS,V	Alphabet minuscule.
w	D7	215	HERE IS,W	Alphabet minuscule.
x	D8	216	HERE IS;X	Alphabet minuscule.
y	D9	217	HERE IS,Y	Alphabet minuscule.
z	DA	218	HERE IS,Z	Alphabet minuscule.
√	DB	219	HERE IS + SHIFT,K	Caractère normal.
⌊	DC	220	HERE IS + SHIFT,L	Caractère graphique.
≠	DD	221	HERE IS + SHIFT,M	Caractère normal.
~	DE	222	HERE IS + SHIFT,N	Caractère normal.
≃	DF	223	HERE IS + SHIFT,O	Caractère normal.
▯	A0	160	HERE IS,	Caractère graphique.
▬	A1	161	HERE IS + SHIFT,1	Caractère graphique.
▭	A2	162	HERE IS + SHIFT,2	Caractère graphique.
▭	A3	163	HERE IS + SHIFT,3	Caractère graphique.

CARACTERE	HEX	DEC	OBTENTION	COMMENTAIRE
	A4	164	HERE IS + SHIFT,4	Caractère graphique.
	A5	165	HERE IS + SHIFT,5	Caractère graphique.
	A6	166	HERE IS + SHIFT,6	Caractère graphique.
	A7	167	HERE IS + SHIFT,7	Caractère graphique.
	A8	168	HERE IS + SHIFT,8	Caractère graphique.
	A9	169	HERE IS + SHIFT,9	Caractère graphique.
	AA	170	HERE IS + SHIFT,:	Caractère graphique.
	AB	171	HERE IS + SHIFT,;	Caractère graphique.
	AC	172	HERE IS,,	Caractère graphique.
	AD	173	HERE IS,-	Caractère graphique.
	AE	174	HERE IS,.	Caractère graphique.
	AF	175	HERE IS,/	Caractère graphique.
	BO	176	HERE IS,Ø	Caractère graphique.
	B1	177	HERE IS,1	Caractère graphique.
	B2	178	HERE IS,2	Caractère graphique.
	B3	179	HERE IS,3	Caractère graphique.
	B4	180	HERE IS,4	Caractère graphique.
	B5	181	HERE IS,5	Caractère graphique.
	B6	182	HERE IS,6	Caractère graphique.
	B7	183	HERE IS,7	Caractère graphique.
	B8	184	HERE IS,8	Caractère graphique.
	B9	185	HERE IS;9	Caractère graphique.
	BA	186	HERE IS,:	Caractère graphique.
	BB	187	HERE IS,;	Caractère graphique.
	BC	188	HERE IS + SHIFT, ,	Caractère graphique.
	BD	189	HERE IS + SHIFT, -	Caractère graphique.
	BE	190	HERE IS + SHIFT,.	Caractère graphique.

CARACTERE	HEX	DEC	OBTENTION	COMMENTAIRE
	BF	191	HERE IS + SHIFT,/	Caractère graphique
	1	1	CTRL,A	Caractère de contrôle (retour du curseur en haut à gauche).
	5	5	CTRL,E	Caractère de contrôle (retour chariot avec effacement de fin de ligne).
	6	6	CTRL,F	Caractère de contrôle (retour chariot sans effacement de fin de ligne).
	8	8	CTRL,H	Caractère de contrôle (retour d'une position en arrière).
	9	9	CTRL,I	Caractère de contrôle (avance d'une position).
	A	10	CTRL,J	Caractère de contrôle (descente du curseur d'une position).
	B	11	CTRL,K	Caractère de contrôle (montée du curseur d'une position).
	C	12	CTRL,L	Caractère de contrôle (effacement écran avec retour curseur en haut à gauche).
	D	13	CTRL,M	Caractère de contrôle (retour chariot avec effacement fin de ligne).
	F	15	CTRL,O	Caractère de contrôle (retour du curseur d'une position en arrière).
	18	23	CTRL,X	Caractère de contrôle (effacement de la ligne courante du curseur).
	19	24	CTRL,Y	Caractère de contrôle (rotation de la page visualisée).
	1A	25	CTRL,Z	Caractère de contrôle (effacement de la ligne courante du curseur).



D'origine, les connections de sélection de vitesse de transmission sont envoyées sur les pastilles F7 et F9.

Pour modifier les vitesses de transmission, déplacez les deux connections allant vers F7 et F9 en vous reportant au tableau ci-joint:

PASTILLE	VITESSE de TRANSMISSION
F1	960 Caractères/seconde
F2	720 Caractères/seconde
F3	480 Caractères/seconde
F4	360 Caractères/seconde
F5	240 Caractères/seconde
F6	180 Caractères/seconde
F7	120 Caractères/seconde
F8	60 Caractères/seconde
F9	30 Caractères/seconde
F10	20 Caractères/seconde
F11	15 Caractères/seconde
F12	13,5 Caractères/seconde
F13	11 Caractères/seconde
F14	7,5 Caractères/seconde

Les deux pastilles restantes F15 et F16 permettent de disposer d'un signal d'horloge pouvant servir à synchroniser des éléments extérieurs au système:

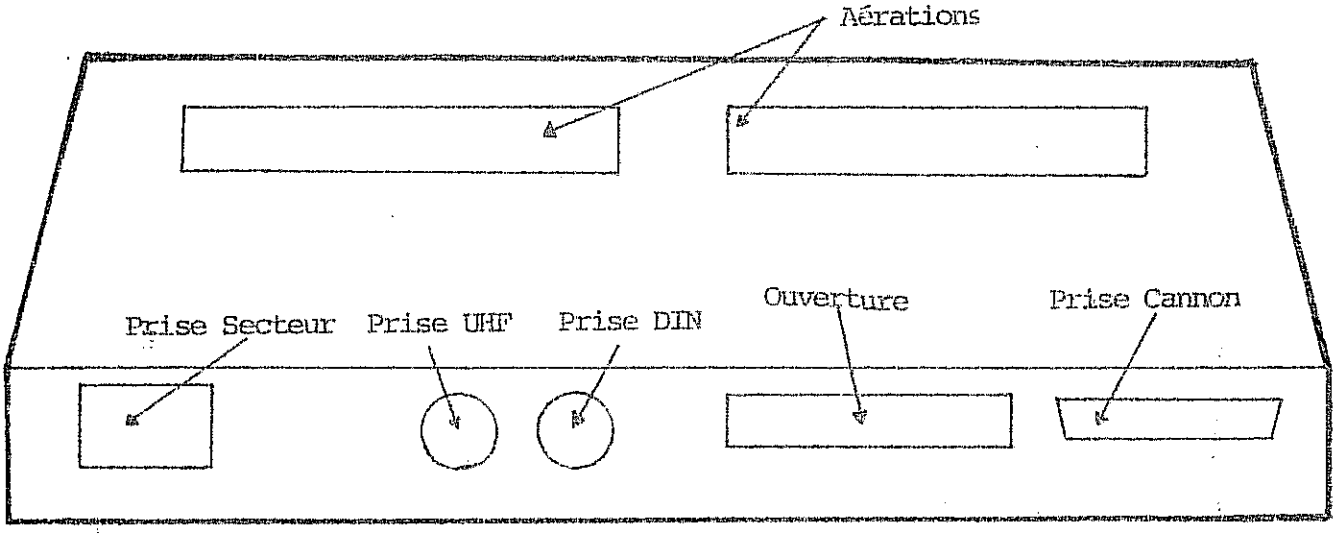
F15 = 921,6 KHZ
F16 = 1,843 MHZ

5°) Il ne vous reste plus maintenant qu'à refermer votre appareil en suivant les indications données précédemment.

BYE : Cet interrupteur n'est utilisé que dans le cas où l'on dispose de l'extension FLOPPY DISK.
Son but est de déclancher la sauvegarde automatique du programme utilisateur et ces données contenues en mémoire en vue d'une utilisation ultérieure.

PANIC: Cet interrupteur n'est utilisé que si l'utilisateur perd le contrôle de son programme et que la frappe de la touche BREAK reste sans effet.
Son but est de restituer le contrôle de l'appareil à l'utilisateur.

Sur la face arrière de votre PROTEUS III sont réunies toutes les prises et connecteurs permettant le raccordement à divers périphériques.

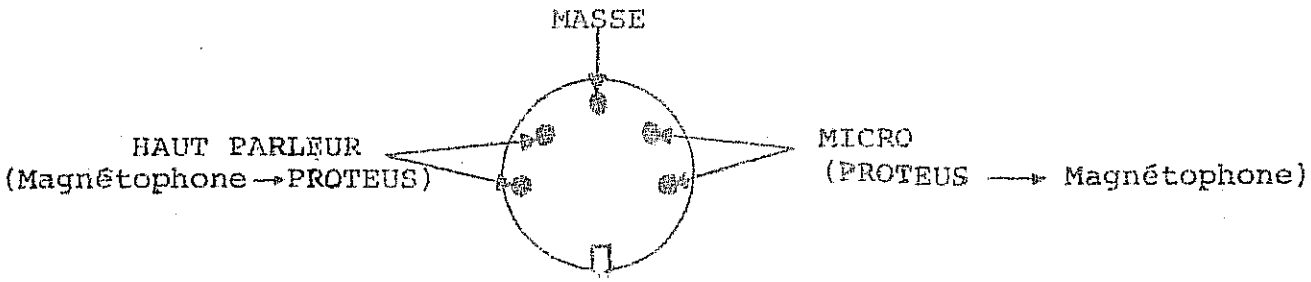


PRISE SECTEUR : Cette dernière sert à alimenter l'appareil.
La tension de service est de 220 V 50 HZ.
La consommation de l'appareil est de 60 W.

PRISE U.H.F. : Votre PROTEUS III est capable d'utiliser un téléviseur classique comme console de visualisation.
Il vous suffit pour cela de brancher dans cette prise, le câble d'antenne de votre téléviseur, et de régler ce dernier de manière à obtenir une réception parfaite du signal image.

PRISE DIN : Cette prise sert au raccordement d'un magnétophone classique, afin d'utiliser ce dernier comme mémoire de masse.
Dans ce cas, nous nous servirons de l'entrée Micro et de la sortie Haut parleur du magnétophone.

VUE ARRIERE de L'APPAREIL

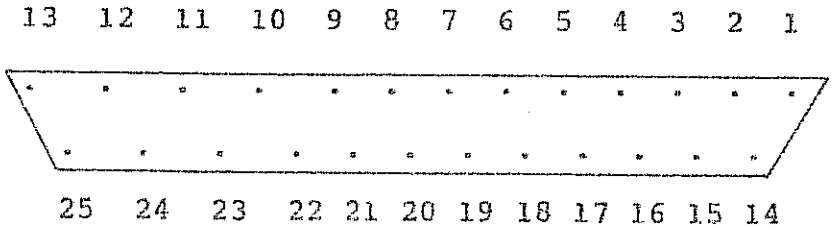


OUVERTURE : Cette dernière a été prévue à l'arrière de votre PROTEUS III de manière à faire passer le câble en nappe devant relier celui-ci au boitier contenant les FLOPPY DISKS.

PRISE CANNON : Sur cette prise, sont reliés les divers signaux de commande destinés à augmenter le choix des périphériques utilisables.

- Imprimante travaillant en RS 232C, boucle de courant, ou TTL
- Moniteur Vidéo
- Certains signaux de contrôle sont également regroupés sur cette prise afin de pouvoir brancher n'importe quel type de périphériques.

Le brochage de cette prise en vue arrière de l'appareil s'établit comme suit:

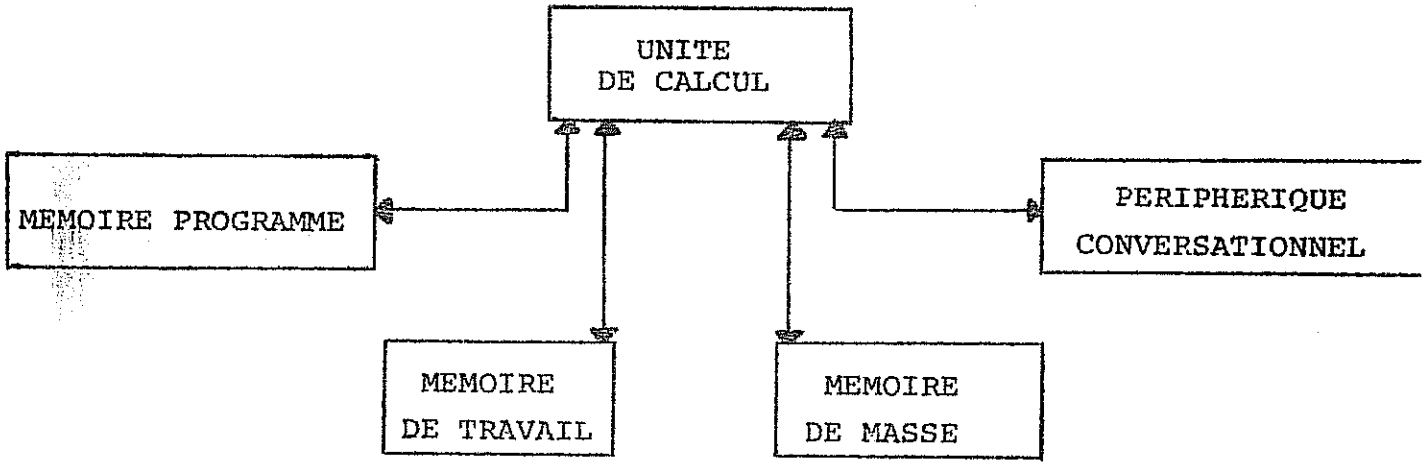


BROCHE	SIGNAL
1	+ 12 Volts
2	- 5 Volts
3	Masse
4	TX RS232 (PROTEUS ⟶ Périphérique)
5	RX RS232 (Périphérique ⟶ PROTEUS)
6	RX 20mA (Périphérique ⟶ PROTEUS)
7	RX TTL (Périphérique ⟶ PROTEUS)
8	TX 20mA (PROTEUS ⟶ Périphérique)
9	TX TTL (PROTEUS ⟶ Périphérique)
10	RDC RS232
11	Non connecté
12	Sortie Vidéo
13	Masse
14	+ 5 Volts
15	+ 5 Volts
16	Masse
17	Masse
18	- 12 Volts
19	Horloge de transmission
20	RTS
21	DCD
22	CTS
23	Masse
24	Non connecté
25	Non connecté

Une fois tous les raccordements nécessaires effectués, il ne vous reste plus qu'à profiter des avantages énormes que pourra vous procurer votre Système PROTEUS III.

I L'ORDINATEUR QU'EST-CE-QUE C'EST ?

I 1 C'est avant tout un outil capable de traiter des informations. Pour cela on lui donne une liste d'instructions appelée programme. On peut représenter un ordinateur de la manière suivante :



I 1a l'unité de calcul :

C'est la partie intelligente de l'ordinateur. Elle s'occupe de faire tous les calculs comme son nom l'indique.

I 1b la mémoire programme :

Une mémoire est une "Boîte Noire" capable de retenir des informations.

Elle contient un programme fixé une fois pour toutes, et écrit par le constructeur. Ce programme est appelé "Interpréteur BASIC". En effet, il interprète le programme de l'utilisateur afin de la rendre compréhensible pour l'unité centrale.

I 1c la mémoire de travail

Elle contient le programme de l'utilisateur, les données de celui-ci et des résultats intermédiaires utilisés par le BASIC.

Plus cette mémoire est importante et plus le programme pourra être puissant.

Sur PROTEUS III, cette mémoire comprend 16 000 cases.

Chacune de ces cartes peut contenir par exemple une lettre (A,B,C...), deux chiffres (12,47...) ou une valeur quelconque dans un langage qui lui est propre.

I 1d la mémoire de masse

C'est en fait une (ou plusieurs) boîte (s) située (s) à l'extérieur de l'ordinateur, raison pour laquelle on l'appelle périphérique.

Elle permet le stockage d'informations ou de programmes en grandes quantités.

La mémoire de masse comme la mémoire de programme est non volatile, contrairement à la mémoire de travail ; c'est à dire que son contenu ne s'efface pas lorsque l'on coupe le courant.

Sur PROTEUS III, ces mémoires sont principalement de deux types :

1° le mini-cassettes :
C'est un moyen simple et pratique pour stocker des informations.

- il a le principal avantage d'être économique (au moins dix fois moins cher qu'un FLOPPY),
- il a le principal défaut d'être relativement lent (30 cases mémoire seconde) et de monopoliser durant ce temps l'attention de l'ordinateur.

2° le FLOPPY disk :
Il a principalement l'avantage d'être plus rapide (250 000 cases mémoire seconde).

De plus, durant son utilisation, l'ordinateur peut faire autre chose.

Par ailleurs, le temps d'accès moyen, c'est à dire le temps pour retrouver une donnée quelconque sur le disque est d'environ 100 millisecondes, alors que sur un mini-cassettes, si la donnée se trouve à l'autre bout de la cassette, ben !...

I 1e LES PERIPHERIQUES CONVERSATIONNELS

Comme leurs noms l'indiquent, ils ne font pas partie intégrante avec l'ordinateur, mais ils sont indispensables.

En effet, ce sont eux qui permettent le dialogue entre l'homme et la machine :

- c'est le clavier qui permet de rentrer les données,

- c'est le téléviseur qui permet de sortir les données,

- c'est l'imprimante qui permet d'imprimer les tableaux, tracer les courbes, sortir les listes de stocks.

Tous ces périphériques obéissent à PROTEUS III au doigt et à l'oeuil, lui-même obéit à son propriétaire, son maître.

I 2e LE BASIC

Nous avons vu qu'un programme rangé quelque part dans l'ordinateur, le BASIC, se chargeait de rendre le programme de l'utilisateur compréhensible par l'unité de calcul (encore appelée Unité Centrale).

Le BASIC a été choisi par PROTEUS INTERNATIONAL pour différentes raisons :

- la première est sa célébrité. Le BASIC est en effet un langage informatique universel. Son nom vient d'une abréviation (Begginers All purpose Symbolic Instruction Code) qui grossièrement traduite signifie langage d'usage général pour

débutant ce qui amène à ...

