TOUT SUR
Le PicBasic 3B et 3H
Sous PICBASIC STUDIO (XP ©)

Sommaire

I PRÉSENTATION

a) Description:

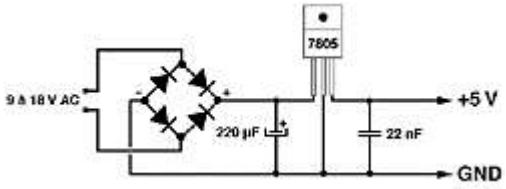
Les PICBASIC se composent d'un microcontrôleur associé à une mémoire non volatile (EEPROM ou FLASH) et se programment très facilement en langage "BASIC" par l'intermédiaire d'un compatible PC et d'un logiciel de développement PICBASIC-LAB qui transformera vos instructions "BASIC" en codes spécifiques, lesquels seront alors transférés dans la mémoire du "PICBASIC" par le biais d'un cordon de liaison spécial préalablement raccordé au port imprimante de votre ordinateur. Une fois le PICBASIC ainsi "chargé", ce dernier pourra être déconnecté du "PC" pour devenir autonome afin de réaliser votre programme par le biais de son microcontrôleur qui récupérera un à un les codes transférés pour les "traduire" en "action" adéquate.

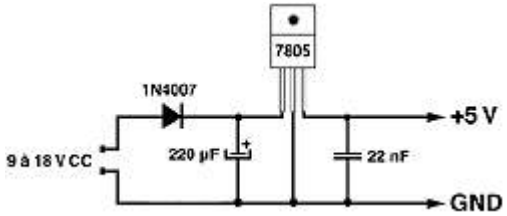
Séries	Série 1		Série 2		Série 3		Série 2000	
Modèle	PB-1B	PB-1S	PB-2S	PB-2H	PB-3B	PB-3H	PBM-R1	PBM-R5
Photo								
Mémoire Programme	2 K (EPROM)	4 K (EPROM)	8 K (EPROM)	16 K (EPROM)	4 K (FLASH)	4 K (FLASH)	64 K (FLASH)	64 K (FLASH)
RAM	96 octets	96 octets	96 octets	96 octets	80 oct	80 octets	8 K	32 K
EEPROM	Comprise dans mémoire programme	Comprise dans mémoire programme	Comprise dans mémoire programme	Comprise dans mémoire programme	-	-	8 K	32 K
Nb broches / boîtier	22 / S.I.L	22 / S.I.L	34 / D.I.L	34 / D.I.L	28 / D.I.L	40 / D.I.L	40 / D.I.L	40 / D.I.L
Nb "E/S" Horloge/ calendrier	16 NON	16 NON	27 NON	27 NON	18 NON	29 NON	34 NON	34 OUI
Convertisseur A/N	-	5 (rés. 8 bits)	8 (rés. 8 bits)	8 (rés. 8 bits)	5 (10 bits)	8 (10 bits)	8 (10 bits)	8 (10 bits) 2 (12 bits)
codes/sec traités	1000	1000	1000	5000	56.000	56.000	40.000	40.000
Dim. (mm)	57 × 27 × 9	57 × 27 × 9	45 × 25 × 15	45 × 25 × 15	DIL 28 étroit	DIL 40 standard	75 × 65 × 16	75 × 65 × 16

b) interfaçage des PICBASIC :

1) Alimentation:

Tous les modules "PICBASIC" doivent être alimentés sous une tension de + 5 V. Le pont redresseur peut être remplacé par une simple diode de protection contre les inversions de polarité, si la tension d'entrée est continue. Enfin le 7805 peut être remplacé par un 78L05 (plus petit), si la consommation totale de l'application n'excède pas 100 mA.





2) Entrée "RESET":

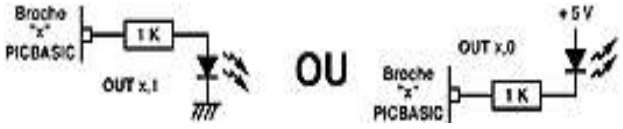
La broche "RESET" des modules "PICBASIC" doit simplement être reliée au +5 V.

3) Recommandations:

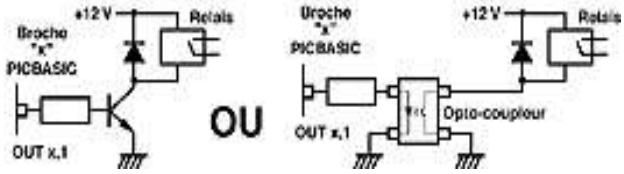
Chacune des broches "I/O" des "PICBASIC" peut indépendamment être configurée pour être utilisée en entrée ou en sortie. Certaines peuvent également faire office d'entrée dans le cadre d'une conversion analogique/numérique.

4) Commande de dispositifs externes:

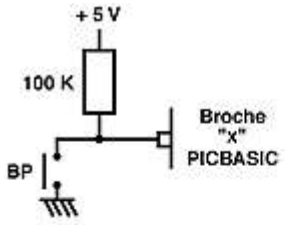
Chaque broche des modules "PICBASIC" peut piloter (lorsqu'elle est utilisée en sortie), un dispositif dont la consommation ne devra pas dépasser les 25 mA (commande par apport de + ou de - grâce à l'instruction OUT x,1 ou OUTx,0 - ou x représente le N° de la broche du "PICBASIC"). Il est ainsi très facile de piloter directement une Led comme indiqué sur les 2 schémas ci-contre.



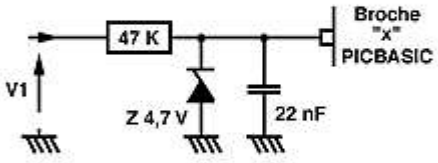
Si la consommation des dispositifs à piloter devait dépasser les 25 mA, il conviendra alors d'avoir impérativement recours à l'utilisation optocoupleurs ou de relais d'interfaçage (voir exemples de schémas ci-contre).



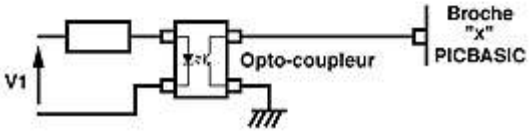
5) Gestion des broches en entrée:
La "lecture" de contacts externes par les broches des modules "PICBASIC" est très simple. Dans le cadre de boutons-poussoirs ou d'interrupteurs, il suffira de réaliser le schéma ci-contre. La longueur des câbles reliant les boutons-poussoirs au "PICBASIC" ne devra pas excéder quelques dizaines de centimètres. En cas contraire, il conviendra d'utiliser une interface adéquate (avec des optocoupleurs par exemple) afin d'éviter que des parasites ne "remontent" par les câbles et ne provoquent des perturbations ou dans certains cas extrêmes n'endommages les broches du "PICBASIC".



Si vous devez interfacer les "PICBASIC" avec des tensions supérieures à + 5 V, il conviendra d'utiliser un des 2 schémas donnés ci-contre.



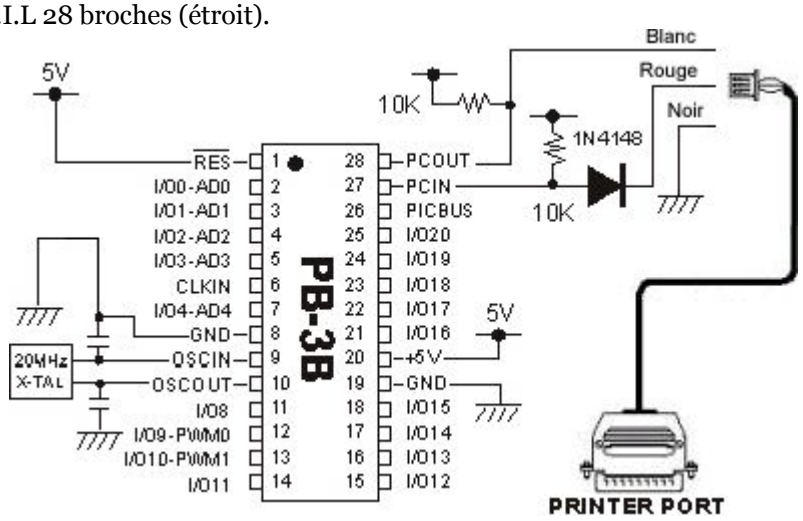
Dans le cadre de mesures de valeurs analogiques dont la tension maximale serait supérieure à + 5 V, il conviendra d'avoir recours à l'utilisation d'un pont diviseur en s'assurant toujours que la tension en entrée du "PICBASIC" ne dépasse jamais les + 5 V.



c) Le PICBASIC 3B

Il se présente sous la forme d'un simple circuit intégré nécessitant seulement 1 quartz de 20 MHz, 2 condensateurs, 2 résistances et une diode nécessaires à sa mise en œuvre. La mémoire programme Flash peut également être utilisée pour stocker des données non volatiles.

* 4 K mémoire FLASH programme données.
* 80 octets de RAM.
* 18 Entrées / Sorties dont 5 pouvant être utilisées en entrée/sortie standard ou en entrée de conversion analogique/numérique 10 bits.
* Nombre de codes traités à la seconde: 56.000.
* Circuit intégré D.I.L 28 broches (étroit).



Broche	Désignation	Bloc = Port 16F876 A=0 B=2 C=1	Fonction
20 1 8 9/10	+5V RES GND OSCIN/OSCOUT		Alimentation Reset Quartz
2 3 4 5 7	I/O 0-AD0 I/O 1-AD1 I/O 2-AD2 I/O 3-AD3 I/O 4-AD4	0 (TTL) 0 (TTL) 0 (TTL) 0 (TTL) 0 (TTL)	E/S ou CAN E/S ou CAN E/S ou CAN E/S ou CAN E/S ou CAN
11 12 13 14 15 16 17 18	I/O 8 I/O 9-PWM0 I/O 10-PWM1 I/O 11 I/O 12 I/O 13 I/O 14 I/O 15	1 (ST) 1 (ST) 1 (ST) 1 (ST) 1 (ST) 1 (ST) 1 (ST) 1 (ST)	E/S E/S ou PWM E/S ou PWM E/S E/S E/S E/S E/S
21 22 23 24 25	I/O 16 I/O 17 I/O 18 I/O 19 I/O 20	2 (ST) 2 (ST) 2 (ST) 2 (ST) 2 (ST)	E/S E/S E/S E/S E/S
6 26 27/28	CLKIN PICBUS PCIN/PCOUT	(ST)	Entrée de comptage Cde afficheur série Utilisés pour la programmation

II UTILISATION DU LOGICIEL PICBASIC STUDIO

TOUJOURS COUPER L'ALIMENTATION DU PIC POUR LE CONNECTER OU DECONNECTER D'UN ORDINATEUR.

