Les instructions des PICBASIC 3B/3H

CLASSEMENT DES INSTRUCTIONS PAR FONCTION

Les carrés sont des liens

1. Déclarations (structures de données)

DIM	déclare une variable ou un tableau
CONST	déclare une constante
CONST BYTE	déclare un tableau de constantes de type octet
CONST INTEGER	déclare un tableau de constantes de type entier
TABLE	crée un tableau d'octets repérables par un index

2. Entrées / Sorties

IN	lecture de l'état logique d'une broche et affectation de la valeur à une variable
OUT	place une broche à un état logique
OUTSTAT	lit l'état, 1 ou 0, d'une sortie
TOGGLE	inverse l'état logique présent sur une broche
PULSE	crée une impulsion de durée déterminée
BYTEIN	lecture de 8 ports consécutifs, simultanément
BYTEOUT	écriture d'un octet sur 8 ports consécutifs, simultanément

3. Gestion "LCD"

LCDINIT	lance les instructions nécessaires à l'initialisation d'un afficheur
CLS	efface l'écran
LOCATE	positionne le curseur à un emplacement choisi
PRINT	affiche un texte
PRINT DEC	affiche une valeur décimale
PRINT HEX	affiche une valeur en hexadécimal
SET PICBUS	fixe la vitesse de transmission
CSRON	fait apparaître le curseur
CSROFF	fait disparaître le curseur
BUSOUT	envoie un octet de configuration à l'afficheur

4. Structures de traitement

IF ... THEN... ENDIF	structure alternative
FOR ... NEXT	structure répétitive à compteur
GOTO	saut inconditionnel
ON ... GOTO	structure de choix multiple (structure de cas)
GOSUB ... RETURN	lancement d'un sous-programme (s/p)

5. Conversion analogique numérique et numérique analogique

ADIN	assure la conversion analogique numérique
PWM	est également utilisé pour la CNA (associé à un circuit externe)
PWMOFF	

6. Commande de moteurs

PWM	crée un signal MLI pour un hacheur (fonctionne en tâche de fond)
PWMOFF	stoppe la tâche de fond
STEPOUT	commande d'un moteur pas à pas
SERVO	commande d'un servomoteur

7. Instructions faisant intervenir le temps

DELAY	produit une temporisation
CAPTURE	mesure la durée d'une impulsion
FREQOUT	produit une fréquence aléatoire

8. Gestion des Interruptions

ON TIMER ... GOSUB	lancement d'un s/p à la survenue d'un événement minuterie
ON INT ... GOSUB	lancement d'un s/p à la survenue d'un front sur un port

9. Liaison série "RS232"

SERIN entrée de données
SEROUT sortie de données

10. Liaison série "I2C ou SPI"

SHIFTIN entrée de données
SHIFTOUT sortie de données

11. Gestion et saisie de touches

ADKEYIN
PADIN
KEYIN

12. Gestion "eeprom"

EEWRITE
EEREAD

13. Génération sonore

SOUND
BEEP
PLAY

14. Instructions diverses

RND produit un nombre aléatoire
BCD assure la conversion d'un nombre en « Décimal Codé en Binaire »
BREAK instruction de mise au point
COUNT crée un compteur d'impulsions
PEEK
POKE

Règles générales d'utilisation

Les règles d'utilisation des instructions des PICBASIC sont très similaires à celles que l'on utilise pour la plupart des autres langages « BASIC ». Les instructions peuvent être classées en 2 catégories : les « commandes » et les « fonctions ».

Une fonction dispose généralement d'une expression exprimée entre deux « ( ) ».

- Exemples de de commandes: PRINT, GOTO, RETURN
- Exemples de fonctions: ADIN(0), EEREAD(0)

Les constantes peuvent être exprimées selon différents formats :

- Décimal: 10, 20, 32, 1234
- Héxadécimal: &HA, &H1234, &HABCD
- Binaire: &B10001010, &B10101

Vous pouvez utiliser les lettres minuscules lorsque vous écrivez les instructions de votre programme. Toutefois le langage des PICBASIC ne fera pas la distinction entre les lettres minuscules et les lettres majuscules (le langage les reconnaît toutes en tant que lettres majuscules). Par exemple, "LoopCNT" sera interprété comme « LOOPCNT ».

« Formatage » lors de la saisie du programme

La "rédaction" de votre programme nécessite comme tout langage une certaine "structure" et une "syntaxe" particulière. Celle-ci est très proche de la plupart des BASIC "standards".