- La deuxième raison : son emploi est d'une facilité

- La troisième raison enfin, qui n'est pas négligeab est que le BASIC à la capacité de traiter des mots quelconques (DUPONT, CHAPEAU) de manière très simple et donc de permettre un dialogue constant entre l'homme et la machine.

I 3 Conclusion de ce premier chapitre

L'ordinateur est donc une boîte ayant la possibilité d'exécuter une liste d'instruction qu'on lui aura donné (programme).

Et c'est en fait la seule chose qu'il soit capable d faire.

Il n'y a rien de plus bête qu'un ordinateur.

Son avantage est d'avoir de la mémoire , de ne jamais se tromper, et de travailler très rapidement.

Lorsqu'il travaille en BASIC, il est en outre facile de lui donner des instructions et de discuter avec lui.

Ce que nous avons appris dans ce chapître :

- Ordinateur
- Unité de Calcul
- Unité Centrale
- Périphériques
- Mémoire de programme
- Mémoire de masse
- Case mémoire
- Périphérique conversationnel
- Floppy
- BASIC

MISE EN SERVICE DU PROTEUS III

PROTEUS III est livré entièrement monté et testé. Il est alimenté en 220 volts et nécessite pour répondre d'être raccordé à un téléviseur, ou mieux, à un moniteur vidéo.

Pour le mettre sous tension, branchez la prise secteur fournie avec l'appareil dans la prise située à l'arrière du coffret.

Assurez-vous ensuite que l'interrupteur POWER est sur OFF

Raccordez alors la prise antenne 2ème chaîne du téléviseur sur la prise arrière du PROTEUS III.

Branchez ensuite la prise secteur des deux appareils au cas où cela ne serait pas déjà fait.

Dans le cas où vous disposeriez de périphériques ou/et de cartes extension, consultez le manuel correspondant pour l'installation de ceux-ci.

Mettre ensuite en marche le téléviseur et le PROTEUS III en mettant l'interrupteur en position ON.

Si vous ne possédez qu'un téléviseur, il faudra ensuite régler le canal d'antenne (sur 39) afin d'obtenir l'image.

Ce réglage n'existe pas sur le moniteur vidéo.

Vous verrez alors apparaître :

```
PROTEUS
       INTERNATIONAL

BASIC 8K
REV 4.7

READY
#
```

Raccordement d'un Magnétophone

Le magnétophone permet de conserver simplement les programmes et les données.

On l'utilise sur PROTEUS III en modeséquentiel ;il va lire les données depuis le début de la bande jusqu'à celles qui l'intéressent.

Le raccordement entre PROTEUS III et le magnétophone se fait de la même manière qu'avec un élément de chaîne HIFI. Il faut disposer d'un cordon de raccordement DIN 5broches. L'une des deux prises ira sur le PROTEUS et la deuxième sur le magnétophone. Dans le cas où le magnétophone ne posséderait pas les normes DIN. Il faudra utiliser des fiches de raccordement adaptées.

Vous pouvez maintenant introduire la cassette de présentation dans le magnétophone et taper sur le clavier LOAD # 4 return;

Mettez ensuite le magnétophone en marche. PROTEUS va lire sous vos yeux, après un certain délai, le programme de présentation.

Lorsque cette opération s'achève le BASIC répond

READY

#

A ce moment tapez RUN return et le programme s'exécute.

Il ne vous rest donc plus maintenant qu'à lire le rest du manuel afin de pouvoir écrire vos propres programmes.

IV PRISE EN MAIN DE PROTEUS III

Après avoir mis l'interrupteur sur la position ON, vous devez voir apparaître sur votre écran de visualisation :

```
PROTEUS
       INTERNATIONAL

BASIC 8K
REV 4.7

READY
#
```

Appuyez alors sur la touche RETURN plusieurs fois de suite.

Remarquez qu'à chaque fois, un = a été imprimé sur la Console de Visualisation. L'ordinateur essaie de vous dire su'il est prêt à prendre en compte les commandes que vous taperez.

Nous allons maintenant essayer de lui donner un ordre:
Après l'apparition du = , tapez

```
PRINT "JE SUIS PROTEUS III"
```

Appuyez maintenant sur la touche RETURN

L'ordinateur nous répondra immédiatement en imprimant sur la console :

```
JE SUIS PROTEUS III
READY
#
```

Il n'a donc fait qu'exécuter ce que nous lui avons demandé. Essayez à nouveau :
Celà marche à tous les coups, n'est-ce-pas ?...

Vous venez d'avoir le premier dialogue avec votre ordinateur et ce dernier n'a fait qu'exécuter ce que vous lui avez ordonné.

Analysons la ligne que vous venez d'écrire:

- La traduction française du terme PRINT est : IMPRIMER

- Le texte que nous avons écrit entre guillemets correspond à ce qu'il fallait imprimer.

- Le fait d'appuyer sur la touche RETURN a eu pour effet

de faire savoir au PROTEUS III, que nous avons terminé de lui donner des ordres et qu'il lui faut maintenant les exécuter.
Essayons à nouveau:

Ecrivons :

PRINT "JE SUIS A VOS ORDRES

N'oubliez pas d'appuyer sur la touche [RETURN]

Que s'est-il passé? Celà n'a pas marché?

L'ordinateur a répondu de la manière suivante :

Error 19 in Line ØØØØ.

Il a voulu par là nous dire que nous avons commis une erreur et qu'il ne lui a pas été possible d'exécuter ce que nous lui avons demandé.
En effet il est capable de détecter les erreurs commises par l'utilisateur et d'en informer aussitôt ce dernier. Les erreurs sont numérotées; et il faut se reporter au tableau des codes erreurs afin d'en connaître la nature. C'est-ce que nous allons faire

Erreur 19 = Pas de " de fermeture dans l'impression d'une chaîne de caractères.

Examinons la ligne que nous venons d'écrire; en effet, nous avons oublié de fermer les guillemets.
Il est donc important de noter que l'ordinateur n'effectue pas les ordres que l'utilisateur lui donne s'il rencontre une erreur de ponctuation ou de grammaire.
Le fait d'écrire PRIMT au lieu de PRINT nous aurait renvoyé à l'erreur Ø2 qui se traduit de la manière suivante :

Le terme utilisé ne fait pas partie du langage BASIC.

Ecriture d'un programme simple

Nous avons vu dans le chapitre précédent, que l'ordinateur était capable d'exécuter les ordres immédiatement.
Pour la résolution de problèmes plus complexes, il est alors nécessaire d'écrire ce que l'on appelle un Programme. Ce dernier est formé de lignes, écrites en BASIC, qui ont pour but de guider l'ordinateur au fur et à mesure de la résolution du problème. Chaque ligne est numérotée à l'aide d'un nombre qui peut varier entre ØØØ1 et 9999.

Ecrivons par exemple :

```
ØØ1Ø  PRINT "JE SUIS VOTRE PROTEUS III"  [RETURN]
ØØ2Ø  PRINT "JE SUIS A VOS ORDRES"  [RETURN]
```

Il faut maintenant spécifier à PROTEUS III que notre pro-

gramme est terminé; à cet effet nous écrivons :

ØØ3Ø END. [RETURN]

Nous allons maintenant demander l'exécution immédiate du programme que nous avons écrit en mémoire :

RUN [RETURN]

Ah !ça marche.

Sur notre écran de visualisation , l'ordinateur vous a répondu :

JE SUIS VOTRE PROTEUS III

JE SUIS A VOS ORDRES

Nous allons maintenant modifier le programme et insérer une nouvelle ligne. Il suffit pour celà de donner, à la ligne que nous allons écrire, un numéro intermédiaire.

Ecrivons:

ØØ15 PRINT "VOUS M'AVEZ APPELE ?" [RETURN]

Exécutons maintenant le programme:

RUN [RETURN]

Nous voyons apparaître immédiatement :

JE SUIS VOTRE PROTEUS III

VOUS M'AVEZ APPELE ?

JE SUIS A VOS ORDRES

Il est donc facile de modifier un programme écrit en BASIC, il suffit pour cela de donner des numéros de lignes intermédiaires.

Relisons notre programme

Il vous suffit pour cela de taper :

LIST [RETURN]

Vous verrez alors apparaître :

```
ØØ1Ø PRINT "JE SUIS VOTRE PROTEUS III"
ØØ15 PRINT "VOUS M'AVEZ APPELE ?"
ØØ2Ø PRINT "JE SUIS A VOS ORDRES"
ØØ3Ø  END

READY
#
```

Nous avons donc maintenant sous les yeux le listing du programme que nous avons écrit.

Afin de rendre la lecture de ce dernier plus agréable, il est intéressant de pouvoir y glisser des remarques ou des explications.

Ecrivons :

```
ØØØ5 REM  MON PREMIER PROGRAMME      RETURN
```

Demandons maintenant l'exécution de notre programme :

```
RUN   RETURN
```

Il s'avère que notre programme n'a pas été modifié car l'ordinateur nous a répondu exactement la même chose que la dernière fois.

"Ah ! je savais bien que l'on ne pouvait pas compter sur ces machines."

Essaypns de voir ce qu'il s'est passé et demandons l'impression du listing du programme.

```
LIST   RETURN

ØØØ5    REM  MON PREMIER PROGRAMME
ØØ1Ø    PRINT "JE SUIS VOTRE PROTEUS III"
ØØ15    PRINT "VOUS M'AVEZ APPELE"
ØØ2Ø    PRINT "JE SUIS A VOS ORDRES"
ØØ3Ø    END
```

Cela n'a pas marché et pourtant la ligne ØØØ5 a bien été prise en compte, puisqu'elle est imprimée lors d'un LIST.

En fait ce qu'i s'est passé est tout à fait normal, votre PROTEUS III a parfaitement marché.

Il apparaît donc que l'ordre REM n'est pas pris en compte au cours du programme et qu'il n'apparaît que lorsque l'on demande l'impression d'un listing.

Cette commande est très utile car elle va nous permettre de glisser des remarques ou des explications à n'importe quel endroit d'un listing.

De même qu'il nous est possible d'écrire un programme nous pouvons également l'effacer :

Il suffit pour cela de taper :

```
NEW   RETURN
```

Que voyons nous apparaître sur notre terminal ?

```
READY
=
```

Cela veut donc dire que notre commande a été prise en compte.

et que, PROTEUS III attend qu'on lui donne des ordres.

Vérifions que notre programme a bien été effacé :

LIST [RETURN]

PROTEUS III nous répond alors :

READY
#

Il ne reste donc aucune trace du programme précédent en mémoire.

Afin de rendre plus agréable le travail de l'utilisateur, PROTEUS III est capable de lui-même de numéroter les lignes d'un programme.
Le constructeur a prévu que les lignes seraient numérotées de 1Ø en 1Ø; il est néanmoins possible de demander une numérotation plus "serrée" (Ex de 5 en 5) ou plus "large" (Ex de 2Ø en 2Ø).

Voyons celà :

Après l'apparition du # indiquant que l'on est prêt à prendre en compte les commandes de l'utilisateur, appuyer sur la touche [RUBOUT]
Vous devez alors voir apparaître sur votre terminal :

ØØ1Ø

Il suffit d'entrer alors la ligne de programme correspondante suivie bien sur d'un [RETURN]

Par Exemple

REM MON DEUXIEME PROGRAMME [RETURN]
[RUBOUT] PRINT "J'APPRENDS A UTILISER PROTEUS" [RETURN]
[RUBOUT] END [RETURN]

Demandons maintenant l'exécution de notre programme :

RUN [RETURN]

Celà marche n'est-ce pas!

Effacez maintenant toute trace du programme contenu en mémoire.

C'est fait ?

J'espère que vous n'avez pas oublié l'utilité de la commande NEW.

Nous allons maintenant réécrire le même programme, mais en numérotant les lignes de 5 en 5, au lieu de 1Ø en 1Ø

Ecrivons :

INCR = 5 [RETURN]

Le BASIC numérotera alors les lignes de 5 en 5 comme vous le lui avez ordonné.

Appuyez maintenant sur la touche [RUBOUT]

Le premier numéro de ligne doit alors apparaître :

ØØØ5

Je vous laisse le soin de réécrire le programme précédent et de vérifier que ce dernier fonctionne parfaitement.

Nous allons maintenant arriver à une des notions les plus importantes en langage BASIC .

Il est en effet possible d'effectuer des branchements inconditionnels grâce à la commande GOTO.

Croyant à la vertu de l'exemple, nous allons immédiatement vous en donner un .

Essayez d'entrer en mémoire le programme suivant en utilisant la numérotation automatique des lignes. N'oubliez pas d'effectuer la commande INCR = 1Ø

```
ØØ1Ø  PRINT  "BONJOUR"
ØØ2Ø  PRINT  "JE SUIS VOTRE PROTEUS III"
ØØ3Ø  GOTO 5Ø
ØØ4Ø  PRINT  "JE NE SUIS PAS A VOS ORDRES"
ØØ5Ø  PRINT  "J'ATTENDS VOS ORDRES"
ØØ6Ø  END
```

Si maintenant nous donnons l'ordre au BASIC d'effectuer la commande RUN, nous verrons sur notre terminal :

```
RUN

BONJOUR
JE SUIS VOTRE PROTEUS III
J'ATTENDS VOS ORDRES


READY
```

Analysons notre programme afin de voir ce qu'il s'est passé

La ligne n° 1Ø demande au BASIC d'imprimer BONJOUR

La ligne n° 2Ø demande au BASIC d'imprimer JE SUIS VOTRE PROTEUS

La ligne n° 3Ø demande au BASIC de sauter à la ligne 5Ø

La ligne n° 4Ø demande au BASIC d'imprimer

JE NE SUIS PAS A VOS ORDRES

La ligne n° 5Ø demande au BASIC d'imprimer J'ATTENDS VOS ORDRES

La ligne n° 6Ø spécifie que c'est la fin de notre programme...

Que s'est-il passé ?

Pourquoi la ligne 4Ø n'a-t-elle pas été exécutée ?

A la ligne n° 3Ø nous avons demandé au BASIC d'effectuer un branchement inconditionnel au numéro de ligne 5Ø et c'est ce qu'il a fait. Il n'a donc pas tenu compte de la ligne 4Ø.

Afin de voir si vous avez bien compris cette instruction fondamentale, nous allons vous faire faire un petit exercice.

Il s'agit de demander à votre PROTEUS III d'imprimer :

BONJOUR, JE SUIS VOTRE PROTEUS III

plusieurs fois de suite, sans s'arrêter ; de plus il faudra numéroter les lignes de programme de 1Ø en 1Ø.
Avant toute chose, n'oubliez pas d'effacer le programme précédemment contenu en mémoire à l'aide de la commande NEW.

Alors, vous avez terminé ?

Voici la solution :

NEW RETURN

RUBOUT PRINT "BONJOUR, JE SUIS VOTRE PROTEUS III" RETURN

RUBOUT GOTO 1Ø RETURN

Demandez maintenant l'exécution de votre programme :

RUN RETURN

Dès la frappe de la touche RETURN , l'exécution du programme commence.

Sur notre terminal, nous voyons apparaître :

BONJOUR, JE SUIS VOTRE PROTEUS III

BONJOUR, JE SUIS VOTRE PROTEUS III

BONJOUR, JE SUIS VOTRE PROTEUS III

-

-

Que se passe-t-il ? ne va-t-il jamais s'arrêter ?

Revoyons ce que nous avons écrit :

```
ØØ1Ø  PRINT "BONJOUR JE SUIS VOTRE PROTEUS III"
ØØ2Ø  GOTO 1Ø
```

Chaque fois que la ligne n° 1Ø est rencontrée, l'impression commence.
A la ligne n° 2Ø nous demandons au BASIC de retourner à la ligne n° 1Ø.

Il est donc normal que l'impression de la phase demandée ne s'arrête jamais.

Dans ce cas, il existe un moyen bien simple de rendre la main à l'utilisateur :
Il suffit d'appuyer simultanément sur les touches CTRL et C
Le BASIC rend alors immédiatement la main à l'utilisateur et affiche le mot READY suivi d'un = .

Nous allons modifier le programme précédent et lui damander d'afficher la phrase "BONJOUR MONSIEUR"

Il suffit de modifier la ligne n° 1Ø.

Nous écrivons donc :

```
1Ø PRINT "BONJOUR MONSIEUR" [RETURN]

RUN   [RETURN]
```

Votre programme fonctionne-t-il ?

La phrase s'imprime-t-elle bien plusieurs fois de suite sur votre terminal ?

N'oubliez pas que la frappe d'un CTRL C aura pour effet de rendre la main à l'utilisateur.

Nous avons donc vu que pour corriger une ligne dans un programme, il suffit de réécrire cette dernière en lui donnant le même numéro.
De même pour annuler une ligne, il suffit d'entrer son numéro suivi d'un [RETURN] .
De plus, il peut être intéressant de pouvoir modifier une ligne d'écriture.

EXEMPLE :

Reprenons l'exemple précédent :

Si en ligne ØØ2Ø nous avions écrit GETE au lieu de GOTO, il aurait suffit d'appuyer simultanément sur les touches CTRL et O trois fois de suite, et d'entrer les caractères valides soit OTO.
La frappe d'un CTRL O nous fait revenir en arrière d'un

caractère.

De la même manière, si l'on veut annuler la ligne en cours d'écriture, il suffit d'appuyer simultanément sur les touches CTRL et X

Exemple

Soit à annuler en cours d'écriture la ligne suivante :

ØØ2Ø GOTO CTRL X

Le BASIC répondra alors "DELETED" afin d'informer l'utilisateur de l'annulation de la ligne.

Remarque :
La commande CTRL X annule la ligne entrée et non la ligne en mémoire :

Exemple :

```
ØØIØ PRINT "BONJOUR"
ØØ2Ø END
```

et que l'on entre

ØØIØ PRINT "BONSOIR" CTRL X

Le programme en mémoire est inchangé.

Nous vous conseillons de bien assimiler ce chapître avant de passer aux suivants, car les termes que nous avons introduits seront utilisés souvent.