C'est grâce à ce logiciel que vous pourrez programmer tous les modules PICBASIC .
Pour utiliser le programme PICBASIC STUDIO, il faut tout d'abord et IMPERATIVEMENT créer une imprimante HP LASER 4 (non prise par défaut) sous Windows.

Lancer le programme : PICBASIC Studio .

a)Description des différents menus

→: Menu File

'New' : Création d'un nouveau fichier.

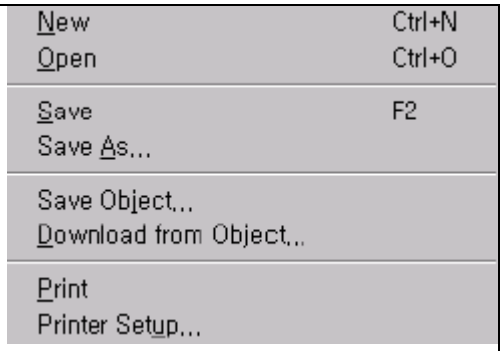
'Open' : Ouvrir un fichier existant .

'Save' : Sauver le fichier en cours. Le 'Save as' permet de changer le nom du programme.

'Save Object' : Sauve unuquement le code objet du fichier en cours.
'Download from object' : Charge uniquement le code objet .

'Print' : Imprimer

'Exit' : Sortie



→ **Menu EDIT** et **SEARCH**: menu édition et recherche habituel de Microsoft

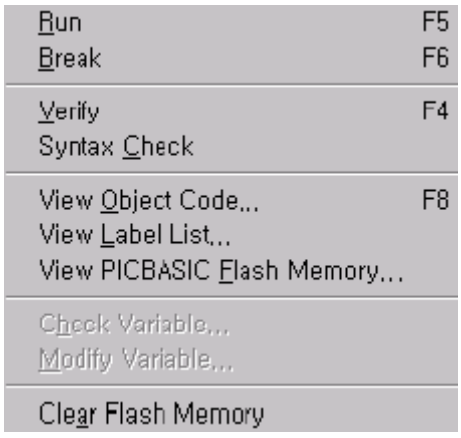
→ Menu RUN

'Run' : Exécute SyntaxCheck , compile et Transfère le programme à l'écran dans le PICBASIC et l'exécute

'Break' : Arrête l'exécution du programme dans le PICBASIC et passe en mode DEBUG.

Les 'View' permettent de visualiser le code objet , l'état des mémoires, du programme téléchargé,…

'Syntax Check' : Permet de vérifier la syntaxe du programme.



→ Menu SETUP

'Printer Port Setup' : Pour définir le port parallèle utilisé .

III **Variables , constantes, opérations, formats**

a) Déclaration des variables des PICBASIC

Avant de pouvoir utiliser une variable dans laquelle vous pourrez stocker des informations, il vous faudra au préalable déclarer celle-ci au module "PICBASIC" afin qu'il réserve de la place au sein de sa mémoire "RAM". 2 Types de variables sont déclarables.

Les variables de type "BYTE" qui pourront correspondre à un nombre compris entre 0 et 255 (elles occuperont 1 octet de mémoire RAM) et les variables de type "INTEGER" qui pourront correspondre à un nombre compris entre 0 et 65535 (elles occuperont 2 octets de mémoire RAM).

La déclaration se fera à l'aide de l'instruction "DIM". On déclarera généralement toutes les variables en début de programme.

ATTENTION: Les noms choisis par vos variables ne doivent pas correspondre ou commencer par des mots attribués aux instructions des "PICBASIC".

b) Déclaration des constantes

Il est possible d'attribuer un "nom" à une constante. La déclaration se fera à l'aide de l'instruction "CONST".

L'instruction "CONST" permet également de définir des "tableaux" de valeurs qu'il vous sera ensuite possible de récupérer très facilement. Dans l'exemple N° 2, on pourra attribuer plusieurs valeurs à la constante "VALEUR" en fonction de la valeur de la variable avec laquelle elle sera appelée.

Exemple N° 1
```
10 CONST bro2 = 2
20 OUT bro2,1            ' Applique 1 à la sortie 2
```

Exemple N° 2
```
10 CONST BYTE VALEUR = (31, 26, 102, 34, 65)
20 DIM A AS BYTE
30 DIM B AS BYTE
40 A = 0
50 B = VALEUR ( A ) ' B = 31
60 A = 2
70 B = VALEUR ( A ) ' B = 102
```

Exemple N° 3
```
10 CONST BYTE VALEUR = ("BTS Electronique", 25, 49)   (les valeurs seront le code ASCII des lettres).
```

Exemple N° 4
```
10 DIM I(10) AS BYTE     '( pas INTEGER)
20 I(0) = 45
30 I(1) = 56
40 ...
```

c) Opérations sur les variables et les constantes

Il est possible d'utiliser toutes sortes d'opérations sur les variables et constantes à l'aide des "opérateurs" habituels:

+ (addition)	AND ("et" logique)	< (inférieur ...)	= (égal)
- (soustraction)	OR ("ou" logique")	> (supérieur ...)	<> (différent)
* (multiplication)	XOR ("ou exclusif" logique")	<= (inférieur ou égale ...)	
/ (division).	MOD ("Modulo")	>= (supérieur ou égale ...)	

Il est possible de travailler en format Décimal (58), hexadécimal (&hFE) et binaire (&b10111001).

I = I << n décalage de n bit de la valeur de la variable "I" vers la gauche (>> = droite).
I = (I<<1) ou I.7 Idem avec récupération du bit de poids fort, "réinjecté" à la place du bit de poids faible.
I.7 = 1 'le bit de poids fort est forcé à 1.
I.H =12 'l'octet de poids fort est forcé à 12. (si I est INTEGER)
I.L = &H5 'l'octet de poids faible est forcé à &H5. (si I est INTEGER)

IV LISTE DES INSTRUCTIONS

Au début de du programme devra apparaître :
CONST DEVICE= XX **Avec XX= 3B ou 3H selon le PICBASIC utilisé.**

Entrées/Sorties
IN
BYTEIN
OUT
BYTEOUT
OUTSTAT
TOGGLE
PULSE

Entrées/Sorties
ADIN
PWM
PWMOFF

Gestion touches
ADKEYIN
PADIN
KEYIN
KEYDELAY

Sauts/Conditions
IF ... THEN... ENDIF
FOR ... NEXT
GOTO
ON ... GOTO
GOSUB ... RETURN

Génération sonore
SOUND
BEEP
PLAY

Gestion "RS232"
SERIN
SEROUT

Adressage "I2C/SPI"
SHIFTIN
SHIFTOUT

Gestion "EEPROM"
EEWRITE
EEREAD

Interruption
ON TIMER ... GOSUB
ON INT ... GOSUB
ON INT(5) ... GOSUB

Gestion "LCD"
LCDINIT
CLS
LOCATE
PRINT
PRINT DEC
PRINT HEX
CSRON
CSROFF
BUSOUT
SET PICBUS

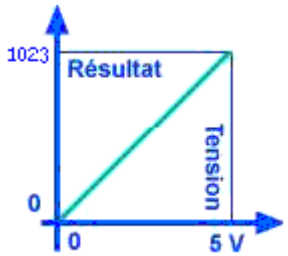
Divers
FREQOUT
CAPTURE
STEPOUT
SERVO
CONST
RND
TABLE
BREAK
COUNT
PEEK
POKE
DELAY
BCD

Par convention, on donnera la syntaxe de chaque instruction à l'aide de "mots clefs" qui vous permettront de rapidement comprendre et assimiler leur structure.

"Ligne": désigne un N° de ligne de votre programme
"Port": désigne un N° de broche (I/O) du module "PICBASIC".
"Val" (ou "Val1" ou "Val2"): désignera une valeur à indiquer.
"Var": désigne une variable dont la valeur est modifiable en cours d'exécution du programme.
"Param" (ou "Param1" ou "Param2" ou "Param3"): désigne un paramètre à indiquer.
"Adr": (ou "Adr1" ou "Adr2") désigne une adresse à indiquer.
"Express": désigne une expression.

ADIN (Port)

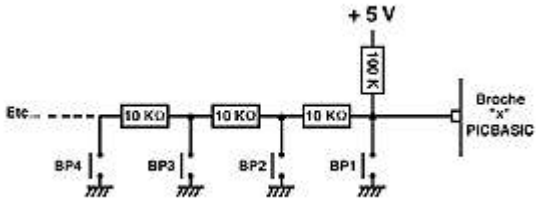
Cette instruction permet de connaître la valeur de la tension analogique présente sur une broche particulière du module "PICBASIC" . La valeur à lire doit être comprise entre 0 et + 5 V . Le paramètre (Port) correspond à la broche du module qui recevra la valeur à mesurer.



Exemple:
10 DIM VALEUR AS INTEGER
20 VALEUR = ADIN (3) (le convertisseur est 10 bits, valeur entre 0 et 1023)

ADKEYIN (Port)

Cette instruction permet grâce à l'utilisation d'une seule broche de conversion analogique/numérique de connaître l'état de 1 à 10 boutons-poussoirs ! Pour ce faire, il suffira de réaliser le montage ci-contre (référez-vous au début pour connaître les pattes bénéficiant d'une conversion analogique/numérique).



Le principe de fonctionnement est en fait très simple et repose sur celui des ponts .Lorsque "BP2" est sollicité à son tour, on se retrouve avec une tension de 450 mV et la valeur "2" est retournée En absence de sollicitation des boutons poussoirs, la valeur retournée est "0".

BCD (Val) ou BCD (Var)

Cette instruction permet de convertir le format d'un nombre en sa "formulation" en "Binaire-Codé-Décimal". Maximum 99 en format BYTE et 999 en format INTEGER
Exemple :

```
10 DIM VALEUR AS BYTE
20 DIM RESULTAT AS BYTE
30 VALEUR =&H63
40 RESULTAT = BCD(VALEUR)
```

BEEP Port

Cette instruction génère un signal carré de fréquence 4 KHz pendant quelques millisecondes sur une broche du module "PICBASIC". En connectant un buzzer (sans oscillateur) sur cette broche, un "bip" sonore se fait alors entendre (voir schéma ci-contre). Le paramètre "Port" indique la broche où sera connecté le buzzer.
Exemple : BEEP 4

BREAK

Cette instruction très utile est utilisée dans le cadre de la mise au point de vos programmes. Lors de déroulement de ce dernier, si vous placez cette instruction alors que le module est relié à votre PC via son cordon de téléchargement, l'éditeur stoppe alors le fonctionnement du programme et vous place automatiquement sur la fenêtre "DEBUG" afin que vous puissiez vérifier les valeurs de toutes les variables et éventuellement tester votre programme en mode "pas-à-pas". Si le module n'est pas relié au PC et qu'une instruction BREAK est effectuée, le "PICBASIC" après un léger temps d'arrêt, continu le déroulement de son programme. Dans tous les cas il conviendra, une fois le programme complètement mis au point de supprimer toutes les instructions "BREAK" de son listing.

BUSOUT Val1, Val2, Val3 ...

La broche "PICBUS" de chaque module "PICBASIC" est spécialement conçue pour piloter des afficheurs à commande série par le biais d'instructions spécifiques qui envoient une série d'ordres à ces derniers. L'utilisateur a également la possibilité de piloter ces afficheurs série en utilisant l'instruction "BUSOUT".

Exemple: 10 BUSOUT &HA1, &H00, &H00, &HA2, &H41, &H42, &H43, &H44, &H00
Revient à locate 0,0 et print « ABCD »
Les signaux générés par les instructions spécialisées pour la gestion des afficheurs séries ou l'instruction "BUSOUT" sont de type RS232 (5 V / niveau TTL). Le débit 4800 ou 19200 Bds est fonction de la déclaration faite par les instructions "PICBUS HIGH" ou "PICBUS LOW".
On peut se servir de busout avec une variable à envoyer, pour par exemple utiliser la broche picbus pour envoyer des données série .

BYTEIN (Param)

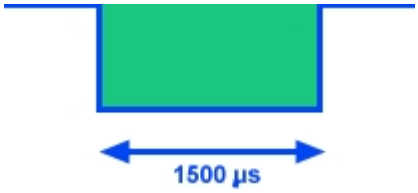
Cette instruction permet de récupérer la valeur de 8 entrées (bloc 1) du module "PICBASIC" dans un "mot binaire" dont chaque bit est l'image de chacune des entrées. Il est ainsi possible de gérer 1 bloc 8 bits différents avec "Param" égal à 1. Voir l'affectation des broches / blocs . (possibilité d'utiliser le bloc 0 et 2, mais seulement sur 5 bits , voir Picbasic 3b)

BYTEOUT Port, Val ou BYTEOUT Port, Var

Cette instruction permet de sortir la valeur binaire d'une donnée (Val) sur un bloc du "PICBASIC".

CAPTURE (Port, Param)

Permet de mesurer la durée d'une impulsion d'un signal extérieur (niveau haut ou bas selon la valeur de "Param") sur une broche (Port) du "PICBASIC". Le nombre récupéré est soumis à un facteur de réduction. Ainsi pour le "PICBASIC-3B", il doit être multiplié par "4". Ceci veut dire qu'il ne sera pas possible de mesurer des largeurs d'impulsions supérieures supérieures à 0,26s (le calcul est très simple: Nombre max.= 65535 x 4 = 262140µs).



La valeur retournée est 1500/4 = 375

Exemple :
10 DIM I AS INTEGER
20 I=CAPTURE (0,1)
'mesure la largeur de l'impulsion haute sur I/O 0'

CONST EXPRESS = Val

Cette instruction permet de remplacer un constante (Val) par une expression (EXPRESS) au sein de votre programme afin d'en faciliter la lecture et la mise en œuvre.
Exemple :
10 CONST BUZZER = 5
20 BEEP BUZZER 'un BEEP est produit sur la I/O 5'

COUNT (Param)

Cette instruction permet de compter le nombre d'impulsions présentent sur l'entrée spécifique "CLKIN" du module "PICBASIC" (idéale pour connaître la fréquence d'un signal carré, le nombre d'impulsions générées par un système externe...). Il est possible, suivant la valeur du paramètre (Param), 0 ou 1 de déterminer une remise à zéro automatique du compteur à chaque exécution de l'instruction. A noter que ce comptage s'effectue en "tâche" de fond, c'est à dire en même temps que le déroulement de votre programme (sans que vous n'ayez à le gérer) et que c'est au moment où vous "appelez" l'instruction COUNT, que vous récupérez le nombre d'impulsions comptabilisées. Les impulsions sont comptabilisées à chaque front montant du signal: passage du niveau logique "0" (0V) au niveau logique "1" (+5V) - La fréquence de comptage maximale est de l'ordre de 20 KHz. Si la variable qui reçoit le compteur doit être déclarée en BYTE, le résultat sera un nombre compris entre 0 et 255. En cas de dépassement du compteur, celui-ci repasse à zéro. A la mise sous tension du "PICBASIC", le compteur est initialisé à 0.
Exemple :
10 DIM COMPT AS BYTE
'déroulement du programme
COMPT = COUNT(0) 'la variable COMPT contiendra le nombre d'impulsions depuis la mise sous tension du PICBASIC, si la fonction COUNT est de nouveau appelée, le nombre de nouvelles impulsions s'ajoutera à celles déjà comptabilisées'

COMPT = COUNT(1) 'la variable COMPT contiendra le nombre d'impulsions depuis la mise sous tension du PICBASIC, et le compteur est remis à zéro'

CSRON

Cette instruction permet lorsque le "PICBASIC" est relié à un afficheur LCD "série" via son port "PICBUS" d'activer l'apparition du curseur.

CSROFF

Cette instruction permet lorsque le "PICBASIC" est relié à un afficheur LCD "série" via son port "PICBUS" de désactiver et de faire disparaître le curseur.

CLS

Cette instruction permet lorsque le "PICBASIC" est relié à un afficheur LCD "série" via son port "PICBUS" d'effacer complètement l'écran.

DELAY Val ou DELAY Var

Cette instruction permet de générer une temporisation dont la durée est fonction de la valeur de "Val". Cette durée exprimée en milliseconde peut s'étendre de 1 à 65535 ms.
DELAY 20 ‘ effectue une pause de 20 millisecondes’

EEREAD (Adr) ou EEREAD (Var)

Cette instruction permet de récupérer une donnée à l'adresse (Adr) dans la mémoire FLASH du "PICBASIC". (4Ko de Flash)
A noter que pour des besoins particulier, il vous est également possible de relire les codes correspondant à votre programme en partie basse de la FLASH.
Exemple :

```
10 DIM DONNEE AS BYTE
20 DONNEE = EEREAD (&HFFF)   ‘lit la valeur mémorisée dans la dernière adresse de la Flash du PB3B’
```

EEWRITE Adr, Val ou EEWRITE Adr, Var

Cette instruction permet d’enregistrer une donnée dans la mémoire FLASH du "PICBASIC". Cette mémoire non volatile (même en cas de coupure d'alimentation) peut être modifiée plus de 10.000 fois. Il vous sera ainsi possible par exemple de sauvegarder des données de configuration qui pourront être relues avec l'instruction "EEREAD". La mémoire FLASH qui sert à la sauvegarde est la même qui sert également à stocker votre programme. Il conviendra bien évidemment à ne pas faire coïncider l'adresse de stockage de votre donnée avec celle de votre programme. Le meilleur moyen est de partir de la "fin" de la mémoire pour stocker vos données en FLASH en "remontant" l'adresse mémoire à chaque nouvelle donnée à stocker. La fin de la mémoire FLASH est fonction du type de "PICBASIC" utilisé .

```
EEWRITE &HFFE, &H55          ‘ Mémorise &h55 dans l’avant dernière adresse de la flash ‘
```

FOR Var = Val1 TO Val2 … NEXT Var

Cette instruction permet de réaliser plusieurs fois de suite certaines actions comprises entre les instructions "FOR" et "NEXT" de votre programme. Le nombre de "répétitions" de ces actions sera déterminé par les valeurs de Val1 et Val2 (Val2 pourra être fixée au maximum à 255 si "Var" a été défini en tant que BYTE et 65279 s'il a été défini en tant que INTEGER).
Exemple :

```
10 DIM I AS BYTE
20 FOR I =0 TO 6
30      BEEP                 ‘ va effectué 7 BEEP’
40 NEXT I
```

FREQOUT Port, Val ou FREQOUT Port, Var

Cette instruction permet de générer un signal carré sur une broche spécifique (Port) du "PICBASIC" de fréquence prédéfinie par (Val). Seules les broches « PWM0 » et « PWM1 » du module "PICBASIC" peuvent être affectées à cette tâche. La valeur de la fréquence de sortie peut être déterminée à l'aide du tableau de correspondance ci-contre. A noter que le signal carré est généré en tâche de fond et que le "PICBASIC" peut réaliser d'autres instructions tout en continuant à délivrer son signal de sortie.