QU'AVONS-NOUS APPRIS DANS CE CHAPITRE

- = signifie que PROTEUS III est prêt à prendre en compte les commandes de l'utilisateur.
- PRINT est une commande permettant l'impression d'une chaîne de caractères contenue entre guillemets.
- Lorsque l'on appuie sur la touche RETURN, celâ a pour effet de valider ce que l'utilisateur vient d'écrire.
- Le BASIC est capable de détecter les erreurs et d'en informer l'utilisateur sous forme d'un code.
- Il est important de bien respecter l'orthographe des commandes aussi que la ponctuation.
- END indique au BASIC qu'il est arrivé à la fin du programme utilisateur.
- RUN a pour effet de faire exécuter immédiatement le programme contenu en mémoire.
- L'ordre dans lequel nous écrivons les lignes n'a pas d'importance, ces dernières sont exécutées par ordre croissant de numéros.
- La commande LIST demande au BASIC d'afficher les lignes du programme utilisateur sur le terminal.
- La commande REM permet de glisser des remarques ou des explications dans un listing de programme.

- La commande NEW permet d'effacer toutes traces du programme utilisateur contenu en mémoire.
- La numérotation automatique de ligne d'obtient en appuyant sur la touche RUBOUT.
- Il est possible de changer la valeur de l'incrément en utilisant la commande INCR.
- La commande GOTO provoque un branchement inconditionnel en numéro de ligne spécifié.
- La frappe d'un CTRL C rend la main à l'utilisateur (READY =)
- Pour revenir d'un caractère en arrière, taper CTRL O .
- Pour annuler une ligne en cours d'écriture, taper CTRL X.
- Pour réécrire une ligne, il suffit d'entrer le même numéro de ligne suivi de la série de caractères valides.
- Pour annuler une ligne, il faut entrer son numéro suivi d'un RETURN.

V LES VARIABLES NUMERIQUES

Nous avons vu dans le chapitre précédent que PROTEUS III était capable d'imprimer du texte sur le terminal de l'utilisateur.

Il est capable aussi de traiter des valeurs numériques associées à des variables.

Une variable est une grandeur dont la valeur est susceptible d'être modifiée à tout instant.

L'instruction LET permet d'assigner des nombres à des variables.

EXEMPLE : ØØ1Ø LET A=342

Dans la suite du programme, lorsque l'on fera référence à la variable A, elle vaudra 342.
Il faut remarquer que dans ce cas le signe = n'a pas la même signification qu'en mathématique.
En effet, à tout instant du programme, nous avons le droit d'écrire A=A+1
Dans ce cas, la variable A prendra la valeur précédente plus 1

Le signe = utilisé en BASIC signifie donc :

"prend la valeur de"

Une variable est définie par un caractère alphabétique ou un caractère alphabétique suivi d'un chiffre compris entre Ø et 9

EXEMPLE :

C est une variable légale
Z est une variable légale
A3 est une variable légale
X9 est une variable légale
AB est une variable illégale

A12 est une variable illégale

Il est très important de se rappeler qu'une même lettre ne peut désigner qu'une variable et une seule.

En effet si nous écrivons :

```
ØØ1Ø LET A=9
ØØ2Ø LET A=3
```

Dans la suite du programme, la variable A prendra la valeur 3.

Nous avons vu précédemment qu'il était possible d'imprimer du texte, il en est de même pour les caleurs assimilées à des variables.

Exemple :

```
ØØ1Ø  LET A=342
ØØ2Ø  PRINT A
ØØ3Ø  END
```

Dans l'exécution de ce petit programme, nous verrons apparaître sur le terminal de l'utilisateur :

```
342

READY
#
```

A titre d'exercice, nous allons maintenant vous demander d'écrire un petit programme.

Soit à afficher une suite de nombres croissants en partant de 1, sur le terminal.

Vous avez terminé ?

Demandez alors l'exécution de votre programme.

Fonctionne-t-il bien ?

Voilà comment il faut procéder :

```
ØØ1Ø  LET A=1
ØØ2Ø  PRINT A
ØØ3Ø  A=A+1
ØØ4Ø GOTO 2Ø
```

Seule la frappe d'un CTRL C permet alors d'arrêter l'exécution du programme.
En effet :

Chaque fois que la ligne 3Ø est rencontrée, la valeur assignée à la variable A s'incrémente de 1.

Chaque fois que la ligne 4Ø est rencontrée, le BASIC retourne à la ligne 2Ø.

Chauqe fois que la ligne 2Ø est rencontrée, la valeur associée à la variable A est affichée sur le terminal.

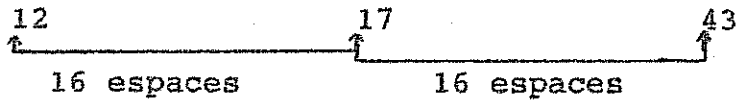
Dans l'expression A=A+1, le signe égal signifie "prend la valeur de" ; ce signe n'a donc pas la même signification qu'en mathématiques.

Il est possible sur une même ligne de programme, d'écrire plusieurs instructions, à conditions toutefois que ces dernières soient séparées par :

Exemple :

```
ØØ1Ø LET A=12: LET B=17: LET C=43
ØØ2Ø PRINT A,B,C
ØØ3Ø END
```

Dès que l'on demande l'exécution du programme le terminal nous affiche alors :



Nous remarquons immédiatement que les 3 valeurs ne sont pas imprimées de faço contigue, mais que 16 caractères séparent chaque premier chiffre de chaque valeur.

Essayons à nouveau:

Cette fois, au lieu de séparer les variables par des virgules nous allons les séparer par des point-virgules.

Le programme s'écrira donc comme suit :

```
ØØ1Ø LET A=12: LET B=17: LET C=43
ØØ2Ø PRINT A;B;C
ØØ3Ø END
```

Nous obtiendrons alors l'impression suivante :

```
12 17 43
```

Nous remarquons immédiatement, que les valeurs des variables A,B,C ont été imprimées les unes à la suite des autre, séparées par 1 blanc.

L'utilisation du terme LET étant facultative, nous aurions pu écrire :

```
ØØIØ A=12:B=17:C=43
```

"Alors pourquoi nous avez vous introduit le terme LET" me direz-vous.

Tout simplement parce qu'en cours de lecture de programme écrit sur une autre machine, vous auriez pu rencontrer ce terme. Il est donc nécessaire de connaître son utilité.

Quel type de notation utilise le BASIC ?

La version PROTEUS III du BASIC travaille en mode virgule flottante sur neuf décimales.
Il est donc possible d'exprimer des nombres sous la forme suivante :

A =32
F = 43.768

Attention : Ce n'est pas une erreur d'impression, nous avons bien écrit 43.768 et non 43;768

Le fait de mettre une virgule au lieu d'un point aurait pour effet de vous renvoyer aux codes erreur.

" La capacité maximale est donc de 999999999 me direz-vous"

Ce n'est pas vrai car le BASIC change alors de manière d'exprimer les nombres.

Exemple :
Soit à traiter le nombre suivant :
1Ø9 ØØØØØØØØØØ

Votre PROTEUS III exprimera alors ce nombre sous la forme exponentielle :

en effet :
IØ9ØØØØØØØØØØ = 1Ø9 E 10

La gamme des nombres pouvant être traités va donc de

1.Ø E^{-99} jusqu'à 999999999 E^{99}

E^{99} représente alors $1Ø^{99}$
E^{-99} représente alors $1Ø^{-99}$

PROTEUS III est donc capable de traiter un large éventail de nombres; ces derniers pouvant s'exprimer sous trois formes :

Notation entière : 1Ø24
Notation avec virgule : 314.768
Notation exponentielle : 618 E^{14}

Nous vous avons précédemment dit que le BASIC travaillait en virgule flottante sur neuf décimales. Il est possible néanmoins de choisir le nombre de décimales voulues après la virgule (qui en fait se trouve être un point)

En effet le prix d'un article peut difficilement s'exprimer sous la forme suivante :

256.67142 Francs .

Il existe donc une commande permettant de choisir le nombre de décimales voulues, elle se nomme DIGITS.

A titre d'exemple, nous allons sortir une liste d'articles avec les prix associés à ces derniers.

Nous écrivons donc :

```
DIGITS=2
ØØ1Ø  T=62O.5O: C:143.2O: L=248.IO
ØØ2Ø  PRINT "LE PRIX DE LA TABLE EST DE";T;"FRANCS"
ØØ3Ø  PRINT "LE PRIX DE LA CHAISE EST DE";C;"FRANCS"
ØØ4Ø  PRINT "LE PRIX DE LA LAMPE EST DE";L;"FRANCS"
```

Demandons maintenant l'exécution de notre programme.

```
LE PRIX DE LA TABLE EST DE 62O.5O FRANCS

LE PRIX DE LA CHAISE EST DE 143.2O FRANCS

LE PRIX DE LA LAMPE EST DE 248.IO FRANCS
```

Nous voyons immédiatement apparaître l'utilité de cette commande.

Les nombres concernés sont tronqués automatiquement pour se conformer à ce degré de précision ; cela veut dire que les derniers chiffres sont abandonnés.
De même, dans le cas contraire, des Ø peuvent être ajoutés à un nombre afin d'arriver à la décimalisation voulue.

Exemples:

Si l'on demande 2 chiffres après le point :

```
3.07248          3.Ø7
3                3.ØØ
```

```
DIGITS =1 demande une décimale après le point
DIGITS =2 demande deux décimales parès le point
DIGITS =3 demande trois décimales après le point
.
.
.
ETC ....
```

Le fait de demander DIGITS=Ø remet le BASIC en mode virgule flottante.

Nous allons maintenant vous initier aux calculs en BASIC.

Il est bien sur possible d'effectuer les opérations arithmétiq
élémentaires.

Ces dernières sont appellées opérateurs mathématiques et
notés de la manière suivante :

- Addition +
- Soustraction -
- Multiplication *
- Division /

Comme dans les calculs courants, il est nécessaire d'util
ser des parenthèses si on veut imbriquer plusieurs opérateurs
mathématiques à la fois.

Exemple:

Soit à effectuer l'opération mathématique suivante

$Y = AX^2 + BX + C$

Traduite en BASIC, cette opération s'écrira :

```
Y = (A*X*X ) +(B*X)+C
```

Il est important de ne pas oublier de mettre les parenth
car leur omission pourrait conduire à des résultats erronés.

Ecrivons le programme correspondant à la résolution de l
quation précédente en prenant les valeurs suivantes :

A=3 , B=7 , C=43 , X=12

```
ØØ1Ø  A=3: B=7: C=43: X=12
ØØ2Ø  Y= (A * X * X) +( B*X) +C
ØØ3Ø  PRINT "LE RESULTAT DEMANDE EST ";Y
ØØ4Ø  END
```

Auusitôt après la validation de la commande RUN, nous v
rons apparaître :

```
LE RESULTAT DEMANDE EST 559
```

Afin de vous familiariser avec les opérateurs mathémati
nous allons vous proposer un petit exercice :

Sachant que le diamètre d'un cercle est de 2,43 mètres, calculer le tiers de sa circonférence :

Nous vous rappelons que la formule permettant de calculer la circonférence d'un cercle est :

C = 2xπ x R

π prenant la valeur 3, 14159

R désignant le rayon du cercle considéré

Avez- vous terminé d'écrire le programme ?

J'espère que vous n'avez pas oublié d'utiliser les parenthèses.

Voici la solution :

```
ØØ1Ø  P= 3.14159 : D=2.43
ØØ2Ø  Z= (2*(D/2)*P) /3
ØØ3Ø  PRINT "LE TIERS DE LA CIRCONFERENCE VAUT";Z;"METRES"
ØØ4Ø  END
```

De même que nous avons vu les opérateus mathématiques, nous pouvons définir les opérateurs relationnels.
Ces derniers servant à effectuer des comparaisons:

< Signifie que le terme exprimé à gauche du signe est plus petit que celui de droite

> Signifie que le terme exprimé à gauche du signe est plus grand que celui de droite

<= Signifie que le terme exprimé à gauche du signe est plus petit ou égal à celui de droite

>= Signifie que le terme exprimé à gauche du signe est plus grand ou égal à celui de droite

<> Signifie que les deux termes considérés sont différents

= Signifie que les deux termes considérés sont égaux

Ces opérateurs servent conjointement à l'instruction IF THEN

Cette instruction permet d'exécuter l'expression située après le terme THEN uniquement si une condition est remplie.
En d'autres termes, il est possible d'effectuer des comparaisons et éventuellement de prendre des décisions.

Exemples :

Soit à calculer le carré des dix premiers nombres en partant de 1
Le programme s'écrirait alors comme suit :

```
ØØ1Ø  A=1
ØØ2Ø  B=A*A
ØØ3Ø  PRINT "LE CARRE DE";A;"EST";B
ØØ4Ø  IF A=1O THEN END
ØØ5Ø  A=A+1
ØØ6Ø  GOTO 2Ø
```

Analysons ce que nous venons d'écrire :

- La ligne n° 1Ø initialise la variable A à la valeur 1
- La ligne n° 2Ø calcule le carré de la valeur assignée de A et ce dernier à la valeur B
- La ligne n° 3Ø permet à l'utilisateur de visualiser le résultat sur son terminal.
- La ligne n° 4Ø compare la valeur de A à 1Ø . Si la relation est vraie (c'est à dire A=1Ø) alors la commande END sera prise en compte.

Si la relation est fausse (c'est à dire si A est différent de 1Ø) le BASIC passe immédiatement à la ligne de programme suivante.

- La ligne n° 5Ø a pour résultat d'incrémenter la valeur de la variable A
- La ligne n° 6Ø nous renvoie à la ligne 2Ø afin de calculer le carré de la nouvelle valeur de A.

Toutes les instructions faisant partie du langage BASIC sont exécutables après l'expression THEN

Nous avons en particulier le droit d'écrire la phrase suivante :

```
IF  A=12  THEN  GOTO 6Ø
```

Nous voyons alors immédiatement apparaître la possibilité d'effectuer des branchements conditionnels.

Dans ce cas, l'écriture de l'expression GOTO est facultative.

En effet, cette ligne pourrait s'écrire de la manière suivante :

```
IF  A=12 THEN 6Ø
```

Nous allons maintenant vous demander de réécrire le petit programme précédent en utilisant la notion de branchements conditionnels.

Avez-vous terminé ? Votre programme fonctionne-t-il correctement ?

Si ce n'était pas le cas, recommencez jusqu'à l'obtention du résultat désiré.

Afin que vous puissiez vérifier votre programme, nous vous en donnons le listing.

```
ØØ1Ø  A=1
ØØ2Ø  B=A * A
ØØ3Ø  PRINT "LE CARRE DE";A;"EST";B
ØØ4Ø  A = A+1
ØØ5Ø  IF A < 11  THEN 2Ø
ØØ6Ø  END
```

Remarque

Dans la ligne n° 5Ø , nous comparons la variable A à la valeur 11 car cette dernière est incrémentée à la ligne précéde,te.

Il est également possible d'effectuer des comparaisons multiples sur une même ligne.

En effet nous avons le droit d'écrire

```
IF A=12  IF B=43  THEN GOTO 6Ø
```

Dans ce cas, on ne va en 6Ø que si on a à la fois A=12 et B=43

Grâce à certaine fonctions mathématiques classiques, nous allons pouvoir étendre notre puissance de clacul BASIC.

En effet, PROTEUS III est capable d'effectuer des fonctions mathématiques complexes dont nous vous donnons ci-après la liste.

- LOG :Cette fonction permet de calculer le logarithme naturel d'un nombre positif quelconque.
  S'il était négatif, vous serez renvoyé à l'erreur n°21 qui se traduit de la manière suivante :
  "on a cherché à calculer le LOG d'un nombre négatif"

Exemple:

```
ØØ1Ø LET A=LOG(42)
```

- DLOG : Ce calcul diffère du précédent du fait que le logarithme calculé est du type décimal.
  Tout comme dans le cas précédent, le fait de calculer le logarithme décimal d'un nombre négatif aura pour effet de vous renvoyer à l'erreur n° 21

Exemple:

```
ØØ1Ø  LET A=DLOG(47)
```

- EXP : Cette fonction permet de calculer l'exponentielle d'un nombre.
  e représentant ici la base des logarithmes naturels.

Exemple:

```
ØØ1Ø LET  C=EXP(27)  (alors C=e^27)
```

- ↑ : Il est possible grâce à cette fonction d'élever des nombres à une puissance.
  Il est à noter que cette fonction se sert du logarithme

et de l'exponentielle.
Du fait du nombre de calculs à effectuer, il est donc plus rapide d'écrire A+A ..... pour les puissances faibles.

Exemple

ØØIØ LET A=B ↑ 17

Remarque :
La fonction A n'est définie que si B est positif.
SI B est négatif, on a une erreur n° 21
Par contre A peut être totalement quelconque.

- SQR: De même qu'il est possible d'élever un nombre à une puissance, votre PROTEUS III est capable de calculer la racine carrée, d'un nombre quelconque (à condition bien sûr que ce dernier reste dans les limites des nombres pouvant être traités)

Exemple:
ØØ1Ø LET C=SQR (13412)

Votre PROTEUS III est même capable d'effectuer des calculs trigonométriques?

- SIN : Cette fonction est utilisée afin de calculer un sinus. La valeur de l'angle doit obligatoirement être exprimée en radians.

Exemple:
ØØIØ LET A=SIN(41)

-COS: A l'aide de cette fonction, il est également possible de calculer un cosinus.
De même que dans le cas précédent, la valeur de l'angle est exprimé en radians.

Exemple:
ØØIØ LET A=COS(12)

- TAN : Cette fonction s'utilise de la même manière que SIN et COS et permet de calculer la tangente.

Exemple:
ØØIØ LET T=TAN(X)
N'oubliez pas d'exprimer X en radians.

- ATAN: Cette fonction permet de calculer la réciproque de TAN. Cette dernière définit donc l'Arc Tangente . La valeur calculée à l'aide de cette fonction sera exprimée en radians.

Exemple:
ØØIØ LET A=ATAN(X)
X s'exprime donc en radians.

"Puisque les données doivent être exprimées en radians pour

les calculs trigonométriques, il faut donc que je m'achète en plus une machine à calculer, car mes données à moi sont en degrés me direz-vous"

NON !!! Votre serviteur personnel et dévoué sait également faire ce genre de conversions grâce aux deux fonctions suivantes.

- RAD: Cette fonction permet de convertir un angle exprimé en degrés, en radians.

Exemple:
ØØIØ PRINT RAD (9Ø)

- DEG: Cette fonction se trouve être l'inverse de la précédente. Elle convertit un angle exprimé en radians, en degrés.

Exemple:
ØØIØ PRINT DEG (3.14159)

Afin de pouvoir faire du traitement de nombres, votre PROTEUS III a appris à exécuter les fonctions suivantes.

- ABS: Il est aisé grâce à cette fonction de connaître la valeur absolue d'une expression quelconque.

Exemple:
ABS(-46.372) = 46.372
ABS(37.42) = 37.42

- INT: Cette fonction désigne le nombre entier immédiatement inférieur de l'expression considérée.
Les nombres ne sont pas arrondis mais tronqués (derniers chiffres abandonnés)

Exemple:
INT (14.412) = 14
INT (-15.2) = -16

Remarques:
Il est également possible d'arrondir un nombre, il suffit de lui ajouter 0.5 et de le ramener à l'entier immédiatement inférieur.
La ligne de programme s'écrira alors de la manière suivante

- ØØIØ LET A=INT(14.412 + 0.5)
le nombre obtenu (14) correspond bien à la valeur arrondie de 14.412
- ØØIØ LET A=INT(14.6I7 + 0.5)
le nombre obtenu (I5) correspond bien à la valeur arrondie de 14.6I7

-SGN: Cette fonction s'utilise dans le cas où l'on a besoin de connaître le signe d'une valeur numérique quelconque

Exemples:
SGN (7.43) =1

```
SGN (-14.3) = -1
SGN (Ø)     = Ø
SGN (-Ø)    = Ø
```

- RND/ Il est intéressant dans certains cas de pouvoir générer des nombres aléatoires.
Cette fonction produit un éventail uniformément distribué.

de nombres aléatoires.
Elle s'écrit sous la forme suivante:

```
ØØ1Ø  LET A= RND(X)
```

Dans le cas ou X=Ø , un nombre compris entre Ø et 1, différent à chaque fois , sera généré.
Dans le cas ou X ≠ Ø , le nombre généré (toujours compris entre Ø et 1) sera toujours le même pour une même valeur deX

Exemples:
Ecrivons le petit programme suivant, et demandons son exécution plusieurs fois de suite.

```
ØØ1Ø LET A= RND(Ø)
ØØ2Ø PRINT A
ØØ3Ø END
```

Nous pouvons immédiatement remarquer le nombre inscrit sur le terminal est différent à chaque fois.

Modifions la ligne n°1Ø

```
ØØ1Ø LET A= RND(3)
```

Chaque fois que l'on demande l'exécution du programme le même nombre est affiché sur le terminal.
Si l'on remplace le chiffre 3 par une valeur quelconque (différent de Ø) le nombre retourné sera différent du précédent.

Afin d'agrementer le travail de l'utilisateur , PROTEUS III possède en mémoire des variables numériques prédéfinies (constant
Celles ci peuvent être appellées à tout moment et utilisées en cours de programme. Pour y faire appel, il faut placer un @ devant son nom

Il y a cinq constantes en mémoires

- PI : appelle la valeur numérique de (3.14159265)
- E : appelle la valeur numérique de e qui est la base des logarithmes naturels (2.7I8281838)
- GALI : appelle la valeur numérique correspondant au nombre de litres par Gallon (3.785412)
- INCH : appelle la valeur numérique correspondant au nombre de centimètres par INCH (2.54)
- LK : appelle la valeur numérique correspondant au nombre de kilogramme par livre (O.453592)

La manière de faire appel à une constante est la suivante:

Exemple:

```
PRINT @PI
```

Aussitôt que l'on appuie sur la touche RETURN, la valeur de est envoyée sur le terminal.
A titre s'exercice nous vous conseillons d'appeler toutes lesconstantes les unes après les autres.

Nous allons maintenant vous donner de plus amples renseignement sur l'utilisation des parenthèses dans les calculs.

Lorsque le BASIC arrive à une ligne qu'il doit exécuter, il commence par l'examiner dans son intégralité de gauche à droite Lorsqu'il trouve des calculs, il les effectue de la même manièr que vous.

- Il exécute d'abord les fonctions à une seule variable.

Exemple:
SIN (X)

- Ensuite l'élévation à la puissance
- Puis multiplication et division
- Et pour finir les additions et soustractions

Il nous est donc possible de définir un ordre de priorités

1°) les fonctions

2°) ↑

3°) * ou /

4°) + ou -

Remarque:
Les opérateurs de même priorité sont traités de gauche à droite.

Il est donc possible d'omettre les parenthèses dans certai cas :

Exemples:

Soit à effectuer l'opération suivante

$Y = AX^2 + BX + C$

Nous aurons en BASIC, le droit d'écrire la ligne de programme de la manière suivante :

LET Y=A*X↑2+B*X+C

Si l'opération à effectuer est du type

Y=B(X+3)

Il est alors impératif de mettre des parenthèses dans la ligne de programme, elle s'écrira donc comme suit.

LET Y=B*(X+3)

De même manière dans l'exemple suivant

X^2 → X↑2 / Y↑½

Si l'on remplace X↑2 pour X*X on doit mettre des parenthèses :

X*X/(Y*Y) sinon celà donnerait

X*X/Y*Y

$X^2/Y*Y$ soit $\frac{X^2}{Y} * Y = X$

Remarques

L'ordre d'exécution des expressions a été établi de façon à calculer n'importe quelle expression mathématique exprimée suivant les règles habituelles.

QU'AVONS-NOUS APPRIS DANS CE CHAPITRE

- Une variable est une grandeur susceptible d'être modifiée.

- Les variables numériques se définissent par un caractère alphabétique suivi ou non d'un chiffre compris entre Ø et 9

- LET est une instruction permettant d'assigner une valeur à une variable, son utilisation est facultative.

Il est possible d'imprimer la valeur assignée à une variable à l'aide de l'instruction PRINT.

- Il est possible d'écrire plusieurs instructions sur une même ligne à condition que ces dernières soient séparées par :

- Lors de l'instruction PRINT, le fait de mettre des virgules sépare les différents impressions de I6 Caractères, le fait de mettre des point-virgules donnera une impression continue.

- Les nombres peuvent s'exprimer sous trois formes:
  - -La notation entière
  - -La notation avec virgule
  - -La notation exponentielle

- L'instruction DIGITS permet de choisir le nombre de décimales voulues après la virgule.

- Les opérateurs mathématiques sont exprimés à l'aide des signes suivants : +,-, *,/

- Les opérateurs relationnels s'expriment de la manière suivante:
  =, <>, <, >, <=, >=

- L'instruction IF...THEN permet d'effectuer des comparaisons et de prendre des décisions.

- Les fonctions permettent d'effectuer des calculs complexes.

- Les constantes sont des variables prédéfinies.

- Les ordres de priorité dans les calculs.

VI Les VARIABLES STRING - Le TRAITEMENT des Chaînes Caractères

Nous appellerons STRING une chaîne de caractères alphanumériques.
Le blanc est un caractère alphanumérique à part entière.
Il nous est possible d'assimiler des chaînes de caractères à des variables afin de pouvoir éventuellement les traiter.
Ces dernières s'expriment sous la même forme que les variables numériques sauf qu'elles sont suivies d'un $, et qu'elles ne sont formées que d'un caractère alphabétique.

Exemple:

```
A$= VARIABLE STRING
A9$ =VARIABLE STRING ILLEGALE
Z1$ =VARIABLE STRING ILLEGALE
Z = VARIABLE NUMERIQUE
AB  =VARIABLE ILLEGALE
```

Pour assigner une chaîne de caractères à une variable STRING, on procède de la même manière que pour les variables numériques.

Exemples:

```
ØØIØ  LET A$="BONJOUR"
ØØ2Ø  PRINT A$
ØØ3Ø  END
```

Demandons l'exécution de notre programme:
Nous voyons immédiatement apparaître

```
     BONJOUR
READY
#
```

Essayons à nouveau :

```
ØØIØ  LET A$="JE SUIS VOTRE PROTEUS III"
ØØ2Ø  PRINT A$
ØØ3Ø  END
```

Votre PROTEUS III a dû vous répondre :

```
JE SUIS VOTRE PROT
```

Que s'est-il passé ?

Tout simplement, nous avons omis de vous dire, qu'il faut prévenir le BASIC, si l'on désire traiter des chaînes de caractères supérieures à I8, afin que ce dernier réserve de l'espace mémoire à cet effet

C'est l'utilité de l'instruction STRING.
Une variable STRING peut contenir 255 caractères au maximun (à condition toutefois que l'on n'oublie pas de le spécifier)

Essayons donc de faire fonctionner le programme précédent :

```
STRING=25 RETURN

RUN
```

Ah! ça marche!!!

Il nous est possible de faire aussi ce que l'on appelle du traitement de chaîne de caractère.
Il nous est en particulier possible de les concatener (mettre bout à bout)

Exemple:

```
ØØ1Ø  LET A$="BONJOUR"
ØØ2Ø  LET B$="COMMENT ALLEZ VOUS"

ØØ3Ø  LET C$=A$+ B$
ØØ4Ø  PRINT C $
ØØ5Ø  END
```

Demandons maintenant l'exécution de notre programme:

RUN RETURN

Nous voyons immédiatement apparaître sur notre terminal

```
BONJOUR COMMENT ALLEZ VOUS

READY
```

Nous pouvons concatener autant de variables STRING les unes sur les autres à condition toutefois de ne pas dépasser la capacité maximale de 255 Caractères.
Il nous est possible également grâce à l'instruction LEN de connaître le nombre de caractères alphanumériques contenus dans une variable STRING.

Exemple:

```
ØØ1Ø  LET A$="ORDINATEUR"
ØØ2Ø  LET A$= LEN(A$)
ØØ3Ø  PRINT A$
ØØ4Ø  END
```

Lorsque l'on a demandé l'exécution de notre programme, nous avons vu apparaître sur notre terminal:

```
 10
READY
#
```

Essayons à nouveau,
Pour celà, nous allons modifier uniquement la ligne n°1Ø

```
ØØ1Ø LET A$="VARIABLE STRING"
```

Cette fois, PROTEUS III nous a répondu 15.

Nous savons qu'à chaque caractère alphanumérique est assimilé un nombre, et que ce dernier est représentatif du caractère considéré.

Il nous est possible, grâce à l'instruction ASC de connaître ce nombre par un caractère déterminé.

Exemple:

```
ØØIØ  LET A$="B"
ØØ2Ø  LET C = ASC(A$)
ØØ3Ø  PRINT C
ØØ4Ø  END
```

Dès que l'on demande l'exécution immédiate du programme utilisateur contenu en mémoire, nous voyons apparaître sur notre terminal.

```
66

READY
#
```

66, représente ici la valeur décimale assimilée au caractère alphanumérique de B.

"Que se passe-t-il lorsqu'au lieu de considérer un caractère, on considère une chaîne de caractères alphanumériques? me direz-vous.

Essayons, et écrivons:

```
ØØIØ  LET A$="BONJOUR MONSIEUR"

ØØ2Ø  LET B = ASC (A$)
ØØ3Ø  PRINT B
ØØ4Ø  END
```

Qu'avons- nous vu apparaître sur notre terminal ?.

```
66

READY
#
```

De ce qui précède, il résulte donc si l'on considère une chaîne de caractères au lieu d'un caractère seul, c'est le premier caractère alphanumérique de la chaîne qui sera considéré.

Il est intéressant de pouvoir assimiler des chiffres contenu dans une chaîne de caractères à une variable numérique afin de pouvoir effectuer des calculs.

A cet effet, on peut se servir de l'instruction VAL.

Exemple:

```
ØØIØ  LET A$ ="1"
ØØ2Ø  LET A  = VAL (A$)
```

```
ØØ3Ø  PRINT A
ØØ4Ø  END
```

Lorsque l'on a demandé l'exécution du programme, la valeur 1 a été affichée sur le terminal.

Cela nous montre que la valeur 1 a bien été assimilée à la variable A.

Afin de bien se rendre compte de l'importance de cette instruction, nous allons vous donner un autre exemple plus pratique.

Exemple:

Soit à calculer le prix de dix articles sachant que le prix unitaire de l'article est contenu dans une chaîne de caractères.

```
ØØIØ  LET A$="1968"
ØØ2Ø  LET A = VAL (A$)
ØØ3Ø  LET B = A*1Ø
ØØ4Ø  PRINT "LE PRIX DE 10 ARTICLES EST";B;"FRANCS"

ØØ5Ø  END
```

Votre PROTEUS III a-t-il bien affiché

LE PRIX DE 1Ø ARTICLES EST I968Ø FRANCS

sur votre terminal ?

Si ce n'était pas le cas, revoyez l'écriture de votre programme.

Remarque:

Il est possible de sortir d'une chaîne de caractères, dont les premiers sont des caractères numériques la valeur correspondante.

Exemple: VAL (10 FRANCS) vaut 10
mais VAL (FRANCS IO) donne une erreur.

Il existe une autre instruction du langage BASIC qui permet d'effectuer exactement l'inverse de ce que nous venons d'exécuter. C'est l'instruction STR$.

Elle permet d'introduire une variable numérique dans une chaîne de caractères.

Exemple:

```
ØØIØ  LET A=32
ØØ2Ø  LET A$=STR$(A)
ØØ3Ø  LET B$="FRANCS"
ØØ4Ø  LET C$= A$+ B$
ØØ5Ø  PRINT C$
ØØ6Ø  END
```

Demandez l'exécution de votre programme.

Votre terminal a-t-il bien affiché 32 FRANCS ?

Cette instruction est très importante car elle permet la transmission de résultats numériques dans une chaîne de caractères.

Nous avons vu que l'on pouvait assimiler à une variable numérique le code correspondant à n'importe quel caractère alphanumérique.

Chaque caractère a un numéro de code compris entre ØØØ et 255 (voir à la fin du manuel la table des numéros de code).

Il est possible, grâce à ce code, de retrouver le caractère correspondant grâce à l'instruction CHR$.

Exemple:

```
ØØIØ  LET A=72
ØØ2Ø  LET A$ = CHR$(A)
ØØ3Ø  PRINT A$
ØØ4Ø  END
```

Le terminal, lors de l'exécution du programme, nous affichera la lettre H.

Les codes décimaux correspondant à chacun des caractères alphanumériques est représenté à la fin du manuel.

Grâce aux instructions que nous avons examinées, nous savons effectuer plusieurs opérations sur les variables STRING.

Mais nous ne savons pas encore sélectionner un mot ou un caractère spécifique dans une chaîne.

Grâce aux trois instructions qui suivent, il nous sera possible d'effectuer de telles opérations.

Il est aisé de considérer un nombre de caractères contenus dans une chaîne en partant de la gauche grâce à l'instruction LEFT$.

Exemple:

```
ØØIØ  LET A$="BONJOUR MONSIEUR"
ØØ2Ø  LET B$= LEFT (A$,7)
ØØ3Ø  PRINT B$
ØØ4Ø  END
```

Analysons ce que nous venons d'écrire ; nous nous limiterons volontairement à la ligne n°2Ø, les autres vous étant maintenant familières.

Nous avons demandé à PROTEUS III de considérer 7 caractères de la chaîne A$ en partant de la gauche, et d'assimiler cette nouvelle chaîne de caractères à la variable STRING B$.

La variable B$ doit donc contenir les caractères suivants : "BONJOUR"

Nous allons pouvoir le vérifier immédiatement en demandant l'exécution de notre programme.

Nous voyons apparaître sur notre terminal.

```
BONJOUR

READY
#
```

Afin de bien comprendre le fonctionnement de cette instruction, nous vous conseillons de modifier le ligne ØØ2Ø du programme précédent et de considérer un nombre différent de caractères.

L'instruction RIGHT$ fonctionne de la même manière que la précédente, sauf qu'elle nous permet de considérer un nombre de caractères en partant de la droite.

Modifions la ligne ØØ2Ø du programme précédent et écrivons:

```
ØØ2Ø  LET B$ = RIGHT (A$,8)
```

Demandons maintenant l'exécution de notre programme.

Nous devrions voir apparaître sur notre terminal:

```
MONSIEUR

READY

#
```

Grâce à cette instruction, nous avons donc la possibilité de considérer un nombre défini de caractères dans une variable STRING en partant de la droite.

"Comment faire si l'on veut considérer un nombre spécifique de caractères si ces derniers sont situés au milieu de la chaîne" me direz-vous.

Dans ce cas, on se sert d'une autre instruction qui s'appele MID$, en effet cette dernière permet de considérer un nombre de caractères quelconque en partant de n'importe quel endroit de la chaîne.

Exemple:

Nous allons considérer que la chaîne de caractères "COMMENT VAS TU ?" est assignée à la variable STRING A$ et que nous voulons assigner à la variable STRING B$ le mot VAS :

Nous écrivons alors le programme suivant :

```
ØØIØ  LET A$="COMMENT VAS TU ?"
ØØ2Ø  LET B$= MID (A$, 9,3)
ØØ3Ø  PRINT B$
ØØ4Ø  END
```

Demandons maintenant l'exécution de notre programme:
Nous voyons alors apparaître sur notre terminal.

```
VAS

READY

#
```

De ce qui précède, nous pouvons déduire que :

- 9 représente le numéro d'ordre du premier caractère à prendre en compte en partant de la gauche.
- 3 représente le nombre de caractères à prendre en compte

Cette instruction s'exprime donc sous la forme générale suivante:

Y$ = MID$ (X$,I,J)

Si le paramètre J n'est pas spécifié, la chaîne de caractères considérée (Y$),commencera au I ième caractère jusqu'à la fin du STRING.

Vérifions cela et modifions la ligne ØØ2Ø du programme précédent:

```
ØØ2Ø  LET B$=MID$(A$,9)
```

La chaîne de caractères B$, devra alors contenir:"VAS TU?"

Vérifions immédiatement en demandant l'exécution de notre programme.

Cela marche n'est-ce pas ?

Afin de s'assurer que vous avez bien compris nous allons nous amuser à effectuer un petit exercice:

Soit à afficher sur le terminal de l'utilisateur des chaînes de caractères dont la longueur augmentera à chaque impression jusqu'à l'obtention de la phrase suivante:

"COMMENT VAS TU?"