Valeur	Fréq.
1	1.227 KHz
10	1.274 KHz
20	1.324 KHz
40	1.439 KHz
80	1.77 KHz
100	2.00 KHz
191	3.00 KHz
200	5.58 KHz
230	12.0 KHz
253	104 KHz

L'instruction PWMOFF permet de stopper le signal carré. Bien qu'il soit possible de programmer le "PICBASIC" pour générer 2 signaux carrés sur chacune des broches , vous vous apercevrez que la fréquence de chacun des signaux carrés différera de la valeur initialement choisie (malgré la puissance du module, le "PICBASIC" devra effectuer 2 générations de signaux en tâche de fond, ce qui aura pour effet d'influencer ses timings).
Exemple :

```
10 FREQOUT 9,100        ‘ génère un signal carré de 2 kHz sur la I/O 9
20 DELAY 255
30 PWMOFF 9             ‘ et le stoppe au bout de 255 ms
```

GOSUB Ligne ... RETURN

Cette instruction permet depuis un ou plusieurs endroits de votre programme, d'exécuter un "bout" d'un autre programme (encore appelé sous routine) puis de revenir à l'endroit initial pour continuer le déroulement du programme principal. L'instruction "GOSUB Ligne" génère le "saut" à la sous routine, tandis que "RETURN" provoque le retour au programme initial. Comme pour l'instruction "GOTO", il est possible de remplacer un N° de ligne par un "nom" plus explicite à condition que ce dernier soit collé complètement à gauche de l'écran et terminé par ":" (voit description de "GOTO").
A noter qu'il est également possible imbriquer plusieurs sous-routines les unes dans les autres jusqu'à concurrence de 6 (cette limitation étant due à la réservation nécessaire de mémoire RAM de la part du 'PICBASIC' pour mémoriser l'adresse de retour au programme principal).
Exemple:

```
10 CONST BUZZER = 5
20 BEEP BUZZER
30 GOSUB 100
40 BEEP BUZZER
50 GOSUB 100
60 GOTO 20

100 DELAY 255
110 RETURN
```

Ce programme génère un "Bip" sonore sur le buzzer connecté au port "I/O 5" du "PICBASIC", puis va exécuter une temporisation de 255 ms par appel de la sous routine en ligne 100. Arrivée à la ligne 110, l'instruction RETURN indique au "PICBASIC" de reprendre l'exécution "normale" du programme en ligne 40, d'où la génération d'un nouveau "Bip" sonore, puis appel à nouveau de la tempo de 255 ms avec retour en ligne 60 suite à l'exécution de l'instruction RETURN, pour finalement recommencer indéfiniment le cycle en revenant systématiquement en ligne 20.

GOTO Ligne

Cette instruction permet d'exécuter un saut à la ligne indiquée (Ligne).

```
10  DIM I AS BYTE
BOUCLE: I = I + 1
30  GOTO BOUCLE
```

Ce même programme montre qu'il est possible de remplacer un N° de ligne par un "nom" plus explicite. Ce dernier doit commencer impérativement à l'extrémité gauche de l'écran et doit se terminer par ":". Les numéros de ligne ne sont pas obligatoire mais permettent de rendre le programme plus lisible. Ils doivent être complètement à gauche de l’écran et séparés des instruction par un espace.

IF condition ... THEN action ... ELSE action ... ELSEIF condition ... END IF

Ces instructions permettent de réaliser des "actions" en fonction de tests et de conditions définies par vos soins.
tests de conditions les unes dans les autres en simplifiant la rédaction du programme grâce à l'instruction
Exemple:

```
10 DIM I AS BYTE
20 DIM J AS BYTE
30 IF I > 5 THEN J = 0 ELSE J = 1
```

Si (IF) "I" est supérieur à 5 alors (THEN) "J" = 0 sinon (ELSE) "J" = 1. Si la commande à réaliser après (THEN) ne se limite pas à attribué une valeur à une variable, il faudra utiliser la syntaxe de l'autre exemple .

Exemple :

```
10 DIM I AS BYTE
20 DIM J AS BYTE
30 IF I>5 THEN
40 J=ADIN(0)
50 ELSE
60 J=ADIN(1)
70 END IF
```

IN (Port)

Cette instruction permet de connaître le niveau logique "0" (pour 0 V) ou "1" (pour +5 V) présent à l'entrée d'une broche (Port) du "PICBASIC". En fait, de part les seuils des niveaux des "PICBASIC", une tension inférieure à 1,4 V sera considérée comme une valeur "0" tandis qu'une tension supérieure à "3,4 V" sera considérée comme une valeur "1"
Exemple :

```
10 DIM I AS BYTE
20 I = IN(0)              'récupère le niveau  de I/O 0  dans I'
```

KEYDELAY (Instruction, Param1, Param2, Param3)

Cette instruction permet d'introduire un délai lors de l'attente d'une action sur une touche. Le résultat de cette instruction est identique à celui que vous obtiendrez avec les instructions ADKEYIN, KEYIN et PADIN. La seule différence est que le résultat peut être différé (suivant Param2) et répété selon la durée (Param3). En cas d'absence d'action sur la touche, la valeur (Param1) est retournée. "Instruction" peut être remplacé par ADKEYIN, KEYIN ou PADIN.

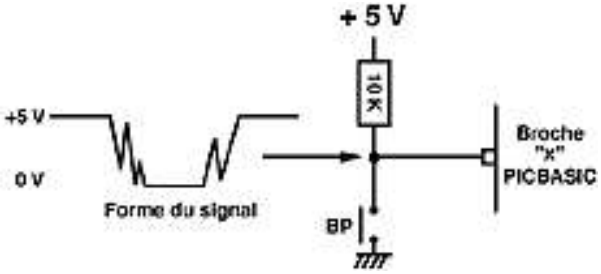
Exemple:

```
10  DIM I AS BYTE
20  I = KEYDELAY (KEYIN(0), 1, 30, 10)
30  IF I = 1 THEN GOTO 10

' Action suite à l'action sur une touche

100 GOTO 10
```

L'instruction de la ligne 20 instaure une temporisation de 30 ms toutes les 10 ms, si aucune touche n'est sollicité, la valeur "1" est restaurée.

KEYIN (Port, Param)

Cette instruction est spécialement conçue pour connaître l'état d'un bouton-poussoir connecté sur un port du "PICBASIC" (Port) en gérant le problème des "rebonds" potentiellement occasionnés lors de l'action sur ce dernier. Le schéma type d'utilisation, donnés ci-contre peut en effet être la source de rebonds pouvant "fausser"' le niveau logique de lecture du programme. Pour y remédier, le paramètre (Param) (en ms) va fixer une durée comprise entre 1 et 100 ms pendant laquelle le "PICBASIC" va attendre un niveau logique stable avant de le valider.



LCDINIT

Cette instruction doit impérativement être exécutée au démarrage du programme si vous désirez piloter un afficheur LCD RS-232 via le port "PICBUS" du "PICBASIC" afin que l'afficheur en question s'initialise correctement.

LOCATE Val1, Val2

Cette instruction permet, si vous pilotez un afficheur LCD à commande RS-232 via le port "PICBUS" du "PICBASIC" de positionner le curseur à un endroit spécifique de l'afficheur ou "Val1" détermine la position horizontale et "Val2", la position verticale .
Il est ainsi possible de piloter des afficheurs dotés au maximum de 4 lignes de 20 caractères selon le même principe.

ON Var GOTO Ligne1, Ligne2, Ligne3 ...

Cette instruction permet de réaliser des accès directs à certaines parties du programme "Ligne1 ou Ligne2 ou Ligne3..." en fonction de la valeur de la variable (Var).

Exemple:
```
10  ON I GOTO 100, 200, 300
20  ' Suite du programme
```
Si I=0, le programme "passe" à la ligne 100, I=1, le programme "passe" à la ligne 200, ...
Si "I" n'est pas égale à une de ces 3 valeurs le programme exécute la ligne 20.

ON INT (Port) = Val GOSUB Ligne

Cette instruction très puissante permet de réaliser un accès direct à un sous-programme à l'adresse (Ligne) suite à l'apparition d'un niveau logique ("0" ou "1") sur une broche du 'PICBASIC' (Port). Cette surveillance est gérée en tâche de fond c'est-à-dire qu'une fois l'instruction exécutée, votre programme principal pourra se dérouler normalement pour être automatiquement interrompu et aller exécuter la sous-routine indiquée par l'adresse (Ligne). A noter qu'après que la première instruction du sous-programme ainsi appelée ai été exécuté, si la broche (Port) présente toujours le même état logique ayant été à l'origine de l'appel de la sous-routine, votre programme effectuera encore la première instruction de l'adresse appelée et ainsi de suite jusqu'à ce que le niveau logique de l'entrée (Pin) change d'état. Cette spécificité pouvant dans certains cas être gênante, il conviendra alors d'utiliser l'instruction **ON INT(16) ... GOSUB** qui réagit non plus sur un niveau, mais sur une transition.

A noter enfin qu'il n'est pas possible de gérer plusieurs interruptions par le biais de plusieurs instructions ON INT attribuées à plusieurs broches du "PICBASIC" (la dernière instruction réalisée annulant toutes les autres).
Exemple:

```
10 ON INT(0) = 0 GOSUB 100
20 OUT 1,0
30 GOTO 20

100 OUT 1,1
110 RETURN
```

Cette exemple illustre bien les possibilités de l'instruction. Une fois ON INT exécuté, si au cours de la boucle 'sans fin' qui consiste à désactiver la sortie "I/O 1" en permanence, un niveau logique bas intervient sur la broche "I/O 0", alors la sortie "I/O 1" sera placée au niveau logique haut gardant cet état tant que le niveau logique sur la broche "I/O 0" ne sera pas "haut".