Le programme s'écrira alors comme suit :

```
ØØIØ  LET A$="COMMENT VAS TU?"

ØØ2Ø  LET A =1
ØØ3Ø  LET B$=LEFT$ (A$,A)
ØØ4Ø  PRINT B$
ØØ5Ø  LET A = A +1
ØØ6Ø  IF  A =16 THEN END
ØØ7Ø  GOTO  3Ø
```

Analysons ce que nous venons d'écrire:

- La ligne ØØIØ nous permet de définir la chaîne de caractères que nous allons traiter, nous l'avons appellée A .
- La ligne ØØ2Ø sert à initialiser la variable A
- La ligne ØØ3Ø définit la nouvelle variable STRING que nous allons considérer. Le nombre de caractères pris en compte à partir de la gauche sera la valeur assignée à la variable numérique A.
- La ligne ØØ4Ø demande l'impression de la nouvelle variable STRING que nous venons de créer.
- La ligne ØØ5Ø demande l'incrémentation de la variable numérique A.

- Dans la ligne ØØ6Ø nous comparons la valeur de A à la valeur maximun que cette dernière doit prendre. L'instruction END qui définit la fin de notre programme ne sera prise en compte que si cette valeur maximun est atteinte.
- La ligne ØØ7Ø qui n'est rencontrée que si A n'a pas atteint sa valeur maximun demande à votre PROTEUS III de retourner à la ligne ØØ3Ø

Demandons l'exécutions de ce programme, nous verrons alors apparaître:

```
C
CO
COM
COMM
COMME
COMMEN
COMMENT
COMMENT
COMMENT V
COMMENT VA
COMMENT VAS
COMMENT VAS
COMMENT VAS T
COMMENT VAS TU
COMMENT VAS TU ?

READY

=
```

Remarque:
Les caractères blancs ou espaces sont considérés comme des caractères à part entière.

A titre d'exercice, nous vous conseillons de refaire la même chose que précédemment en partant de la droite au lieu de la gauche.

Vous avez terminé ? votre programme fonctionne-t-il ?

Si, ce n'était pas le cas, nous allons vous aider:

- Seule la ligne ØØ3Ø est à modifier et il faut utiliser l'instruction RIGHT au lieu de LEFT .

Nous savons maintenant traiter des chaînes de caractères grâce aux instructions que nous avons vu dans ce chapître.

Nous allons maintenant vous apprendre à organiser l'impression de vos résultats sur votre terminal.

Essayons d'afficher la phrase suivante:

```
ØØIØ  PRINT "NOUS ALLONS VOUS APPRENDRE A FAIRE DE LA MISE EN PAGE"

ØØ2Ø  END
```

Demandons maintenant l'exécution du petit programme que nous venons d'écrire.

Que s'est-il passé ?

Pourquoi l'impression a-t-elle été affichée sur deux lignes ?

Simplement parce que votre PROTEUS III sait faire automatiquement de la Mise en Page.
En effet, dans le but de ne pas couper accidentellement un caractère, il effectue automatiquement un retour chariot dès qu'il se trouve dans le dernier quart de la ligne et qu'il rencontre un espace.
D'origine, il est prévu pour travailler sur des lignes longues de 64 caractères.
Certaines imprimantes disposent de lignes plus longues, il est possible de régler cet automatisme ; ceci grâce à la commande LINE.

Si l'on veut inhiber ce phénomène, on aura soin de demander un LINE supérieur d'un quart au nombre de caractères effectivement désirés.
Dans le cas où l'on travaille à l'aide de la console de visualisation, et que l'on veut inhiber ce mécanisme, nous demanderons:

```
LINE =8Ø
```

En effet 64 + (1/4)64 = 8Ø

Nous vous conseillons, à titre d'exercice, d'effectuer la commande LINE =8Ø et de demander l'exécution du programme précédent.

Il nous est possible, lors de l'exécution de l'instruction PRINT, de faire de la mise en page automatique grâce à la ponctuation
(, ;)

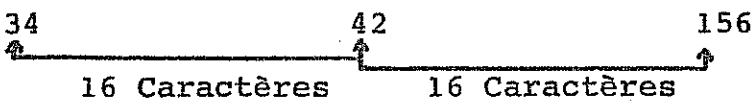
Voyons vela et écrivons le programme suivant :

```
ØØIØ  LET A=34: LET B=42: LET C=156
ØØ2Ø  PRINT A,B,C
ØØ3Ø  END
```

Après l'exécution de ce petit programme, que remarquons-nous ?

L'espace séparant chaque premier caractère imprimé correspond à 16 caractères blancs.

En effet, nous avons obtenu l'imprssion suivante :



Il nous est également possible d'avoir une impression continue, il suffit pour cela de mettre des points virgules au lieu des virgules.
Reprenons l'exemple précédent et modifions la ligne ØØ2Ø

```
ØØ2Ø  PRINT A;B;C
```

Lorsqu'on demande l'exécution du programme, nous remarquons qu'un espace sépare les valeurs imprimées, ce qui n'est pas le cas avec des chaînes de caractères.

Exemple:

```
PRINT "BONJOUR";"MONSIEUR"

Donne :  BONJOURMONSIEUR
```

Donc quand on sépare des impressions par ; on effectue les impressions:

1°) immédiatement après une variable STRING

2°) après insertion d'un blanc après une variable numérique.

Si l'on désire, en cours d'impression, sauter une ligne, il suffit d'écrire l'instruction PRINT.

Exemple:

```
ØØIØ LET A=174: LET B=12: LET C=42: LET D=36
ØØ2Ø PRINT A,B
ØØ3Ø PRINT
ØØ4Ø PRINT C,D
ØØ5Ø END

Dans ce cas, nous obtiendrons l'impression suivante :

174                  12

  16 Caractères          1 ligne d'espacement
42                   36
```

Lors de la mise en page, il faut prendre garde à la commande LINE.
En effet, si un espace est rencontré dans le dernier quart de la ligne, un retour chariot automatique sera effectué si l'on n'a pas pris soin d'inhiber ce mécanisme.

Il est également possible de faire de la mise en page manuelle, c'est la raison d'être de l'instruction TAB.

A titre d'exemple, écrivons:

```
ØØIØ PRINT TAB (I8); "BONJOUR"
ØØ2Ø END
```

Nous obtiendrons alors l'impression suivante :

```
I8 CARACTERES     BONJOUR
```

La première position d'impression correspond à TAB (1).

Le fait de demander un TAB (Ø) vous renverra à l'erreur n° 15

Cette instruction s'exprime donc sous la forme PRINT TAB (X) X peut être soit un nombre, soit une variable numérique.

Il est donc possible d'obtenir des espacements variables, et en particuliers de dessiner la courbe représentative d'une fonction mathématique quelconque.

Exemple:

Soit à dessiner la courbe représentative de la fonction $Y = X^2/4$ Pour des valeurs de X variants entre 1 et 16

Nous écrivons donc:

```
ØØIØ  LET X=1
ØØ2Ø  LET Y=X*X/4
ØØ3Ø  PRINT TAB (Y);"*"
ØØ4Ø  IF X=16 THEN END
ØØ5Ø  X =X+1
ØØ6Ø  GOTO 2Ø
```

N'oubliez pas, avant de demander l'exécution de votre programme, d'inhiber la mise en page automatique.
Vous aurez donc soin de spécifier LINE=8Ø avant de demander l'exécution du programme.

Ce dernier fonctionne-t-il correctement ?

"Le nombre d'espaces à imprimer devrait être un nombre entier, que se passe-t-il lorsque ce dernier ne l'est pas" me direz-vous?

Ce dernier est tout simplement tronqué (derniers chiffres abandonnés) avant d'être pris en compte par l'instruction TAB.

Si l'on désire plus de précision dans les tracés de courbes demandés, nous aurons soin d'ajouter 0.5 à la variable prise en compte pour effectuer l'instruction TAB.
Cela aura pour effet d'obtenir un nombre arrondi à l'entier le plus proche au lieu d'obtenir un chiffre tronqué.

Il est également possible de connaître, à tout instant, la position de la tête d'impression ou du curseur dans le cas d'une console de visualisation. C'est l'utilité de l'instruction POS.

Ecrivons un petit programme.

```
ØØIØ  A=1
ØØ2Ø  PRINT TAB (A);"*"; IF POS=12 THEN END
ØØ3Ø  A=A+1
ØØ4Ø  GOTO 2Ø
ØØ25  PRINT
```

Analysons ce que nous venons d'écrire:

- Dans la ligne n° ØØIØ, la variable A est initialisée.

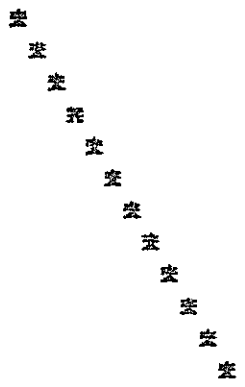
- dans la ligne ØØ2Ø, l'impression de l' * se fait après l'espacement de A caractères blancs.

Lorsque la tête d'impression se trouvera à la douzième position, l'instruction END sera prise en compte et votre PROTEUS III vous rendra la main (READY *)

- dans la ligne n° ØØ3Ø, la variable A est incrémentée.

- dans la ligne n° ØØ4Ø, on demande au PROTEUS III de retourner à la ligne n° 2Ø.

Lors de l'exécution du programme, nous obtenons l'impression suivante :



READY

*

Vous êtes maintenant capable de faire de la mise en page avec votre PROTEUS III, ce qui est très important car cela vous permet d'organiser l'impression de vos résultats comme vous le désirez.

QU'AVONS NOUS APPRIS DANS CE CHAPITRE

- Nous appelons STRING une chaîne de Caractères alphanumériques.
- Les variables STRING se différentient des variables numériques à l'aide d'un $.
- Si le nombre de caractères contenus dans une variable STRING est supérieure à 18, il faut utiliser l'instruction STRING.
- Il est possible d'additionner deux variables STRING afin d'un créer une troisième.
- Une variable STRING peut contenir au maximun 255 Caractères.
- L'instruction LEN permet de connaître le nombre de caractères contenus dans une variable STRING.
- Les caractères blancs sont considérés comme des caractères normaux.
- L'instruction ASC permet de connaître le code décimal correspondant à un caractère alphanumérique.
- L'instruction VAL permet de transmettre un nombre d'une variable STRING dans une variable numérique.
- L'instruction STR$ permet la transmission de résultats numériques dans une chaîne de caractères.
- L'instruction CHR$ permet d'obtenir un caractère alphanumérique à partir de son code décimal.
- L'instruction LEFT$ permet de considérer un nombre spécifique de caractères en partant de la gauche.
- L'instruction RIGHT$ permet de considérer un nombre spécifique de caractères en partant de la droite.
- L'instruction MID$ permet de considérer un nombre spécifique de caractères situés au milieu d'une variable STRING.
- L'instruction LINE permet de changer à volonté le format d'impression.
- Les virgules et point virgules sont utilisés pour faire de la mise en page automatique.
- Le fait d'écrire PRINT tout seul à pour effet de sauter une ligne lors de l'impression.
- L'instruction TAB permet d'effectuer de la mise en page manuelle.
- Il est possible à tout moment de connaitre la position de la tête d'impression grâce à l'instruction POS.

VII LES VARIABLES INDICEES

Dans certains cas, nous pouvons être amenés à traiter des grandeurs de même nature.

Nous utiliserons à ce moment là ce que l'on nomme des Variables Indicées ; ces dernières s'expriment sous la forme A(x).

X peut prendre toutes les valeurs inférieures ou égales à 255.

Exemple:

```
A(12)   = Variable numérique indicée
Z9(43)  = Variable numérique indicée
C4(643)= Variable illégale
```

Il est également possible de donner un indice à des variables STRING :

```
B$(3)   = Variable STRING indicée
A7$(254)= Variable STRING indicée
Z$ (256)= Variable illégale
```

Pour examiner tout un groupe de variable de même nature, il suffit de faire varier l'indice.

Prenons un exemple concret:

Considérons des objets quelconques dont le prix varie en fonction de la taille.

Nous allons classer les tailles de la manière suivante:

```
5 . IØ . I5
```

Assignons ces tailles à une variable indicée:

Nous dirons donc:

```
A(1) = 5
A(2) = 1Ø
A(3) = 15
```

Considérons maintenant les différents prix et assignons les à une autre variable indicée:

```
B(1) = 1ØØ
B(2) = 15Ø
B(3) = 2ØØ
```

Nous remarquons immédiatement que pour un indice quelconque, la taille et le prix correspondent et qu'en faisant varier l'indice entre 1 et 3 nous examinons toutes les tailles ainsi que les prix qui s'y rapportent.

Pour afficher toutes les tailles et tous le sprix, il suffit donc d'écrire:

```
ØØIØ LET A(1)=5: LET A(2)=1Ø: LET A(3)=15
ØØ2Ø LET B(1)=1ØØ: LET B(2)=15Ø: LET B(3)=2ØØ
ØØ3Ø X=1
ØØ4Ø PRINT"POUR LA TAILLE ";A(X);"LE PRIX EST DE ";B(X);"F"

ØØ5Ø IF X=3 THEN END
ØØ6Ø X=X+1
ØØ7Ø GOTO 4Ø
```

Demandons maintenant l'exécution de notre programme, nous obtenons:

```
POUR LA TAILLE 5 LE PRIX EST DE 1ØØ F

POUR LA TAILLE 1Ø LE PRIX EST DE 15Ø F

POUR LA TAILLE 15 LE PRIX EST DE 2ØØ F


READY
#
```

Votre PROTEUS III réserve automatiquement de la place pour stocker les variables indicées, et il est possible de stocker IØ données par variables.

Si l'on veut pouvoir en stocker plus, il faut demander à ce que l'on réserve de l'espace mémoire à cet effet.

C'est l'utilité du STATEMENT DIM.

Exemple:

Soit à réserver de l'espace mémoire pour pouvoir stocker 243 données assimilées à une variable indicée.

Nous appellerons arbitrairement cette variable D4, il faudra alors dire : DIM D4(243)

La place mémoire nécessaire au stockage sera alors réservée.

Il est également possible de donner un double indiçage à des variables; cela à pour principal avantage de permettre à l'utilisateur de classer ses données ou ses résultats sous forme de tableaux.

Les variables à double indiçage s'exprimeront donc sous la forme D(X,Y).

Les termes X et Y peuvent varier entre 1 et 255.

Exemple:

Soit à calculer le résultat d'une équation en faisant varier deux de ses paramètres.

Il faudra en plus stocker les résultats en mémoire sous forme de tableau.

Nous écrivons alors:

```
ØØIØ  LET C=1
ØØ2Ø  LET D=1
ØØ3Ø  Y(C,D)=C*C*C=C+2*D
ØØ4Ø  PRINT Y(C,D),"";
ØØ5Ø  IF D=3 THEN GOTO 7Ø
```

```
ØØ6Ø  LET D=D+1:GOTO 3Ø
ØØ7Ø  PRINT
ØØ6Ø  IF C=3 THEN END
ØØ9Ø  LET C=C+1: GOTO 2Ø
```

Nous obtiendrons alors l'impression suivante:

3	5	7
18	2Ø	22
83	85	87

Nous voyons immédiatement apparaître les avantages relatifs à cette notation ; en effet il est aisé d'aller chercher un résultat quelconque dans le tableau afin de le traiter.

Il suffit pour celà de donner les numéros de ligne et de colonne

Par Exemple:

A (3,2) = 85

Comme dans le cas des variables à simple indiçage, il faut réserver de l'espace mémoire afin de stocker des données ou des résultats.

Reprenons le cas précédent, si l'on avait voulu faire varier le paramètre C entre 1 et 194 et le paramètre D entre 1 et 18, il aurait fallu dire en début de programme : DIM Y (194,18)

Votre PROTEUS III réservant automatiquement de l'espace mémoire pour le stockage de ces variables, si les indices restent compris entre 1 et 1Ø , il n'est pas nécessaire d'utiliser l'instruction DIM.

QU'AVONS NOUS APPRIS DANS CE CHAPITRE:

-Nous utilisons les variables indicées lorsque nous avons des grandeurs de même nature a manipuler

-PROTEUS III réserve automatiquement de la mémoire pour le stockage de donnees dont les indices varient entre 1 et 1Ø (ou entre 1,1 et 1Ø,1Ø dans le cas du double indicage).Si l'on veut pouvoir en stocker plus,il faut se servir de l'instruction DIM.

-La valeur maximale de l'indice est de 255 pour le simple indicage,et de 255x255 pour le double indicage.

VIII ENTREE DES DONNEES

Nous avons vu, dans les chapîtres précédents qu'il était possible de traiter des variables numériques et alphanumériques.
Jusqu'à présent, nous avons entré les données lors de l'écriture du programme.
Ce chapître a pour but de vous apprendre à rentrer vos données de manière plus rationnelle.
A cet effet, nous allons utiliser l'instruction INPUT .

Exemple:

```
ØØ1Ø  INPUT A
ØØ2Ø  LET B=SQR (A)
ØØ3Ø  PRINT B
ØØ4Ø  END
```

Demandons maintenant l'exécution du programme que nous venons d'écrire:
Nous voyons immédiatement apparaître un ?

L'impression de ce caractère est destinée à informer l'utilisateur que PROTEUS III attend qu'on lui rentre des données.
A correspondant à une variable numérique, nous allons lui donner une série de chiffres.

Exemple:

```
16    RETURN
```

La frappe du RETURN a pour but de spécifier que l'on a terminer de rentrer les données.
Le résultat de l'opération (SQR(16)) est alors immédiatement apparu sur notre Terminal.

Essayons de nouveau, pour celà, réclamez votre programme.

Dès l'apparition du ? tapez un caractère non numérique suivi d'un RETURN.

Votre PROTEUS III vous répond immédiatement RE ENTER ; c'est à dire qu'il a détecté que les caractères frappés ne pouvaient être assimilés à une variable numérique.
Il suffit alors de rentrer la série de caractères valides (Chiffres) afin d'obtenir un déroulement correct du programme.

Il est bien sur , également possible de rentrer une chaîne de caractères alphanumériques et de l'assimiler à une variable STRING.

Il suffira alors d'écrire INPUT A$
Le nombre maximun de caractères que l'on peut rentrer lors d'un INPUT est de 8I.
Si, cette capacité est dépassée, vous serez renvoyé à l'erreur numéro Ø9.