Lorsqu'on a validé une INT on peut la masquer en déclarant une autre INT sur une broche qui sera reliée matériellement à un niveau la rendant inopérante.

```
'*************** MISE EN ROUTE DE L'INTERRUPTION ***************
ON INT(16)= 0 GOSUB 10
'******************************** PROGRAMME PRINCIPAL ************
out 10,1                          'L'interruption peut être prise en compte car ON INT(16)!!
GOSUB TEMPO
toggle 10
ON INT(4)=0 GOSUB 300             'Si le port 4 est relié au +5V =>l' INT(16) ne peut plus être prise en
                                  'compte dans la suite...
...
```

Si on masque une INT, et qu'un front ou un niveau intervient pendant ce temps, il ne sera évidemment pas pris en compte, MAIS mémorisé (FLAG). Lorsqu'on permettra de nouveau l'interruption, il y aura branchement immédiat au sous-programme à cause du Flag. On peut effacer ce Flag en faisant un GOSUB à la fonction suivante , juste avant de revalider l'INT:

```
MASK_FLAG:
j= peek(&H0b) 'lecture du registre INTCON
j=j and &HFD 'on ne remet que le bit 1 (FLAG de l'INT) à 0, Pas d'action sur les autres bits.
poke &H0b,j 'écriture dans INTCON
return
```

ON INT(16) = Param GOSUB Ligne

Cette instruction très puissante permet de réaliser un accès direct à un sous-programme à l'adresse (Ligne) suite à l'apparition d'une transition logique (passage de "0" à "1" ou inversement) sur la broche (I/O 16) du "PICBASIC". Cette surveillance est gérée en tâche de fond c'est-à-dire qu'une fois l'instruction exécutée, votre programme principal pourra se dérouler normalement pour être automatiquement interrompu et aller exécuter la sous-routine indiquée par l'adresse (Ligne). A noter que si au cours de l'exécution de votre sous programme une nouvelle transition identique à celle du test intervient, le programme repartira une nouvelle fois à partir de la première instruction de la sous-routine. Ceci nous amène à vous inciter à correctement filtrer au moyen d'un condensateur toute entrée bouton-poussoir surveillée par ce type d'instruction afin que les rebonds potentiels générés ne viennent pas interférer dans le bon fonctionnement de votre programme. (int(24) pour le picbasic 3H)

ON TIMER (Param) ... GOSUB Ligne

Cette instruction également très puissante permet de réaliser automatiquement et à intervalle de temps défini par la valeur (Param), un accès direct à un sous-programme à l'adresse (Ligne) - cette gestion étant traitée en tâche de fond. Il vous sera ainsi possible par exemple de "rafraîchir" automatiquement l'affichage d'un écran toutes les 2 secondes, ou émettre un "bip" sonore toutes les secondes, etc... L'intervalle de temps défini par la valeur (Param) se fait grâce à une table de correspondance (la table ne donne que les 10 premières valeurs, les autres peuvent être calculées aisément, sachant que la valeur (Param) ne peut excéder 60).
Il est impératif que la durée d'exécution du programme de la sous-routine appelée soit plus courte que la durée d'appel générée par l'instruction ON TIMER, sans quoi le programme de la sous-routine n'aura sans cesse d'être appelé en continu.
ON TIMER(5) GOSUB 100 'toutes les secondes , « passez » à la ligne 100'

Param	Durée
0	0,105 sec
1	0,210 sec
2	0,420 sec
3	0,630 sec
4	0,840 sec
5	1,05 sec
6	1,26 sec
7	1,47 sec
8	1,68 sec
9	1,89 sec
10	2,10 sec

LE timer qui est utilisé est le TIMER1 du PIC16F876 , le timer 0 étant utilisé pour la fonction delay( ).

Pour arrêter l'action de ON TIMER( ) dans une partie de programme, placer l'instruction :
poke &H10,&h30 'dévalide le bit d'enable INT

pour l'autoriser de nouveau, placer :
poke &H10,&h31 'valide le bit d'INT TIMER
La prochaine instruction ON TIMER ( ) fonctionnera.

Exemple :

```
200 ON TIMER(5) GOSUB 10
for i=0 to 4                'cette partie est interrompue par ON TIMER(5)
out 12,1 : out 10,1
Gosub tempo
toggle 12
gosub tempo
next i                      'cette partie est toujours interrompue par ON TIMER(5)
poke &H10,&h30              'dévalide le bit d'enable INT
for i=0 to 30
delay 100
next i
poke &H10,&h31              'valide le bit d'INT TIMER => ON TIMER(5) se relance
...
```

OUT Port, Val

Cette instruction permet de faire passer une broche "I/O" (Port) dès lors configurée en sortie à un niveau logique donné (Val). La "sortance" des broches du 'PICBASIC' permet de tirer jusqu'à 25 mA au maximum (sur un état haut ou bas). Ceci est largement suffisant pour allumer une.

OUT 0,1 'Place le I/O 0 à l'état haut'
OUT A,B 'Place le I/O A à l'état B(0 ou 1)'

OUTSTAT (Port)

Cette instruction permet de connaître l'état logique d'une broche "I/O" (Port) (que celle-ci soit utilisée en tant qu'entrée ou que sortie !). Ceci peut être très utile, si on utilise une broche du "PICBASIC" en "sortie" sans avoir son "reflet" dans une variable mémoire que l'on peut consulter.

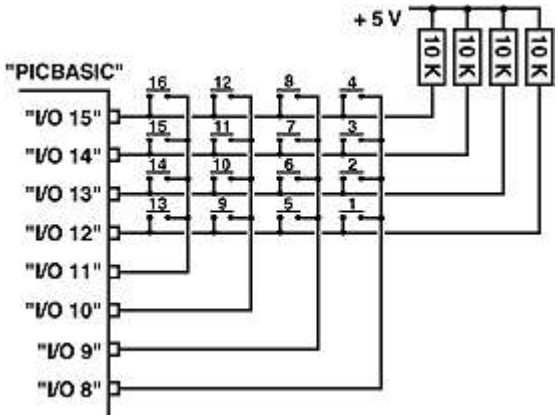
Exemple:

```
10 DIM I AS BYTE
20 OUT 0,1
30 I = OUTSTAT(0)
```

Dans cet exemple, la variable "I" aura comme valeur "1".

PADIN (1)

Cette instruction permet de gérer automatiquement des claviers 16 touches de type matriciel. Ce dernier devra être connecté comme indiqué ci-contre sur les broches "I/O 8" à "I/O 15" des modules "PICBASIC" (les colonnes entre "I/O 8 à I/O 11" et les lignes entre "I/O 12" à "I/O 15"). Dès lors, en effectuant l'instruction PADIN(1), le module effectuera automatiquement un 'scanning' des 16 touches et vous retournera une valeur spécifique à la touche sollicitée. Si plusieurs touches sont sollicitées en même temps, seule la touche "renvoyant" le N° le plus petit sera prioritaire. Si aucune touche n'est sollicitée, la valeur "0" est retournée.



Exemple :

```
10 DIM I AS BYTE
20 I = PADIN(1)
```

PEEK (Adr)

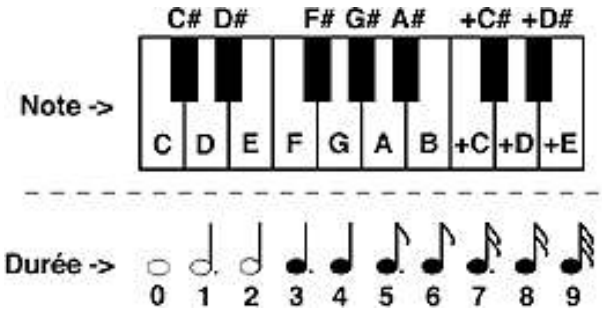
Cette instruction permet de lire directement dans les registres internes du microcontrôleur PIC qui assure la gestion du PICBASIC (PIC16C876 ou PIC16C877 suivant les modèles). La valeur de l'octet se trouvant à l'adresse (Adr) peut ainsi être connu. Nous recommandons aux personnes qui ne sont pas familiarisées avec les microcontrôleurs PIC de **ne pas utiliser cette instruction**.

POKE Adr, Val

Cette instruction permet "d'écrire" la valeur (Val) directement à l'adresse mémoire (Adr) du microcontrôleur PIC qui assure la gestion du PICBASIC (PIC16C876 ou PIC16C877). Nous recommandons aux personnes qui ne sont pas familiarisées avec les microcontrôleurs PIC de **ne pas utiliser cette instruction**.

PLAY Port, "Val1Val2Val3 ..."

Cette instruction permet la génération de notes musicales sur une broche du module PICBASIC" (qui devra être connectée à un buzzer sans oscillateur, voir instruction "BEEP"). Le paramètre 'Port' indique la broche où sera connecté le buzzer. Les paramètres 'Val1Val2Val3...' indiquent la valeur des notes conformément au "clavier" ci-contre (la lettre indique la note et le chiffre, la durée de la note). Les notes doivent être écrites en majuscules. Ne pas mettre plus de 4 notes par play.



Exemple :
10 PLAY 5, "C4D4E4F4" 'joue « do ré mi fa sol »

PRINT "Texte"

Cette instruction permet d'afficher des messages sur un afficheur LCD spécifique (à commande RS-232) préalablement connecté sur le port "PICBUS" du "PICBASIC". Avant d'utiliser cette commande, il faudra bien évidemment initialiser l'afficheur ("LCDINIT") et positionner le curseur à l'endroit où le texte devra commencer ("LOCATE").

Exemple1:
10 LCDINIT
20 LOCATE 0,0
30 PRINT "LEXTRONIC-2001"

Exemple3:
10 LCDINIT
20 LOCATE 0,0
30 PRINT "LEXTRONIC-200",&H31

CONST BYTE VALEUR = (9, " LUNDI ", " g ") ' tableau
PRINT(VALEUR (1) , VALEUR (2), VALEUR (3), VALEUR (6)) 'affiche : LUNg
On obtient la même chose avec valeur(a) si on a mis avant a=1

PRINT DEC (Var, Val1, Val2)

Cette instruction permet d'afficher la valeur décimale d'un nombre selon plusieurs formats possibles sur un afficheur LCD spécifique (à commande RS-232) préalablement connecté sur le port "PICBUS" du "PICBASIC". Avant d'utiliser cette commande, il faudra bien évidemment initialiser l'afficheur ("LCDINIT") et positionner le curseur à l'endroit ou le texte devra commencer ("LOCATE"). Le nombre à afficher (Var) peut être une valeur fixe ou une variable, le paramètre (Val1) détermine le nombre de caractères que devra occuper le nombre à l'écran (1 à 5) - Si (Val1) est inférieur au nombre de chiffres qui compose le nombre à afficher, ce dernier sera alors affiché en partie. Le paramètre (Val2) permet de déterminer si le nombre doit être précédé de '0' à la place des caractères non utilisés (Val2 = 0 -> affichage de '0', Val2 = 1 -> Affichage d'espace). En absence de paramètre (Val1 et Val2), le "PICBASIC" affichera automatiquement le nombre sur 5 caractères, sans '0' devant.

Exemple :
```
10  DIM ANNEE AS INTEGER
20  ANNEE = 2001
30  LCDINIT
40  LOCATE 0,0
50  PRINT "LEXTRONIC-",DEC(ANNEE,4,1)
```

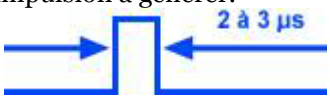
PRINT HEX (Var, Val1, Val2)

Cette instruction qui s'utilise exactement de la même manière que l'instruction "PRINT DEC" permet d'afficher la valeur hexadécimale d'un nombre selon plusieurs formats possibles sur un afficheur LCD spécifique (à commande RS-232) préalablement connecté sur le port "PICBUS" du "PICBASIC".

PULSE Val

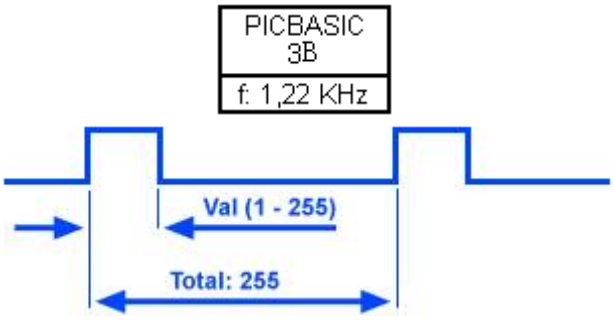
Cette instruction permet de générer des impulsions (positives ou négatives) de durée (Val) sur les broches "I/O" du "PICBASIC". Elle doit être préalablement précédée d'une initialisation au niveau "haut" ou "bas" de la broche en question afin de déterminer le type d'impulsion à générer.

10 OUT 2,0
20 PULSE 3



PWM Port, Val

Cette instruction permet de générer sur une des broches du "PICBASIC" (Port), un signal carré de fréquence fixe mais dont le rapport cyclique est variable et fonction de (Val). Ce type de signal encore appelé "PWM" (impulsions à durée variable) est généralement utilisé pour le pilotage de moteur à courant continu (via une interface de puissance) ou pour la génération de tensions analogiques. Seules les broches "PWM0" (I/O 9) et "PWM1" (I/O10) des modules "PICBASIC" peuvent être affectées à cette tâche (donc Port = 9 ou Port = 10). A noter que ce signal est généré en tâche de fond et que le "PICBASIC" peut réaliser d'autres instructions en même temps. L'instruction PWMOFF permet de stopper le signal (voir ci-après). Il est également possible de générer indépendamment 2 signaux "PWM" de valeurs différentes sur les broches (I/O 9) et (I/O 10) d'un même "PICBASIC". En revanche, il ne sera pas possible de générer un signal "PWM" sur une sortie et un signal carré avec l'instruction "FREQOUT" sur une autre sortie en même temps.



10 PWM 10,128 sur le port I/O 10 ,apparaît un signal carré de rapport cyclique égal à 0,5.

PWMOFF Port

Cette instruction permet de désactiver le signal "PWM" (initialement généré par l'instruction "PWM") ou un signal carré (initialement généré par l'instruction "FREQOUT") sur la broche (Port) du "PICBASIC".

RND (0)

Cette instruction permet de générer une suite de chiffres "pseudo" aléatoires.

Exemple :

```
10 DIM I AS INTEGER
15 LCDINIT
20 I = RND(0)
30 LOCATE 0,0
PRINT DEC( I)
DELAY 300
GOTO 20
```

Ce programme va afficher sur l'écran LCD une suite ne nombre aléatoire en INTEGER. Il y a possibilité de travailler sur une variable BYTE

SERIN Port, Param1, Mode, Param2, Adr, [Var1]

Cette instruction permet d'attendre la réception de données sous forme série selon un protocole compatible "RS-232". Une fois exécutée, la broche (Port) du "PICBASIC" attendra la ou les données (Var1) à une vitesse définie par (Param1), selon la correspondance du tableau ci-contre.
Durant cette phase, le module ne pourra pas effectuer d'autres instructions et attendra en permanence les données pendant une durée (en ms) définie par (Param2). Si la durée d'attente est dépassée, sans qu'aucune donnée ne soit reçue, le programme "sautera" alors directement à l'adresse "Adr". Le paramètre (Mode) n'est pas utilisé et doit être mis à "0".

Vitesse Bauds	PICBASIC 3B
4800	207
9600	103
19200	50
38400	23
57600	13
115200	5

Exemple 1 :
10 SERIN 2, 207, 0, 5000, TIMEOUT, [I]

Attend pendant 5 secondes une donnée série à 4800 bds sur la broche "I/O 2". Si cette donnée arrive à temps, elle sera stockée dans la variable "I". En cas contraire, le programme continu à l'adresse "TIMEOUT".

Exemple 2 :
10 DIM I(10) AS BYTE
20 SERIN 2, 207, 0, 5000, TIMEOUT, [I(0)~ 5]

Dans cet exemple le "PICBASIC" va attendre 5 données et transférer celles-ci dans: I(0), I(1), I(2), I(3), I(4). Il est également possible d'instaurer des conditions lors de la réception des données à l'aide d'instructions complémentaires.

Exemple 3 :
10 SERIN 2, 207, 0, 5000, TIMEOUT, [WAIT("AB"),I]

Attend pendant 5 secondes la suite de caractères "AB" sous forme de données série à 4800 bds sur la broche "I/O 2". Si cette "trame" caractéristique n'arrive pas à temps, le programme continu à l'adresse "TIMEOUT". Si cette "trame" caractéristique est reconnue avant 5 secondes, le "PICBASIC" enregistre le caractère suivant dans la variable "I".

Exemple 4 :
10 DIM I(30) AS BYTE
20 SERIN 2, 207, 0, 5000, TIMEOUT, [UNTIL("$"),I(0)~ 20]

Attend pendant 5 secondes une suite de données série à 4800 bds sur la broche "I/O 2". Si ces données n'arrivent pas à temps, le programme continu à l'adresse "TIMEOUT". Si ces données arrivent, les 20 premières seront stockées dans I(0), I(1), I(3)... Si parmi les 20 données, le "PICBASIC" détecte le caractère "$", la réception est interrompue.

Exemple 5 :
10 DIM I(30) AS BYTE
20 SERIN 2, 207, 0, 5000, TIMEOUT, [SKIP("E"),I(0)~ 20]

Attend pendant 5 secondes une suite de données série à 4800 bds sur la broche "I/O 2". Si ces données n'arrivent pas à temps, le programme continu à l'adresse "TIMEOUT". Si ces données arrivent, les 20 premières seront stockées dans I(0), I(1), I(3)... Lors de la réception, tous les caractères "E" seront "sautés" et ne feront pas l'objet d'une mémorisation.

Si vous devez interfacer les modules "PICBASIC" avec d'autres microcontrôleurs, sachez que le signal "RS-232" généré est de type 8 bits avec 1 bit de stop, sans bit de parité. Dans ces conditions, si vous appliquez un niveau bas permanent sur l'entrée de réception série du "PICBASIC", ce dernier considérera qu'il est en train de recevoir une trame de données série. Le signal d'entrée est bien sûr compatible TTL. Ainsi, dans le cadre d'un interfaçage avec un port "RS-232" d'un compatible "PC", il vous faudra impérativement utiliser un circuit intégré type "MAX-232" afin de rendre compatible les signaux du "PICBASIC" avec les niveaux ±10 V du port de l'ordinateur .

SEROUT Port, Param1, Mode, Param2, [Var1]

Cette instruction permet de transmettre des données sous forme série selon un protocole compatible "RS-232". Une fois exécutée, la broche (Port) du "PICBASIC" transmettra la ou les données (Var1) à une vitesse définie par (Param1), selon la correspondance du tableau ci-contre. Le paramètre (Mode) permet d'instaurer une temporisation entre chaque caractère émis dont la durée en "ms" est fonction de (Param2).
Les données à envoyer doivent être de type "BYTE". Si vous essayez d'envoyer des données de type "INTEGER", seuls les 8 bits de poids faible seront transmis.

Vitesse Bauds	PICBASIC 3B
4800	207
9600	103
19200	50
38400	23
57600	13
115200	5

Exemple 1 :
```
10 SEROUT 3, 207, 0, 1, [&HA0]

' La broche "I/O 3" transmet à 4800 bds la donnée avec un délai de 1mS.
```

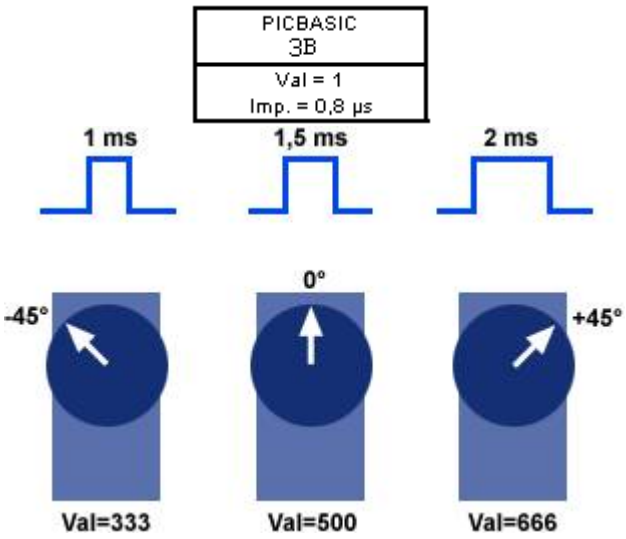
Exemple 2 (sur "PICBASIC-1B"):
```
10 SEROUT 1, 50, 0, 0, ["PICBASIC", 13, 10]

' La broche "I/O 1" transmet à 19200 bds les caractères de la phrase "PICBASIC", puis les données "13" et "10", sans aucun délai entre les caractères.
```

Exemple 3 :
```
10 SEROUT 1, 207, 0, 0, [DEC(I)]
20 SEROUT 1, 207, 0, 0, [HEX(I)]

' Exemples de transmission avec conversion simultanée.
```

Exemple 4 :
```
10 DIM I(30) AS BYTE
I(0) = 28
I(1) = 58
I(2) = 4
20 SEROUT 1, 207, 0, 0, [I(0) ~ 20]

' Transmet les 20 premières valeurs contenues dans I(0), I(1), I(2), I(3)...
```

SERVO Port, Val

Cette instruction permet de faire varier la position du palonnier d'un servomoteur de 0 à 90° en fonction de la durée des impulsions qu'on lui applique via la sortie (Port). La durée des impulsions sera proportionnelle à (Val). Ainsi pour Val=1 -> l'impulsion = 0,8 µ s .
Donc pour la position (0°) -> Val = 1,5 ms / 0,0008 ms = 1875. A noter que les impulsions générées devront être espacées de 10 à 15 ms chacune.