Comment doit-on procéder si l'on désire qu'une question soit imprimée afin que l'utilisateur puisse y répondre.

Reprenons l'exemple précédent et modifions la ligne N° 1Ø

```
ØØIØ INPUT "ENTREZ VOTRE DONNEE",A
```

Demandons l'exécution du programme précédent ainsi modifié. Nous voyons immédiatement apparaître sur notre terminal.

```
ENTREZ VOTRE DONNEE ?
```

Il suffira alors de rentrer une suite de caractères valides suivies d'un RETURN.

*DATA Il est également possible, lorsque les données ne sont pas susceptibles d'être modifiées, de les spécifier toutes ensembles au début ou en cours d'écriture du programme.
L'instruction permettant d'effectuer cette opération au niveau des données s'appelle DATA.

Exemple:
Soit à rentrer en mémoire les données suivantes dont nous aurons besoin en cours de programme. Il suffira alors de dire:

```
ØØIØ DATA 43,56,12,23,68,132,412
```

Maintenant que nous avons le moyen de rentrer nos données de manière rationnelle, nous devons apprendre à pouvoir les relire.
A cet effet, nous utiliserons l'instruction READ.

Voyons de quelle manière sont stockées les données en mémoire.
Ces dernières sont rangées dans l'ordre dans lequel elles ont été spécifiées. Elles ressortiront également dans le même ordre.

Exemple:

```
ØØIØ DATA 43,56,12,23,68,132,412,817,I7
ØØ2Ø READ A,B,C,D,E,F,G,H,I,
ØØ3Ø PRINT A,B,C
ØØ4Ø PRINT D,E,F
ØØ5Ø PRINT G,H,I
```

Analysons le programme que nous venons d'écrire:

- La ligne N°1Ø a pour effet de rentrer les données en mémoire afin que ces dernières y soient stockées.
- Dans la ligne N°2Ø chaque donnée précédemment spécifiée est associée à une variable numérique, et ceci dans l'ordre où elles ont été écrites.
- Les ligne suivantes ont été écrites dans le but de visualiser

les valeurs numériques associées à chaque variable sous forme de tableau.

Demandons maintenant l'exécution de notre programme:

Nous voyons immédiatement apparaître les données sur notre Terminal.

```
43          56          12
23          68          132
412         817         17

READY
#
```

- La première valeur numérique a été associée à la première variable.
- La deuxième valeur numérique a été associée à la deuxième variable.

En généralisant, nous pouvons donc dire que la Xième donnée sera assimilée à la Xième variable.

Les deux premières lignes du programme que nous venons d'écrire sont équivalentes à :

```
LET A=43
LET B=56
LET C=12
LET D=23
LET E=68
LET F=132
LET G=412
LET H=817
LET I=17
```

Nous voyons immédiatement apparaître la simplicité avec laquelle nous pouvons assigner des valeurs à des variables grâce à cette notation.

Comment procède-t-on si l'on désire qu'une même valeur numérique soit assimilée à deux variables ?
A ce moment là, il faut utiliser l'instruction RESTORE.

Les données sont stockées sous forme de pile, chaque fois que l'on assigne une valeur à une variable à l'aide de l'instruction READ, la donnée lue est alors enlevée de la pile.
C'est donc la donnée suivante qui sera alors disponible.
Grâce à l'instruction RESTORE, il est possible de reconstituer la pile de départ et d'assigner les mêmes valeurs à d'autres variables.
Afin de bien comprendre ce qui se passe, écrivons un petit programme:

```
ØØ1Ø  DATA 43,58,37
ØØ2Ø  READ A,B,C
ØØ3Ø  RESTORE
```

```
ØØ4Ø  READ D,E,F
ØØ5Ø  PRINT A,B,C
ØØ6Ø  PRINT D;E,F
ØØ7Ø  END
```

Demandons l'exécutions de notre programme:

Nous voyons immédiatement apparaître sur notre terminal:

```
43            58            37

43            58            37
```

Les mêmes valeurs numériques ont donc bien été assimilées aux deux groupes de variables.

Que se serait-il passé si nous avions omis d'écrire l'instruction RESTORE.

Essayons, pour voir, d'annuler la ligne N° 3Ø

Pour cela tapez le numéro de ligne(ØØ3Ø) suivi d'un RETURN

Demandez alors l'exécution de votre programme.

Vous voyez immédiatement apparaître sur votre terminal.

```
ERROR Ø3 IN LINE ØØ4Ø
```

En effet, nous demandons l'assignation de données inéxistantes (3 Données) à des variables numériques (6 Variables).

Si l'on utilise pas l'instruction RESTORE, il faut donc que le nombre de DATA soit supérieur ou égal au nombre de READ.

De la même manière, il est possible d'assigner des chaînes de caractères à des variables STRING.

Ecrivons:

```
ØØIØ  DATA "CHAPEAU","BATEAU"
ØØ2Ø  READ A$,B$
ØØ3Ø  PRINT A$,B$
ØØ4Ø  END
```

Nous obtiendrons alors sur notre terminal

```
CHAPEAU          BATEAU
```

Il est également possible de mélanger des variables numériques et des variables STRING.

Exemple:

```
ØØIØ  DATA 12;"BATEAU";72
ØØ2Ø  READ A, A$,B
ØØ3Ø  PRINT A, A$,B
ØØ4Ø  END
```

Nous vous laissons le soin de vérifier que le programme précédent fonctionne et que par conséquent il est bien possible de mélanger des variables numériques et des variables STRING.

Essayons de faire la même chose à l'aide des variables indicées.

A cet effet, écrivons :

```
ØØ1Ø  DIM A(5)
ØØ2Ø  DATA 512,48,53,8,1ØØØ
ØØ3Ø  LET C=1
ØØ4Ø  READ A(C)
ØØ5Ø  PRINT A(C)
ØØ6Ø  IF C=5 THEN END
ØØ7Ø  LET C=C+1
ØØ8Ø  GOTO 4Ø
```

Demandons l'exécution de notre programme ; nous voyons immédiatement apparaître sur notre Terminal.

```
512
48
53
8
1ØØØ

READY
```

Ce dernier a donc parfaitement fonctionné.

REMARQUE :

Dans la ligne numéro 1Ø, nous avons spécifié la dimension de la variable indicée bien que celle-ci soit inférieure à 1Ø.
Cela nous a permis d'économiser de l'espace mémoire ; en effet PROTEUS III a juste réservé l'espace mémoire nécessaire au stockage de 5 données et non pas de 1Ø.

QU'AVONS NOUS APPRIS DANS CE CHAPITRE:

- L'instruction INPUT permet de rentrer des données numériques ou alphanumériques en cours de programme.

- Le fait de rentrer plus de 8I caractères lors d'un INPUT aura pour effet de nous renvoyer à l'erreur Ø9.

- Si un caractère non valide est détecté, PROTEUS III répond REENTER et attend alors la série de caractère valides.

- Si l'on désire imprimer du texte lors d'un INPUT, il faut alors écrire :

```
INPUT "........",X
```

- L'instruction DATA est utilisée conjointement à l'instruction READ pour assigner des données à des variables.

- L'instruction RESTORE permet de réassigner les données à d'autres variables.

- Si l'on ne se sert pas de l'instruction RESTORE, le nombre de DATA doit être supérieur ou égal au nombre de READ, sans quoi nous seront renvoyés à l'erreur Ø3.

IX LES BOUCLES DANS UN PROGRAMME

Nous avons vu dans les chapitres précédents qu'il était possible, dans un programme, d'exécuter un même séquence d'instructions plusieurs fois.

Par exemple nous écrivons :

```
ØØ1Ø  LET A = 1
ØØ2Ø  PRINT "BONJOUR"
ØØ3Ø  IF A = 1Ø THEN END
ØØ4Ø  A=A+1
ØØ5Ø  GOTO 2Ø
```

Dans ce programme long de cinq lignes, quatre d'entre elles servent à demander l'exécution de la ligne ØØ2Ø dix fois de suite. Cette écriture est longue et fastidieuse. Grâce à l'instruction FOR TO utilisée conjointement à l'instruction NEXT, il nous est possible d'alléger considérablement l'écriture de nos programmes.

Réécrivons le programme précédent :

```
ØØ1Ø  FOR A=1 TO 1Ø
ØØ2Ø  PRINT "BONJOUR"
ØØ3Ø  NEXT A
ØØ4Ø  END
```

Demandons l'exécution immédiate de notre programme.

Votre PROTEUS III a-t-il bien affiché BONJOUR dix fois de suite sur votre Terminal ?

Analysons ce que nous venons d'écrire :

- la ligne n° 1Ø demande que l'on fasse varier la valeur assimilée à la variable A entre 1 et 1Ø ; cette variation se faisant par incrémentation de 1.
- la ligne n° 2Ø demande l'impression de BONJOUR.
- la ligne n° 3Ø spécifie que l'on est arrivé à la fin de la boucle et que si la variable A n'a pas atteint sa valeur maximum il faut retourner à l'instruction qui suit immédiatement l'expression FOR TO.
- la ligne n° 4Ø sert à informer votre PROTEUS III que la fin du programme est arrivée.

Grâce à l'utilisation de ces deux instructions, il nous est possible d'effectuer une même séquence plusieurs fois de suite dans un programme.

Comment faire lorsque l'on désire que l'incrémentation de la variable se fasse autrement que de 1 en 1 ?
Il faut utiliser conjointement à l'expression FOR TO, l'instruction STEP. En effet, cette dernière permet de spécifier la valeur dont la variable sera incrémentée.

EXEMPLE : soit à calculer le carré des nombres pairs compris entre 2 et 2Ø.

Le programme s'écrira comme suit:

```
ØØ1Ø  FOR X=2 TO 2Ø STEP 2
ØØ2Ø  PRINT X↑2
ØØ3Ø  NEXT X
ØØ4Ø  END
```

Les résultats demandés s'affichent-ils bien sur votre Terminal?

Si ce n'était pas le cas, vérifiez bien l'écriture de votre programme.

Nous remarquons immédiatement que pour la boucle soit exécutée plusieurs fois de suite, il faut que l'expression située à la gauche de TO soit supérieure à celle située à droite.
Si ce n'était pas le cas, la boucle sera quand même exécutée une fois:/

Exemple:

```
ØØ1Ø  FOR X=3 TO 2
ØØ2Ø  PRINT "BONJOUR"
ØØ3Ø  NEXT X
ØØ4Ø  END
```

Nous vous laissons le soin de vérifier que la boucle ne s'exécute bien qu'une seule fois.

Il est bien sûr possible d'imbriquer plusieurs boucles entres elles, à condition toutefois que ces dernières soient situées à l'intérieur l'une de l'autre, et que leur nombre ne soit pas supérieur à 16.

Exemple:

```
ØØ1Ø  FOR I=1 TO 3
ØØ2Ø  PRINT "BONJOUR"
ØØ3Ø  FOR J=2 TO 5
ØØ4Ø  PRINT "BATEAU"
ØØ5Ø  NEXT J
ØØ6Ø  NEXT I
ØØ7Ø  END
```

Demandons l'exécution de ce programme et voyons ce qu'il se passe sur notre Terminal.

```
BONJOUR
BATEAU
BATEAU
BATEAU
BATEAU
BONJOUR
BATEAU
BATEAU
BATEAU
BATEAU
BONJOUR
BATEAU
BATEAU
```

```
BATEAU
BATEAU

READY
```

Le fait d'inverser les lignes ØØ5Ø et ØØ6Ø nous aurait renvoyés à l'erreur n° 11, en effet, les deux boucles ne seraient plus situées à l'intérieur l'une de l'autre.

Il est aisé grâce à cette instruction de tracer des courbes mathématiques point par point.

EXEMPLE : soit à tracer la courbe représentative de la fonction mathématique $Y = X^2/4$ pour des valeurs allant de - 7 à + 7

Le programme s'écrira alors comme suit :

```
ØØ1Ø  FOR X=-7 TO 7
ØØ2Ø  LET Y=(X↑2)/4
ØØ3Ø  PRINT  TAB (Y);" +"
ØØ4Ø  NEXT X
ØØ5Ø  END
```

Demandez l'exécution de votre programme et vérifiez que ce dernier fonctionne parfaitement.

QU'AVONS NOUS APPRIS DANS CE CHAPITRE

- L'instruction FOR TO utilisée conjointement avec l'instruction NEXT sert à effectuer des boucles dans un programme.

- L'instruction STEP permet de faire varier la valeur dont est incrémentée la variable.

- Il est possible d'imbriquer jusqu'à 16 boucles entre elles, à conditions toutefois que ces dernières soient situées à l'intérieurs les unes des autres.

- Si l'expression situées à gauche de TO est inférieure à celle située à droite, la boucle ne sera exécutée qu'une seule fois.

X LES SOUS PROGRAMMES

Si une même séquence doit ête exécutée en divers endroits d'un programme, il est utile de n'avoir à l'écrire qu'une seule fois.

Nous appellerons cette séquence un sous-programme.

Lorsque nous aurons besoin d'appeller ce sous-programme, nous utiliserons l'instruction GOSUB.

La fin du sous-programme sera définie grâce à l'instruction RETURN.

Exemple:

Soit à calculer le nombre de combinaison de trois cartes prises au hasard dans un jeu comportant 32. Grâce aux mathématiques nous savons que la formule pour trouver ce nombre s'exprime sous la forme suivante:

32!/3! (32-3)! ! désignant la factorielle.

Nous voyons donc que nous avons à calculer 3 fois la factorielle. Il est intéressant dans ce cas de pouvoir écrire un sous-programme et de l'appeller chaque fois que nous en aurons besoin.

Nous écrivons donc:

```
ØØIØ  LET A=32
ØØ2Ø  GOSUB 13Ø
ØØ3Ø  LET X=S
ØØ4Ø  LET A=3
ØØ5Ø  GOSUB 13Ø
ØØ6Ø  LET Y=S
ØØ7Ø  LET A=29
ØØ8Ø  GOSUB 13Ø
ØØ9Ø  LET Z=S
ØIØØ  LET R=(X*Z)/Y
ØIIØ  PRINT R
ØI2Ø  END
ØI3Ø  LET S=1
ØI4Ø  FOR I=1 TO A
ØI5Ø  S=S*I
ØI6Ø  NEXT I
ØI7Ø  RETURN
```

Analysons tout d'abord le sous-programme:

- Ce dernier commence à la ligne 13Ø et se termine à la ligne I7Ø
- Dans la ligne n°13Ø, nous initialisons la variable S qui nous servira à calculer la factorielle.
- Dans la ligne n°14Ø, nous créons une nouvelle variable (I), et nous allons lui faire prendre toutes les valeurs comprises entre 1 et A.
- Dans la ligne n° 15Ø, nous allons multiplier la variable S par l variable I et assimiler ce nouveau résultat à S.
- Dans la ligne n° 16Ø, nous incrémentons la variable I si cette dernière n'a pas atteint sa valeur maximun et nous retournons à l'instruction suivante immédiatement l'expression FOR... TO.
- Dans la ligne n° I7Ø, nous spécifions que le sous-programme est terminé et qu'il faut retourner au programme principal.

Si nous examinons ce sous-programme dans son ensemble, nous voyons bien qu'il calcule l'expression 1x2x3.....xA et en assigne le résultat à S.

Voyons maintenant le programme principal.

- Dans la ligne n° 1Ø, nous assignons 32 à la variable A
- Dans la ligne n° 2Ø, nous demandons l'exécution du sous-programme afin de calculer la factorielle de 32.
- Dans la ligne n° 3Ø, nous assignons le résultat obtenu (la factoriel le) à la variable X
- Dans les lignes dont le numéro d'ordre est compris entre 4Ø et 9Ø, nous calculons les factorielles de 3 et de 29 que nous assignons respectivement aux variables Y et Z.
- Dans la ligne n° 1ØØ, nous calculons le résultat de l'expression 32!/3! (32-3)! que nous assignons à la variable R.
- Dans la ligne n° 11Ø, nous demandons l'impression du résultat sur notre terminal.
- Dans la ligne n° 12Ø, nous spécifions que nous sommes arrivés à la fin de notre programme.

Demandons maintenant l'exécution de ce dernier.

Il fonctionne parfaitement n'est-ce-pas ?

Grâce à cet exemple, l'utilité de pouvoir écrire des sous-programmes nous est immédiatement apparue. En effet, le calcul de la factorielle n'a été écrit qu'une fois, alors que nous l'utilisons trois fois.

Il est bien sûr possible d'imbriquer plusieurs sous-programmes les uns dans les autres; en effet un sous-programme peut en appelller un autre qui en appellera lui-même un autre.

Nous pouvons imbriquer au maximun 16 sous-programmes les uns dans les autres.
Si nous dépassons cette capacité, nous serons renvoyés à l'erreur n° 25 qui se traduit de la manière suivante:

"Imbrication de sous routines excessive (16 au maximun"

Nous avons également la possibilité de créer de nouvelles fonctions mathématiques afin de pouvoir les appeller à tout instant en cours de programme.
Nous pouvons en définir plusieurs car ces dernières se différencient à l'aide d'un caractère alphabétique.
Dans le déroulement du programme nous les appellerons donc par leur nom.

Exemple:
Nous allons définir une fonction qui calculera le quart du carré de l'expression considérée.

Ecrivons:

```
ØØIØ  INPUT A
ØØ2Ø  LET B=FNZ(A)
ØØ3Ø  PRINT B
ØØ4Ø  END
ØØ5Ø  DEF FNZ(X)=(X↑2)/4
```

Analysons ce que nous venons d'écrire:

- Dans la ligne n° 1Ø, nous assimileorns à la variable A, la série de chiffres frappée à partir du clavier.
- Dans la ligne n° 2Ø, nous demandons l'exécution de la fonction que nous avons définie dans la ligne n° 5Ø
- Dans la ligne n°3Ø, nous demandons l'impression du résultat sur notre terminal.
- Dans la ligne n° 4Ø,nous spécifions à PROTEUS III que le programme est terminé.
- La ligne n° 5Ø, défini la fonction.

A REMARQUER:
que la fonction est située après ke END.

Demandons maintenant l'exécution du petit programme que nous venons d'écrire.

Cela marche n'est-ce-pas?

Si, vous n'êtes pas convaincu, essayez à nouveau.