Enfin, de part le manque de précision des servomoteurs, les valeurs indiquées peuvent différer d'un modèle à l'autre. Dans tous les cas, on veillera à ne pas alimenter le servomoteur sur la même source que le module "PICBASIC" qui pourra être potentiellement gêné par les parasites importants générés lors de la rotation du moteur.
(il est impératif que la génération des impulsions soit sans discontinuité, assez difficile à réaliser)

SET PICBUS Param

Cette instruction permet de paramétrer la vitesse de communication du "bus" spécialisé "PICBUS" afin de l'adapter en fonction du type d'afficheur à liaison série à commander.

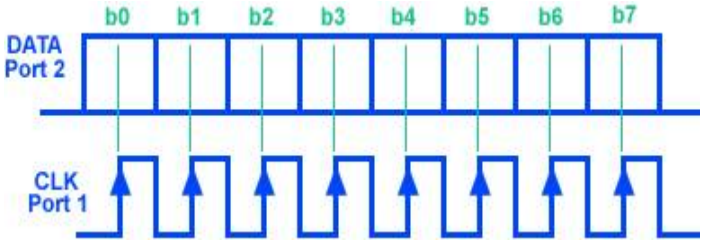
```
Exemple:    SET PICBUS HIGH     'Configure le "BUS" à 19200 bps
            SET PICBUS LOW      'Configure le "BUS" à 4800 bps
```

WARNING :
Les 2 instructions SHIFTIN et SHIFTOUT sont réservées au maître car elles génèrent l'horloge. Elles ne permettent donc que de communiquer avec des circuits spécialisés I2C et non entre PICs.

SHIFTIN (Port1,Port2,Mode,Bit)

Cette instruction permet de "communiquer" très facilement avec la plupart des composants à adressage série 2 fils (type I2C™, SPI™...). Son exécution génère un signal d'horloge de synchronisation sur la sortie (Port1) du "PICBASIC", tout en venant "lire" sériellement les données présentent sur l'entrée (Port2). Le paramètre (Mode) permet de définir le mode de lecture (voir syntaxe ci-après). Le paramètre (Bit) permet de définir le nombre de bits à lire (8 ou 16).

Exemple:
SHIFTIN(3,4,0)



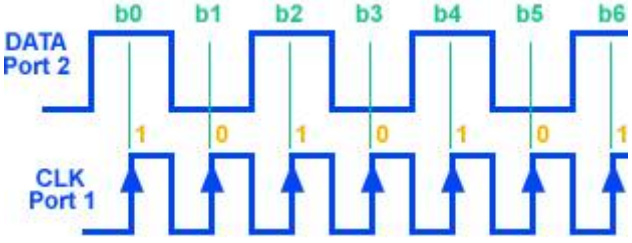
' Mode:
' 0 = LSB prioritaire, lecture après le front montant d'horloge.
' 1 = MSB prioritaire, lecture après le front montant d'horloge.
' 2 = LSB prioritaire, lecture après le front descendant d'horloge.
' 3 = MSB prioritaire, lecture après le front descendant d'horloge.
' BIT: 8 ~ 16 bits (par défaut 8 bits).

Exemple :
SEC=SHIFTIN(1,2,2,8) 'horloge sur le port 1, donnée sur le port 2, LSB, 1 octet de donnée

SHIFTOUT Port1,Port2,Mode,Val,Bit

Cette instruction permet de "communiquer" très facilement avec la plupart des composants à adressage série 2 fils (type I2C™, SPI™...). Son exécution génère un signal d'horloge de synchronisation sur la sortie (Port1) du "PICBASIC", tout en venant "écrire" en série les données sur la sortie (Port2). Le paramètre (Mode) permet de définir le mode d'écriture (voir syntaxe ci-après). Le paramètre (Bit) permet de définir le nombre de bits à lire (8 ou 16).

Exemple : SHIFTOUT 0,1,0,&H55,8



' Mode :
' 0 = LSB prioritaire
' 1 = MSB prioritaire
' 2 = MSB prioritaire avec génération d'un signal « ACK » (convient pour le pilotage de composant I2C™).

' BIT: 8 ~ 16 bits (par défaut 8 bits).

Exemple :
SHIFTOUT 1,2,0,&H81,8 'horloge sur le port 1 , donnée sur le port 2 , LSB, envoie &H81 sur 1 octet

SOUND Port, Val1, Val2...

Cette instruction permet de générer un signal sonore de tonalité proportionnelle à la valeur (Val1) et de durée proportionnelle à la valeur (Val2) sur la broche (Port) du "PICBASIC". Il est possible de cumuler plusieurs tonalités les unes à la suite des autres.

```
10 SOUND 1,239,10,159,10  ' sirène pompier sur le port 11
20 GOTO 10
```

STEPOUT Port1, Var1, Var2, Port2, Var3

Cette instruction permet en association avec des interfaces spécialisées, de piloter très facilement des moteurs pas-à-pas. Ces interfaces doivent pouvoir faire tourner le moteur suivant un nombre de "pas" proportionnel au nombre d'impulsions qui leur sont envoyées. Ainsi "STEPOUT" permet de générer des impulsions sur la sortie (Port) dont la fréquence est proportionnelle à (Var1), suivant le tableau ci-contre et dont le nombre est directement fonction de (Var2).

Valeur de Var1	PICBASIC 3B
1	f: 60,9 KHz
2	f: 53,2 KHz
5	f: 38,47KHz
10	f: 26,31 KHz
20	f: 16,13 KHz
50	f: 7,463KHz
100	f: 3,937 KHz
200	f: 2,024 KHz
255	f: 1,597 KHz

Exemple 1: 10 STEPOUT 2, 50, 10
Cette ligne va générer 10 impulsions de fréquence 7,46 KHz .

Il est également possible ajouter une condition (sur une entrée du "PICBASIC") permettant de stopper la génération des impulsions (idéal si vous disposez par exemple de contacts de fin de course). Ainsi lors de la génération des impulsions, si le niveau logique présent sur (**Port2**) est égal à (**Var3** -> 0 ou 1), alors les impulsions cesseront. SI avant la fin du nombre d'impulsions programmé, le niveau logique reprend un état autorisé, les impulsions reprendront.

Exemple 2:
10 STEPOUT 2, 50, 10, 3, 0
Génère 10 impulsions de fréquence 7,46 KHz et stoppe ces dernières si le niveau logique présent sur "I/O 3" tombe à "0".

TABLE (Var, Val1, Val2, Val3...)

Cette instruction très utile permet d'attribuer une valeur particulière (Val1 ou Val2 ou Val3...) à une variable en fonction de la valeur d'une autre variable (Var).

Exemple:
```
10  DIM I AS BYTE
20  DIM J AS BYTE
30  I = 2
40  J = TABLE (I,192,45,35,68,99)
```

Dans cet exemple, la variable "J"=35. Si "I" avait initialisé avec la valeur "0" alors "J" aurait été égal à 192. Si "I" avait initialisé avec la valeur "1" alors "J" aurait été égal à 45, etc...

TOGGLE Port

Cette instruction permet d'effectuer un changement de d'état logique d'une broche (Port). Si la broche était au niveau logique "0" (0 V), celle-ci passera au niveau logique "1" (+5 V) et inversement.

IV Note d'application des afficheurs LCD

a) Utilisation avec un PICBASIC

On utilisera le connecteur 3 fils (RX / +5 V / GND) que l'on raccordera comme indiqué dans la notice des PICBASIC en vérifiant bien que la tension d'alimentation n'excède pas +5 V sous peine de destruction du module. On se référera ensuite à la notice des "PICBASIC" pour "piloter" ces derniers grâce aux instructions:
SET PICBUS: initialisation de la vitesse de communication avec l'afficheur.
LCDINIT: initialisation de la communication avec l'afficheur.
CLS: Efface l'afficheur.
LOCATE: Positionne le curseur à un endroit précis sur l'afficheur:
PRINT: Afficheur un message, des nombres, etc...

Protocole "RS-232" reconnu":
Le protocole du signal "RS-232" reconnu par les afficheurs "séries" est du type: 1 bit de start, 8 bits de données, sans parité, 1 bit de stop. Le débit est de base configuré à 9600 bds (il est toutefois possible de passer à 4800 bds en coupant le piste au dos du circuit marqué "JP1").

b) Commandes reconnues par l'afficheur

Le tableau ci-dessous indique l'ensemble des commandes reconnues par les afficheurs:

AO	(Initialisation "soft" de l'afficheur - commande facultative).
A3 01	Efface le contenu de l'écran et positionne le curseur à la position "0,0" (en haut à gauche).
A1 X Y	Place le curseur à la position définie par "X" (0 à 19 suivant le nombre de caractères) et "Y" (0 à 4 suivant le nombre de lignes). Si X=0 et Y=0, le curseur se retrouvera tout à fait en haut à gauche de l'écran.
A2 C1, C2, C3.... 0	Cette suite d'octets permet d'afficher à l'écran les caractères des codes ASCII correspondant aux octets C1, C2, C3, etc... à partir de la position du curseur. Le dernier caractère à afficher doit automatiquement être suivi de l'octet "0" afin que l'afficheur puisse quitter la phase d'affichage et attendre à nouveau une nouvelle commande.
A3 0C	Cette suite de 2 octets permet de désactiver la présence du curseur sur l'afficheur.
A3 0E	Cette suite de 2 octets permet d'activer le curseur sur l'afficheur (mode défaut).
A4 CARACT,C1,C2,C3,C4,C5,C6,C7,C8	Cette suite de 10 octets permet de redéfinir l'apparence de certains caractères et de les afficher à l'écran. Il vous sera ainsi possible de "redessiner" totalement les caractères correspondants aux emplacements des codes "ASCI"I 8 à 15 de l'afficheur. Chaque caractère est représenté sur une matrice de points composée de 8 lignes et de 5 colonnes. Lors de la redéfinition, vous devrez indiquer quels seront les points de chacune des 8 lignes qui devront être allumées. Cette commande s'exécute donc en commençant par l'octet **A4**, suivi du code "ASCII" du caractère à redéfinir (8 à 15) et de la description des 8 lignes de caractère. Après son exécution, le caractère s'affiche directement.
A5 CARACT,C1,C2,C3,C4,C5,C6,C7,C8	Cette instruction s'utilise exactement comme la ligne ci-dessus, mis à part que le caractère bien que modifié, ne s'affiche pas à l'écran. Pour le visualiser, il vous faudra avoir recours au code "A2", suivi du code "ASCII" du caractère modifié, suivi de l'octet "0" ('voir indication du tableau du code A2).