Puisqu'il y a 26 lettres dans l'alphabet, il est possible de définir jusqu'à 26 fonctions utilisateur. Chaque fois que l'on voudra créer une nouvelle fonction, il faudra utiliser l'instruction DEF suivie du nom de la dite fonction et de sa décomposition.

Exemple:
Soit à créer une fonction permettant de calculer le cube de la racine carrée de n'importe quelle impression. Nous appellerons arbitrairement cette fonction FNC(X).

Nous écrivons:

```
IØØØ  DEF FNC(X)=(SQR(X))↑3
```

Dans la suite du programme, chaque fois que nous aurons à calculer le cube d'une racine carrée, il suffira d'appeller la fonction FNC.

Exemple:

```
ØØ2Ø  INPUT A
ØØ3Ø  LET B=FNC(A)
ØØ4Ø  PRINT B
ØØ5Ø  END
```

Si nous essayons d'appeller une fonction utilisateur inexistante, nous serons renvoyés à l'erreur n° Ø8 qui se traduit de la manière suivante:

"Il a été essayé d'appeller une fonction FN inexistante, ou erreur dans la fonction FN"

Remarque:

L'instruction DEF n'est pas pas exécutable.
Elle doit être située à la fin du programme après le END, sinon on a une erreur N° Ø8.

QU'AVONS-NOUS APPRIS DANS CE CHAPITRE

- Il est possible, grâce à l'instruction GOSUB, d'appeller à tout instant des sous-programmes.

- Un sous-programme doit toujours se terminer par l'instruction RETURN

- Il est possible d'imbriquer jusqu'à 16 sous-programmes.

- Nous pouvons, grâce à l'instruction DEF, définir des fonctions utilisateurs.

- Les fonctions utilisateurs s'appellent par FN suivi d'un caractère alphabétique.

- Si votre PROTEUS III détecte une erreur dans la définition ou l'utilisation d'une fonction utilisateur, il vous renverra à l'erreur n° Ø8.

XI UN AUTRE MOYEN D'EFFECTUER DES BRANCHEMENTS

Nous avons vu dans les chapîtres précédents, qu'il était possibl d'effectuer des branchements à l'aide de l'instruction GOTO.
Nous avons vu également que grâce à l'utilisation simultanée des instructions IF THEN et GOTO, il était possible d'effectuer des branchements conditionnels.

Plaçons nous dans le cas où suivant la valeur d'une variable, il faut sauter à différents endroits du programme.

A l'aide des instructions que nous connaissons, nous écrivons par exemple:

```
.
.
.
.
ØØ5Ø IF A=1 THEN GOTO 17Ø
ØØ6Ø IF A=2 THEN GOTO 21Ø
ØØ7Ø IF A=3 THEN GOTO 25Ø
ØØ8Ø IF A=4 THEN GOTO 54Ø
.
.
.
.
etc
```

Toutes les lignes de programme que nous venons d'écrire peuvent se résumer en une seule grâce à l'instruction ON GOTO.
Il suffira alors d'écrire:

```
ØØ5Ø ON A GOTO 17Ø,21Ø,25Ø,54Ø
```

- Lorsque A prend la valeur 1; le branchement s'effectuera au 1er numéro de ligne spécifié après le GOTO.
- Lorsque A prend la valeur 2; le branchement s'effectuera au 2 ème numéro de ligne spécifié après le GOTO.

En généralisant, nous pouvons donc dire:

- Si A prend la valeur N, le branchement s'effectuera au Nième numéro de la ligne spécifié après le GOTO.

Avant d'être prise en compte, la valeur assimilée à la variable A est tronquée (derniers chiffres abandonnés)

```
ØØIØ  LET A=RND(Ø)* 4 +Ø.5
ØØ2Ø  IF A<1 THEN GOTO 1Ø
ØØ3Ø  ON A GOTO 5Ø,6Ø,4Ø,7Ø
ØØ4Ø  PRINT  "A A PRIS LA VALEUR 3":END
ØØ5Ø  PRINT  "A A PRIS LA VALEUR 1":END
ØØ6Ø  PRINT  "A A PRIS LA VALEUR 2":END
ØØ7Ø  PRINT  "A A PRIS LA VALEUR 4":END
```

Analysons ce que nous venons d'écrire:

- Dans la ligne N° 1Ø nous assimilons à la variable A un nombre généré aléatoirement compris entre Ø et 4.5
- Dans la ligne N°2Ø nous retournons à la ligne N° 1Ø si la valeur assimilée à A est inférieure à 1
- Dans la ligne N° 3Ø suivant la valeur de A (comprise entre 1 et4) nous sautons à différentes lignes de programme.
- Les lignes 4Ø à 7Ø sont pratiquement identiques et ont pour seul but de visualiser le resultat de l'opération affectuée en ligne 2Ø

"Pourquoi en ligne 2Ø, avons nous éliminé la valeur Ø me direz-vous

Tout simplement parce qu'en ligne ØØ3Ø, sautez au Ø ième numéro de ligne ne veut rien dire et nous renvoie à l'erreur N° Ø4.
Elle se traduit de la manière suivante; "erreur d'utilisation de l'instruction ON".

Demandons maintenant l'exécution de notre programme:

Celà marche n'est-ce-pas

Relançons le plusieurs fois de suite.

Nous pouvons observer que la valeur prise par A n'est pas toujours la même et que par conséquent nous ne sautons pas systèmatiquement à la même ligne de programme.

De même qu'en cours de programme nous pouvons sauter à des numéros de lignes différents, il est possible de sauter des sous programmes différents grâce à l'instruction ON GOSUB
La procédure d'utilisation est la même que précédemment; toutefois il ne faut pas oublier que les sous programme doivent obligatoirement se terminer par l'instruction RETURN

Exemple:

```
ØØ1Ø INPUT "DONNEZ MOI UNE VALEUR NUMERIQUE",A
ØØ2Ø PRINT " POUR CALCULER LE SINUS, TAPEZ 1"
ØØ3Ø PRINT " POUR CALCULER LE COSINUS, TAPEZ 2"
ØØ4Ø PRINT " POUR CALCULER L'EXPONENTIELLE,TAPEZ 3"
ØØ5Ø PRINT " POUR CALCULER LE LOGARITHME, TAPEZ 4"
ØØ6Ø INPUT B
ØØ7Ø IF B<1THEN GOTO ØØ2Ø<1
ØØ8Ø IF B>4 THEN GOTO ØØ2Ø
ØØ9Ø ON B GOSUB 12Ø,15Ø,13Ø,14Ø
Ø1ØØ PRINT C
Ø11Ø GOTO ØØ1Ø
Ø12Ø LET C=SIN(A): RETURN
Ø13Ø LET C=EXP(A): RETURN
Ø14Ø LET C=LOG(A): RETURN
Ø15Ø LET C=COS(A): RETURN
```

Bien que la plupart des lignes de ce programme vous soient familières, analysons tout de même ce que nous venons d'écrire:

- Dans la ligne N° 1Ø, nous demandons à l'utilisateur de rentrer une donnée à partir du clavier, et assimilons cette dernière à la variable A
- Les lignes dont les numéros sont compris entre 2Ø et 5Ø servent à

donner des instructions à l'utilisateur.
- Dans la ligne N° 6Ø la réponse donnée par l'utilisateur à partir du clavier est assimilée à la variable B
- Les lignes 7Ø et 8Ø détectent les réponses non valides (le chiffre frappé doit être compris entre 1 et 4) et renvoient PROTEUS III à la ligne N° 2Ø
- Dans la ligne N° 9Ø suivant le calcul demandé nous sautons au sous programme correspondant.
- La ligne N° 1ØØ demande l'impression du résultat obtenu
- La ligne N° 11Ø demande à votre PROTEUS III de retourner au début du programme (ligne ØØ1Ø)
- Les sous-programmes situés en ligne 12Ø,13Ø,14Ø,15Ø ont pour effet d'effectuer le calcul demandé par l'utilisateur.

Nous vous laissons maintenant le soin de lancer votre programme et de répondre aux questions qui vous seront posées .

A VOUS !!!!

QU'AVONS NOUS APPRIS DANS CE CHAPITRE

- A l'aide des instructions ON GOTO et ON GOSUB, il nous est possible d'effectuer des branchements conditionnels de manière rationnelle.

- La valeur considérée pour effectuer ces branchements est tronquée avant d'être prise en compte, le contrôle est alors donné à la ligne dont le numéro d'ordre est égal à cette dernière.

- Lorsque nous aurons un numéro d'ordre égal à Ø ou inexistant, nous serons renvoyés à l'erreur N° Ø4

- Lors de l'utilisation de l'instruction ON GOSUB, n'oubliez pas que le contrôle sera donné à un sous-programme et que ce dernier doit obligatoirement se terminer par un RETURN.

XII LA NOTION DE PERIPHERIQUE ASSOCIE

Votre PROTEUS III, dans sa version d'origine peut se connecter à plusieurs appareils qui sont extérieurs à l'unité centrale:
un Ecran de Visualisation, un Magnétophone à cassette, une Imprimante.

- L'Ecran de Visualisation permet de travailler avec l'Unité Centrale par l'intermédiaire du clavier alphanumérique.

- Le Magnétophone à cassette permet à l'utilisateur de recopier ses programmes sur bande magnétique afin de pouvoir les stocker.

- L'Imprimante est utilisée lorsque l'on désire conserver des traces écrites des opérations effectuées à partir de l'Unité Centrale.

Il est possible de travailler simultanément ou séparement avec ces trois périphériques.
Lors de la mise sous tension de PROTEUS III, les commandes sont frappées sur le clavier de l'Unité Centrale et les messages sont émis sur la Console de Visualisation.
Il est néanmoins possible, grâce à l'instruction PORT de choisir le périphérique qui sera utilisé.

Ecrivons par exemple:

```
PORT # 1  RETURN
```

Regardez maintenant votre terminal de visualisation ainsi que votre imprimante (dans le cas où cette dernière est installée); le message READY # est apparu sur chacun des périphériques.

Ecrivez maintènant un petit programme, et demandez son exécution.

Nous pouvons remarquer que l'impression s'effectue sur les deux périphériques de manière simultanée.

Suivant le numéro de PORT spécifié, le ou les périphériques utilisés seront différents.

Le numéro Ø, spécifie que l'on désire utiliser la Console de Visualisation comme périphérique.

Le numéro 2, spécifie que l'on désire utiliser l'Imprimante comme périphérique.

Le numéro 3, spécifie que l'on désire utiliser l'Interface Cassette comme périphérique.

Il est également possible de demander l'utilisation de plusieurs périphériques de manière simultanée.

Lorsque l'on demande PORT#1, nous obtiendrons l'utilisation simultanée de la Console de Visualisation et de l'Imprimante; c'est en fait la combinaison PORT Ø + PORT 2

Lorsque l'on demande PORT#4, nous obtenons l'utilisation simultanée de l'Interface Cassette et de la Console de Visualisation. Voyons cela

plus en détail.

L'entrée se fait à partir de l'Interface cassette avec écho sur la Console de Visualisation.

La sortie se fait simultanément sur la Console de Visualisation et l'Interface cassette.

Lorsque l'on demande PORT #5, l'entrée se fait à partir de l'Imprimante ou la Console de Visualisation avec écho sur l'Interface cassette et la sortie se fait simultanément sur ces trois périphériques.

Il est possible, grâce à l'instruction PORT de changer de périphérique en cours de programme.

Si vous disposez à la fois d'une Console de Visualisation et d'une Imprimante, essayez de rentrer le programme suivant.

```
ØØIØ  PORT #Ø
ØØ2Ø  PRINT "BONJOUR MONSIEUR"
ØØ3Ø  PORT #2
ØØ4Ø  PRINT "COMMENT ALLEZ VOUS"
ØØ5Ø  GOTO 1Ø
```

Demandez maintenant l'exécution de votre programme et observez ce qui se passe.

Chaque phrase est imprimée séparément sur chaque terminal.

- La phrase "BONJOUR MONSIEUR" est imprimée sur la Console de Visualisation.

- La phrase "COMMENT ALLEZ VOUS" est imprimée sur l'Imprimante.

Seule la frappe d'un CTRL C ou l'utilisation de la touche PANIC permet à l'utilisateur de reprendre la main.

Les instructions d'entrée sortie (INPUT;PRINT) peuvent également être suivies d'un numéro de PORT.

Dans le cas du programme précédent, nous aurions pu écrire.

```
ØØIØ  PRINT #Ø,"BONJOUR MONSIEUR"
ØØ2Ø  PRINT #2,"COMMENT ALLEZ VOUS"
ØØ3Ø  GOTO 1Ø
```

Demandons l'exécution de ce petit programme, nous obtenons la même chose que précédemment.

Nous allons maintenant essayer de rentrer une chaîne de caractè à partir du clavier et de l'imprimer sur papier (toujours dans le cas où vous disposez d'une imprimante).

Nous écrivons donc:

```
ØØ1Ø  INPUT #Ø,A$
ØØ2Ø  PRINT #2,A$
ØØ3Ø  GOTO 1Ø
```

N'oubliez pas que si l'on désire traiter des chaînes de caractè supérieures à 18 Caractères, il faut le spécifier à l'aide de l'instruction STRING.

La sauvegarde de programmes sur cassettes ainsi que leur charge ment nécessite l'emploi de trois instructions spécifiques:

SAVE , LOAD , APPEND.

Après vous être assuré que le magnétophone est correctement branché, essayez de sauvegarder le programme précédent.

- Tapez sur le clavier SAVE #3

- Mettez le magnétophone en marche (en enregistrement)

- Attendez quelques instants afin aue l'amorce de la bande soit pas et appuyez sur la touche RETURN.

Après quelques instants, la phrase READY # apparait sur votre terminal.
Le programme contenu en mémoire a donc bien été sauvegardé sur cassette.

Nous allons maintenant effectuer l'opération inverse:

- Avant toute chose, effaçons toute trace du programme résidant en mémoire à l'aide de l'instruction NEW.

- Rembobinez la cassette afin que cette dernière se situe en début bande.

- Tapez LOAD # 3 suivi d'un RETURN et mettez le magnétophone en mar en position lecture.

Dès que le chargement sera terminé, la phrase READY #, apparait sur votre terminal.
Le programme, contenu sur la cassette est maintenant recopié en mémoire.

Demandez son exécution.

Ce dernier fonctionne correctement n'est-ce-pas?

Effaçons le programme résident en mémoire et écrivons.

```
ØØØ5  PRINT #1,"JE SUIS VOTRE PROTEUS III"
ØØ1Ø  END
```

Demandons l'exécution de ce petit programme; l'impession s'effe tue-t-elle bien simultanément sur votre Imprimante et votre Consol de Visualisation ?.

Rembobinez la cassette afin que cette dernière se trouve en début de bande puis tapez APPEND #3 suivi d'un RETURN.

Mettez maintenant votre magnétophone en marche en position Lecture

Dès l'apparition de READY # demandez la lecture du programme résidant en mémoire à l'aide de l'instruction LIST.

Nous voyons immédiatement apparaître:

```
ØØØ5  PRINT #1,"JE SUIS VOTRE PROTEUS III"
ØØIØ  INPUT #Ø,A$
ØØ2Ø  PRINT #2,A$
ØØ3Ø  GOTO 1Ø
```

Essayons à nouveau, en utilisant cette fois l'instruction LOAD.

Ecrivons après avoir rembobiné la bande.

```
LOAD #3
```

N'oubliez pas de mettre le magnétophone en marche en position lecture et appuyer sur la touche RETURN.

Dès l'apparition de READY # indiquant que la procédure de chargement est terminée, demandez l'impression du listing de votre programme.

Vous verrez alors apparaître sur votre terminal:

```
ØØIØ INPUT#Ø,A$
ØØ2Ø PRINT#2,A$
ØØ3Ø GOTO 1Ø
```

Nous pouvons donc en conclure que:

- Lorsque l'on utilise l'instruction LOAD, toute trace de programme résident en mémoire est effacée.

- Lorsque l'on utilise l'instruction APPEND, le programme est chargé sans effacement préalable de l'espace mémoire utilisateur.

Il est également possible de visualiser le programme lors de la sauvegarde ou le chargement de ce dernier.
Il faut alors utiliser le PORT 4 au lieu du PORT 3.

Les instructions servant à commander l'Interface cassette s'écriront alors comme suit:

```
SAVE #4 , LOAD #4 , APPEND #4
```

A titre d'exercice, nous vous conseillons d'écrire un petit programme, de le sauvegarder sur cassette et de le relire en utilisant le PORT 4

ATTENTION:
Lors de l'utilisation des différents périphériques à l'aide des PORTS, il faut prendre garde à ne pas donner le contrôle à un périphérique n'existant pas.

Lorsque l'on travaille uniquement à l'aide de la Console de Visualisation (PORT #0), il est possible de choisir la vitesse d'impression à l'aide de l'instruction SPEED.
Lors de la mise en marche de l'appareil, la vitesse est initialisée à sa valeur maximun (1200 BAUDS) soit 120 caractères par seconde.

Demandons par exemple, une vitesse d'impression à 30 caractères par seconde (300 BAUDS).

Ecrivons:

SPEED =#40 RETURN

Vous pourrez observer en écrivant un petit programme, que la vitesse d'impression sur la Console de Visualisation a changé.

Lorsque l'on travaille avec les autres périphériques (Imprimante, Magnétophone à cassette), l'instruction SPEED n'a aucun effet, ceci parce que chaque périphérique travaille à sa vitesse propre.

Ecrivons un petit programme:

```
#0E0  SPEED =#40
#0F2  PRINT "UN "+CHR$(BELL)
#0F3  SPEED =120
#0F4  PRINT "PRO  ES III"
#0F5  GOTO #0
```

Demandons l'exécution du programme que vous venez d'écrire.

Vous pouvez observer que la chaîne moitié de la phrase s'imprime à une vitesse différente sur la Console de Visualisation.

Demandons maintenant l'impression simultanée sur l'Imprimante et la Console de Visualisation (PORT #

Remarquez maintenant que la phrase est imprimée de manière continue à la même vitesse.

QU'AVONS NOUS APPRIS DANS CE CHAPITRE

- L'instruction PORT permet d'utiliser le ou les périphériques que l'on désire utiliser.

- Les instructions d'Entrée Sortie (PRINT,INPUT) peuvent également être suivies d'un numéro de PORT.

- L'instruction SAVE permet à l'utilisateur de sauvegarder son programme résidant en mémoire sur cassette.

- Les instructions LOAD et APPEND permettent de charger en mémoire un programme contenu sur cassette.

- Lorsque l'on se sert de l'instruction PORT, il faut faire attention au fait que tous les messages seront emis vers ce PORT et que toutes les commandes devront en provenir.

- L'instruction SPEED permet de choisir la vitesse de transmission lorsque l'on travaille avec la Console de Visualisation.

XIII AIDE A LA MISE AU POINT DES PROGRAMMES

Lorsque l'on écrit un programme en BASIC, il peut arriver que l'on fasse des erreurs.

Parfois il est facile de les retrouver, c'est le cas lorsque l'on fait une erreur de syntaxe, puisque le BASIC le signale.

Mais lorsque l'erreur en question est une erreur de raisonnement, le BASIC ne peut s'en apercevoir. Néanmoins, il est capable de vous aider à la rechercher.

C'est l'utilité des fonctions:

```
STOP
TRACE ON
TRACE OFF
```

Voyons cela plus en détail à l'aide d'un exemple concret.

Nous voulons écrire un programme qui range une suite de noms, rentrés n'importe comment, par ordre alphabétique.

Celui-ci s'écrit ainsi:

```
ØØIØ  I=1
ØØ2Ø  PRINT "NUMERO";I;TAB(12);
ØØ3Ø  INPUT A$(I)
ØØ4Ø  IF A$(I)="FIN" THEN 7Ø
ØØ5Ø  I=I+1
ØØ6Ø  GOTO 2Ø
ØØ7Ø  I=I-1
ØØ8Ø  J=Ø
ØØ9Ø  J=J+1
ØIØØ  IF J=I THEN I7Ø
ØIIØ  IF A$(I+1) =A$(I) THEN 9Ø
ØI2Ø  B$ =A$(I)
Ø13Ø  A$(I)=A$(I+1)
ØI4Ø  A$(I+1)=B$
Ø15Ø  J=J-1
Ø16Ø  GOTO 11Ø
Ø165  STOP
Ø17Ø  FOR J=1 TO I
Ø18Ø  PRINT "NUMERO CLASSE";J;TAB(2Ø);A$(J)
ØI9Ø  NEXT J
Ø2ØØ  END
```

Faisons le tourner. Cela donne:

```
READY
# RUN
NUMERO 1      ? A
NUMERO 2      ? D
NUMERO 3      ? C
NUMERO 4      ? B
NUMERO 5      ? FIN
 NUMERO CLASSE 1          A
 NUMERO CLASSE 2          D
 NUMERO CLASSE 3          C
 NUMERO CLASSE 4          B
```

Eh bien, je trouve que ce n'est pas tellement classé. On a cassé le PROTEUS ? Non! et il ne s'est pas trompé.
Simplement le programme ne convient pas.
Pourtant à priori il avait l'air normal. Le début du programme à l'air correct (il est si simple). L'erreur serait-elle plus loin?
On va recommencer et arrêter en 95.
Pour cela, on utilise l'instruction STOP.

```
READY
# 95 STOP
# RUN
NUMERO 1      ? A
NUMERO 2      ? D
NUMERO 3      ? B
NUMERO 4      ? C
NUMERO 5      ? FIN
STOP ØØ95
READY

#
```

On voit alors que le programme s'arrête en 95. Il imprime le message suivant:

```
STOP ØØ95
```

Maintenant on va utiliser la fonction TRACE.
Elle permet d'imprimer le numéro des lignes exécutées par le BASIC.
On a donc:

```
READY
# TRACE ON
```

Le programme repart et imprime:

```
(ØIØØ)
(ØIIØ)
(ØØ9Ø)
(ØIØØ)
(ØIIØ)
(ØØ9Ø)
(ØIØØ)
(ØIIØ)
(ØØ9Ø)
(ØIØØ)
(ØI7Ø)
(ØI8Ø)
 NUMERO CLASSE 1          A
(ØI9Ø)
(ØI8Ø)
 NUMERO CLASSE 2          D
(ØI9Ø)
(ØI8Ø)
 NUMERO CLASSE 3          C
(ØI9Ø)
(ØI8Ø)
 NUMERO CLASSE 4          B
```

On voit donc que le programme fait donc la boucle IØØ-11Ø-9Ø, trois fois avant d'aller imprimer les résultats.
Le test en IØØ marche donc bien, mais quelque chose ne va pas en 11Ø (2ème test).

Sommes-nous bêtes! On teste les A(I) alors que I ne varie pas! C'est A(J) qui nous intéresse.
D'ailleurs dans les lignes 12Ø,13Ø, et 14Ø c'est pareil.

D'abord suprimer la fonction TRACE

```
READY
# TRACE OFF

READY
# ØIIØ IF A$(J+1)> =A$(J) THEN 9Ø
# ØI2Ø B$ =A $(J)
# ØI3Ø A$ (J)=A$ (J+1)
# Ø14Ø A $ (J+1)=B $
```

Voilà si on rentre les valeurs de tout à l'heure le programme marche!.

Essayons en d'autres

```
CRABE
BATEAU
ALLO?
DURAND
```

Et là, il plante! et bien je vous laisse chercher, cette fois-ci. (Pour ceux qui ne trouveraient pas, je leur laisse quand même la solution).

```
READY
# LIST

ØØØ[illegible] A$(7Ø)
ØØIØ  I=1
ØØ2Ø  PRINT "NUMERO";I;TAB(12);
ØØ3Ø  INPUT A$(I)
ØØ4Ø  IF A$(I)="FIN" THEN 7Ø
ØØ5Ø  I=I+1
ØØ6Ø  GOTO 2Ø
ØØ7Ø  I=I-1
ØØ8Ø  J=Ø
ØØ9Ø  J=J+1
ØIØØ  IF J=I THEN 17Ø
ØIIØ  IF A$(J+1)> =A$ (J) THEN 9Ø
ØI2Ø  B$=A$(J)
ØI3Ø  A$(J)=A$(J+1)
Ø14Ø  A$(J+1)=B$
Ø145  IF J=1 THEN9Ø
Ø15Ø  J=J-1
Ø16Ø  GOTO 11Ø
ØI7Ø  FOR J=1 TO I
```

```
ØI8Ø   PRINT "NUMERO CLASSE";J;TAB(2Ø);A$(J)
ØI9Ø   NEXT J
Ø2ØØ   END

READY
#
```

LES CODES ERREURS

Au cours de la lecture de ce manuel, vous avez pu vous rendre compte du fait que votre PROTEUS III est capable de détecter de lui même certaines erreurs, et d'en informer l'utilisateur.

Le message qui est alors emis est de la forme suivante:

ERROR = XX IN LINE YYYY

YYYY désigne le numéro de la dernière ligne traitée.

XX désigne le numéro de l'erreur détectée.

Il faut alors vous reporter au tableau ci-après pour connaître la nature exacte de l'erreur détectée.

Il ne vous reste alors plus qu'à examiner et à corriger la ligne spécifiée.

ØØ Il ne reste plus de place dans l'espace mémoire alloué à l'utilisateur.

Ø1 Le numéro de ligne spécifié n'a pas été trouvé.

Ø2 Instruction non identifiable (le terme ne fait pas partie du langage BASIC)

Ø3 Erreur d'utilisation de l'instruction READ

Ø4 Erreur d'utilisation de l'instruction ON

Ø5 Erreur d'utilisation de la fonction VAL; la variable STRING considérée ne commence pas par une variable numérique.

Ø6 Erreur de syntaxe ou utilisation de l'instruction FOR

Ø7 Erreur de syntaxe ou d'utilisation de la fonction DEF

Ø8 Il a été essayé d'appeler une fonction FN inexistante ou erreur dans la fonction FN.

Ø9 Il a été essayé de rentrer plus de 81 caractères alphanumériques lors d'une instruction INPUT

1Ø Des caractères non valides ont été rentrés lors d'un INPUT

11 Erreur d'imbrication de boucles FOR.... NEXT

12 Nombre de boucles FOR....NEXT trop important

13 Le BASIC a rencontré l'instruction NEXT avant l'instruction FOR

14 La variable ou le caractère rencontré est illégal

15 Variable trop importante (supérieure à 255) en mode TAB,CHR$ ou ON

16 La variable rencontrée est illégale

17 Erreur d'utilisation de l'instruction STRING, ou mélange de variables alphanumériques et numériques

18 Variable STRING trop importante, ou erreur d'utilisation des fonction LEFT$, RIGHT$,MID$

19 Les " de fermeture ont été oubliés dans l'impression d'une chaîne de caractères

2Ø Erreur dans l'utilisation des parenthèses

21 Il a été essayé de trouver le LOG d'un nombre négatif

22 Il a été essayé d'effectuer une division par Zéro

23 Arithmétique illégale

24 Le BASIC a rencontré l'instruction RETURN avant l'instruction GOSUB

25 Instruction excessive de sous-programmes (16 au maximun)

26 Il a été demandé un ER (Exécutive Request) sans carte "Operating System" installée

27 Le PORT spécifié n'a pas de carte communication installée

28 Erreur d'utilisation de l'instruction DIM

29 Erreur dans l'utilisation d'un tableau.

LISTE ALPHABETIQUE des COMMANDES

Afin de vous éviter des fastidueuses recherches, nous avons regroupé dans ce chapître toutes les Instructions que vous avez rencontrées au cours de la lecture de votre Manuel.

- Instructions pouvant être exécutées directement

APPEND : est utilisée pour charger un programme sans effacer le contenu de la mémoire.

CONT : est utilisée pour relancer un programme utilisateur.

CTRL C : est utilisée pour interrompre le déroulement du programme utilisateur.

CTRL O : est utilisée pour effacer le dernier caractère frappé.

CTRL X : est utilisée pour annuler une ligne en cours d'écriture.

DIGITS : est utilisée pour spécifier le nombre de décimales.

ER : est utilisée pour rendre la main à l'opérating System, si ce dernier est installé.

INCR : est utilisée pour spécifier la valeur de l'incrément en numérotation automatique.

LINE : spécifie le nombre de caractère que l'on désire voir s'afficher sur une ligne.

LIST : permet à l'utilisateur de relire son programme.

LOAD : permet le chargement d'un programme.

NEW : à pour effet d'effacer toute trace de programme contenu en mémoire.

PORT : permet de choisir le périphérique de contrôle.

RUBOUT : demande la numérotation automatique d'une ligne.

RUN : demande l'exécution du programme utilisateur.

SAVE : permet de l'utilisateur de sauvegarder son programme.

SPEED : permet de choisir la vitesse d'impression sur le Terminal Vidéo.

STRING : spécifie le nombre maximun de caractères que pourra contenir une chaîne.

TRACE ON : initialise le mode trace.

TRACE OFF : arrête le mode trace

- Instructions rentrant dans la composition d'un programme.

DATA	: permet à l'utilisateur de introduire des données.
DEF	: permet à l'utilisateur de définir ses propres fonctions.
DIM	: réserve de l'espace mémoire pour le stockage de variables
END	: détermine la fin du programme utilisateur.
FOR TO	: permet d'effectuer des boucles dans un programme.
GOSUB	: permet de sauter un sous-programme.
GOTO	: permet d'effectuer un branchement.
IF THEN	: permet d'effectuer des comparaisons.
INPUT	: permet à l'utilisateur de rentrer des données en cours de programme.
LET	: permet d'assimiler une valeur à une variable.
NEXT	: détermine la fin d'une boucle (voir FOR TO).
ON GOSUB	: permet de sauter à plusieurs sous-programmes suivant la valeur d'une variable.
ON GOTO	: permet d'effectuer divers branchements suivant la valeur d'une variable.
PRINT	: permet d'imprimer du texte ou des variables.
READ	: permet de lire des données stockées en mémoire. (voir DATA)
REM	: permet de glisser des remarques ou des explications en cours de listing d'un programme.
RETURN	: détermine la fin d'un sous-programme.
RESTORE	: permet de relire des données précédemment lues grâce à l'instruction READ.
STEP	: permet de choisir la valeur de l'increment dans une boucle (voir FOR TO).
STOP	: permet d'interrompre le déroulement du programme.

- Les FONCTIONS

ABS : considére la valeur absolue.

ASC : permet de connaître le caractère alphanumérique assimilé à une valeur décimale.

ATAN : permet de calculer l'arc tangente d'une valeur ou d'une expression.

CHR$: permet de considérer la valeur numérique assimilée à un caractère alphanumérique.

COS : permet de calculer le cosinus d'une valeur ou d'une expression.

DEG : permet de faire une conversion de radians en degrés.

DLOG : permet de calculer le logarithme décimal d'une valeur ou d'une expression.

EXP : permet de calculer l'exponentielle d'une valeur ou d'une expression.

FN : permet d'appeler une fonction définie par l'utilisateur.

INT : permet de considérer la valeur entière d'une expression.

LEFT$: considère un nombre déterminé de caractères en partant de la gauche.

LEN : permet de connaître le nombre de caractères contenus dans une chaîne .

LOG : permet de calculer le logarithme neperien d'une valeur ou d'une expression.

MID$: permet de considérer un nombre quelconque de caractères contenu dans une chaîne.

PEEK : permet de lire le contenu d'une case mémoire.

POKE : permet d'écrire dans une case mémoire.

POS : permet de connaître à tout instant la position de la tête d'impression.

RAD : permet de faire une conversion de degrés en radians.

RIGHT$: permet de considérer un nombre quelconque de caractère contenus dans une chaîne en partant de la droite

RND : permet d'obtenir un nombre aléatoire.

SIN : permet de calculer le sinus d'une valeur ou d'une expression.

SGN : permet de connaître le signe d'une valeur ou d'une expression.

SQR : permet de calculer la racine carrée d'une valeur ou d'une expression.

STR$: permet d'introduire une valeur numérique dans une chaîne de caractères.

TAB : permet d'effectuer de la mise en page.

TAN : permet de calculer la tangente d'une valeur ou d'une expression.

VAL : permet de considérer un nombre contenu dans une chaîne de caractère.

- Les Opérateurs

↑	: élévation à la puissance.
-	: négation.
*	: multiplication.
/	: division.
+	: addition.
-	: soustraction.
=	: égal.
<>	: différent.
<	: plus petit que
>	: plus grand que.
<=	plus petit ou égal.
>=	plus grand ou égal.

ENTRETIEN de L'APPAREIL

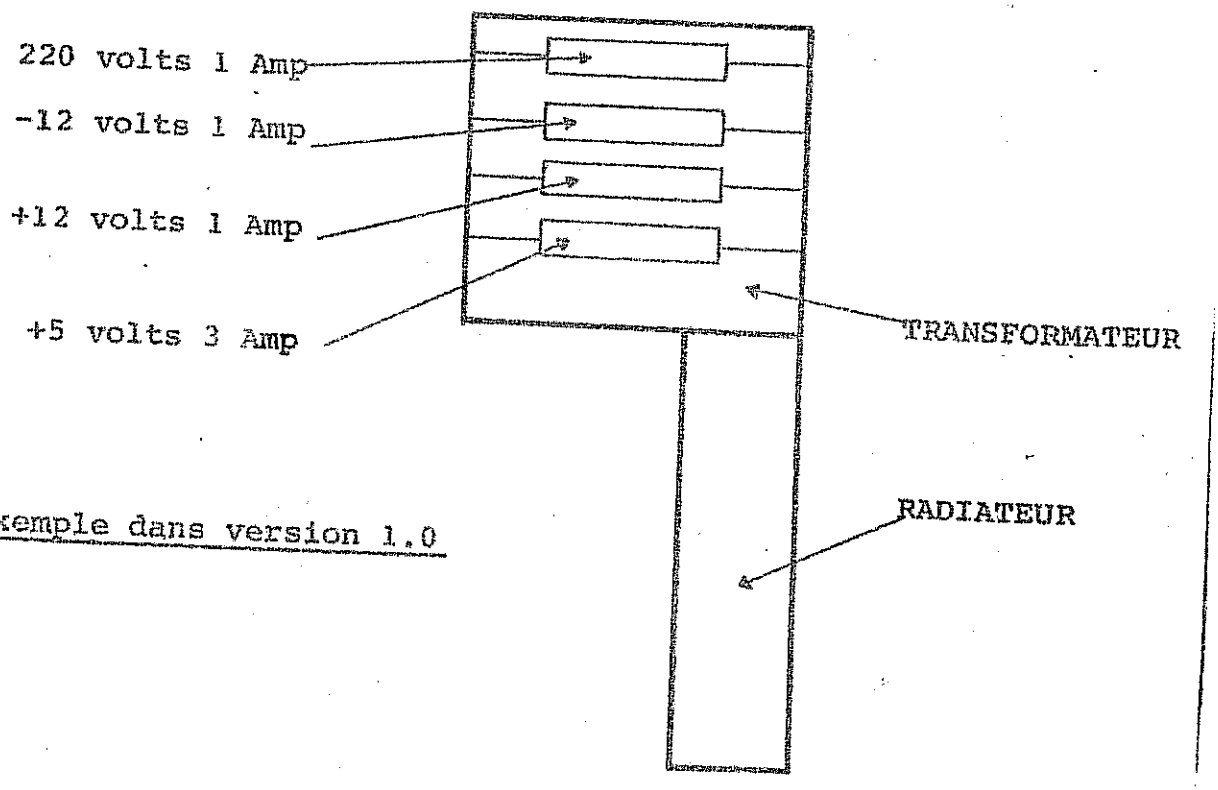
- Pour nettoyer le coffret de votre PROTEUS III, nous vous conseillons d'utiliser un chiffon légérement imbibé d'eau ou de produit d'entretien pour les vitres.

ATTENTION, n'utilisez pas de produits corrosifs (Trichlorotylène, etc....)

- Ne laissez pas la poussière se déposer sur le clavier, car celà pourrait causer un mauvais fonctionnement des touches.

- En cas de non-fonctionnement de votre appareil, Vérifiez les fusibles.

Ces derniers sont situés à l'intérieur du boitier sur le transformateur d'alimentation.



*:Par exemple dans version 1.0