Exemple de redéfinition de caractères avec un "PICBASIC" :

Il est également possible de piloter les afficheurs LCD à l'aide des PICBASIC en utilisant les mêmes codes que ceux expliqués dans les tableaux ci-dessus (pour ce faire, il faudra utiliser la commande BUSOUT qui envoie des données séries directement sur la sortie PICBUS des PICBASIC).

```
SET PICBUS HIGH ' Cette instruction doit être placer impérativement avant "LCDINIT"
LCDINIT
BUSOUT &HA3,&H01 ' Efface l'écran

BUSOUT &HA3,&H0C ' Désactive l'affichage du curseur

BUSOUT &HA5,8,0,0,0,15,15,0,0,0
' Redéfini (sans l'afficher le caractère présent à l'emplacement du code "ASCII N° 8)

BUSOUT &HA5,9,0,&H10,&H18,&H1C,&H1C,&H18,&H10,0
' Redéfini (sans l'afficher le caractère présent à l'emplacement du code "ASCII N° 9)

BUSOUT &HA1, &H00,&H01 ' Se place sur la 2ème ligne de l'afficheur

BUSOUT &HA2, &H08,&H09,&H00
 ' Affiche les 2 caractères correspondants aux codes ASCII N° 8 et N° 9
```

V NOTE D'APPLICATION DES PLATINES "SGN"

1) Interfaçage des modules:
L'interfaçage de ces modules avec un "PICBASIC" est on ne peut plus simple. Reliez la borne +5V au + 5 V d'alimentation et la borne GND à la masse du "PICBASIC" (pensez à vérifier la valeur de la consommation globale de tous les afficheurs qui peuvent "monter" au delà de ce que peut supporter le régulateur +5V - Utilisez également un "radiateur approprié sur le régulateur qui risque de chauffer). Reliez enfin la broche "RX" à une des broches du "PICBASIC" configurée en sortie série RS-232. Dans le cadre d'une commande via un ordinateur, il conviendra d'utiliser IMPERATIVEMENT un circuit d'interfaçage du type MAX232 - sous peine de destruction du module, non prise en compte par la garantie) afin de mettre à niveau les signaux électriques entre les 2 systèmes. Le signal de commande série émis par l'ordinateur devra être au format RS-232 envoyé à 9600 bds, 8 bits, avec 1 bit de start, 1 bit de stop et sans bit de parité.

2) Format des données à envoyer:
Les différentes informations à envoyer pour piloter ces afficheurs sont:

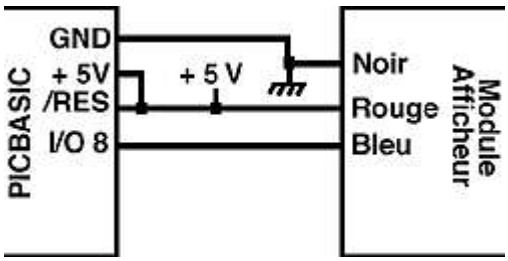
> L'adresse de base de l'afficheur:
Chaque afficheur dispose d'un mini-commutateur D.I.L 4 positions vous permettant de configurer une adresse "matériel" sur ces derniers afin de pouvoir piloter indépendamment jusqu'à 16 afficheurs différents (a ce titre, un connecteur spécial est disponible sous l'afficheur afin de pouvoir facilement chaîner ces derniers en "reportant" les signaux +5 V, GND et RX d'un module à l'autre). Il vous faudra donc impérativement lors de chaque commande envoyer en priorité l'adresse exacte de l'afficheur que vous désirez piloter. Lorsque les 4 dils sont en position "OFF", l'adresse correspondante est "&HE0". Si seul le DIL 1 est en position "haute", l'adresse correspondante est "&HE1", le DIL 2 seul activé correspond à l'adresse "&HE2", les DILS 1 et 2 seuls en position haute à l'adresse "&HE3" et ainsi de suite...

> Le type de l'afficheur:
Il existe (indépendamment de la taille des afficheurs) 3 types de modèles en fonction du nombre de digits présent sur le circuit imprimé. Lors de la première utilisation il vous faudra impérativement "déclarer" à l'afficheur le type de modèle utilisé. Le code "&HA3" correspond à un modèle 3 digits, "&HA4" à 4 digits et "&HA5" à 5 digits.

> Le type de données affichable:
Il est possible d'afficher soit:

- La valeur d'une variable (au format hexadécimal ou BCD).
- Directement un caractère parmi la table de correspondance donnée ci-après.
- Soit un ou plusieurs points de séparation (clignotant ou non).
- Soit d'envoyer une commande d'activation/désactivation de clignotement d'un ou plusieurs digits.

Les exemples qui suivent vous permettront de maîtriser toutes les possibilités d'adressage. Dans tous les cas, les exemples donnés peuvent être réalisés avec des modules "PICBASIC-1B/1S/2S/2H" dont la broche P8 est reliée à l'entrée RX du module afficheur. Dans le cadre d'une utilisation avec un module de type "PICBASIC2000", il conviendra simplement de réajuster la valeur de correspondance du débit de 9600 bauds (et de ne pas oublier que la sélection des octets de poids fort/faible ne se fait pas avec le caractère "." mais avec le caractère ":".



3) Affichage d'une variable:
La demande d'affichage d'une variable s'effectue simplement en envoyant:

> La donnée "&HFA" (si vous désirez afficher la valeur hexadécimale).
> Le donnée "&HFB" (si vous désirez afficher la valeur BCD).

Dans tous les cas, cette donnée devra être envoyée après l'adresse de base l'afficheur. L'envoi de la variable de type "INTEGER" devra se faire d'abord par l'octet de fort ("Variable.H" sur les PICBASIC ou "Variable:H" sur les PICBASIC2000), puis l'octet de poids faible ("Variable.L" sur les PICBASIC ou "Variable:L" sur les PICBASIC2000). L'exemple ci-dessous permet d'afficher un compteur sur l'afficheur (en remplaçant "&HFB" par "&HFA" vous obtiendez un affichage de la valeur hexadécimale.

```
10  DIM I AS INTEGER
20  SEROUT 8,30,0,0,[&HE0,&HA5] ' Déclaration d'un module 5 digits (&HA5) à la 1ère adresse (&HE0)
30  I = I + 1
40  SEROUT 8,30,0,0,[&HE0,&HFB,I.H,I.L] ' Affichage de la valeur BCD (&HFB) de "I"
50  GOTO 30
```

4) Affichage d'un caractère:

Octet de poids faible

Octets de poids fort	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2		]	_	=	≡		⌜	⌞	⌝	⌟	U	n	/			
3	0	1	2	3	4	5	6	7	8	9	.	-				
4		A	b	C	D	E	F	g	H	I	J		L		n	O
5	P	q	r	S	t	U				y						
6		A	b	c	d	e	F	g	h	i	j		I		n	o
7	P	q	r	s	t	u				y						

L'affichage d'un caractère s'effectue très simplement en envoyant (après l'adresse de l'afficheur) un chiffre (de 1 à 5) correspondant à la position du digit, suivit du caractère à afficher à cette position. Le caractère peut être choisi en "clair" ou selon son code ASCII, conformément au tableau ci-dessus. En raison de la nature de l'afficheur, certaines lettre dispose d'une représentation simplifiée.

EXEMPLES :

```
10  SEROUT 8,30,0,0,[&HE0,&HA5]    ' Déclaration d'un module 5 digits (&HA5) à la 1ère adresse (&HA0)
20  SEROUT 8,30,0,0,[&HE0,1,"P"]    ' Affichage du caractère "P" à la 1ère place
30  SEROUT 8,30,0,0,[&HE0,2,"L"]     ' Affichage du caractère "L" à la 2ème place
40  SEROUT 8,30,0,0,[&HE0,3,"A"]    ' Affichage du caractère A" à la 3ème place
50  SEROUT 8,30,0,0,[&HE0,4,"Y"]    ' Affichage du caractère "Y" à la 4ème place
60  SEROUT 8,30,0,0,[&HE0,5,&H23] ' Affichage du caractère "=" à la 5ème place
```

5) Affichage / clignotement d'un "point" de séparation:
Il est possible d'afficher ou de faire clignoter un ou plusieurs "points" de séparation entre les "Digits". Pour ce faire, il suffit simplement d'envoyer (après l'adresse de l'afficheur) une commande relative au "point" à afficher ou à faire clignoter.

&HD1 (Allumage "point" N° 1) ou &HD6 (clignotement "point" N° 1)
&HD2 (Allumage "point" N° 2) ou &HD7 (clignotement "point" N° 2)
&HD3 (Allumage "point" N° 3) ou &HD8 (clignotement "point" N° 3)
&HD4 (Allumage "point" N° 4) ou &HD9 (clignotement "point" N° 4)
&HD5 (Allumage "point" N° 5) ou &HDA (clignotement "point" N° 5)
&HD0 (Extinction de tous les "points") ou &HDF (Arrêt clignotement de tous les points)

Exemple de programme pour allumer le "point" N° 4:

```
10 SEROUT 8,30,0,0,[&HE0,&HA5]
20 SEROUT 8,30,0,0,[&HE0,&HD4]
```

6) Clignotement d'un digit:
Il est possible de faire clignoter un ou plusieurs "digits". Pour ce faire, il suffit simplement d'envoyer (après l'adresse de l'afficheur) une commande relative au "digit" à faire clignoter.

&HF1 (clignotement du "Digit" N° 1)
&HF2 (clignotement du "Digit" N° 2)
&HF3 (clignotement du "Digit" N° 3)
&HF4 (clignotement du "Digit" N° 4)
&HF5 (clignotement du "Digit" N° 5)
&HF0 (clignotements de tous les digits)
&HFF (Arrêt de tous les clignotements)