En premier lieu, par convention chaque instruction de votre programme devra être au moins précédée d'un espace. Il vous sera possible d'attribuer un N° de ligne avant vos instructions afin d'effectuer des rappels et saut vers ce N° de ligne (dans ce cas, le N° devra impérativement être complètement "collé" sur la gauche de l'écran et il devra y avoir au minimum un espace de libre entre ce N° et l'instruction. Pour des raisons évidentes de lisibilité, il est conseillé d'utiliser la touche de tabulation pour générer un espace constant. Les N° de lignes sont optionnels, vous pouvez en mettre uniquement sur les lignes "importantes". Le "pas" d'incrémentation des N° n'a pas d'importance du moment que les numéros se suivent dans l'ordre chronologique (du plus petit au plus grand). Il est possible de mettre des commentaires au sein de votre programme. Pour ce faire, il suffit simplement que ce dernier soit précédé d'une apostrophe.

```
10      OUT 5,0
        DELAY 20
12      OUT 5,1
20      DELAY 20
        OUT 5,0
21      OUT 5,1             ' Ceci est un commentaire
```

Type de variables utilisables

Suivant le modèle de PICBASIC utilisé, il est possible d'exploiter 5 types de variables différentes.

- ***Byte*** : Nombre sur 8 bits non signé (0 ~ 255)
- ***Integer*** : Nombre sur 16 bits non signé (0~65535)

Déclaration des variables

Avant de pouvoir utiliser une variable dans laquelle vous pourrez stocker des données, il vous faudra au préalable déclarer celle-ci au début de votre programme afin que le "PICBASIC" réserve de la place au sein de sa mémoire "RAM".

>>> Comme vu précédemment, 2 types de variables sont déclarables avec les PICBASIC de la série « PB ».

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. A noter qu'une même instruction "DIM" peut servir à déclarer plusieurs variables. L'instruction DIM ne doit pas être collée à gauche de l'écran et au moins un espace (ou plusieurs) doit être ajouté à gauche de l'écran.

```
DIM   I   AS BYTE
DIM   J   AS INTEGER
DIM   K   AS BYTE, L   AS   BYTE
```

Les noms des variables doivent démarrer avec une lettre et la longueur des noms doit être inférieure à 255 caractères. Vous ne pouvez attribuer à une variable le nom d'une commande ou d'une fonction.

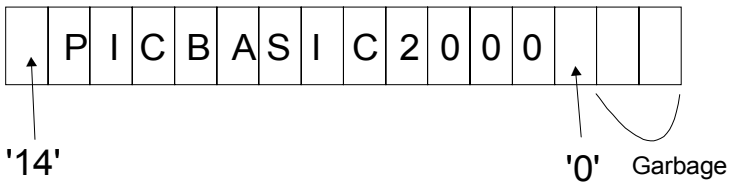
- Noms de variables pouvant être utilisés: A, B0, I, J, TH, BF1
- Noms de variables NE pouvant PAS être utilisés: 23, 3A, INPUT, GOTO

>>> Comme vu précédemment, 5 types de variables sont déclarables avec les PICBASIC de la série « PBM ».
Les règles de déclaration sont identiques à celles-vues ci-avant.

```
DIM   I   AS BYTE              ' Monopolise 1 octet en RAM
DIM   J   AS INTEGER           ' Monopolise 2 octets en RAM
DIM  K  AS LONG                ' Monopolise 4 octets en RAM
DIM  L   AS SINGLE             ' Monopolise 4 octets en RAM
DIM ST AS STRING*14            ' Déclare la variable TEXTE afin de pouvoir recevoir 16 caractères max.
```

Dans ce dernier cas (et comme pour toutes les variables de type STRING), le PICBASIC réservera toujours 2 octets supplémentaires par rapport à la taille des initialement réservée. Cette notion est donc très importante si vous utilisez un grand nombre de variables. Il vous faudra ainsi en tenir compte afin que vous ne tombiez pas à cours de RAM.

Ainsi dans l'exemple de déclaration précédent, 16 octets seront monopolisés (bien que la chaîne ST ne fasse que 14 caractères). La figure ci-dessous montre comment le PICBASIC stocke les données de type STRING au sein de sa mémoire.



Attention à bien déterminer la taille de votre variable lorsque vous définissez une variable de type STRING.

```
DIM ST AS STRING * 16                ' Déclare la variable ST comme une chaîne de 16 octets max.
ST = "COMFILE TECHNOLOGY"
```

Si vous essayez ensuite d'attribuer le nom COMFILE TECHNOLOGY à la variable ST, les 2 dernières lettres GY neseront pas mémorisée car il n'y aura pas eu assez de place (vous n'avez réservé que 16 octets).

Dans un même ordre d'idée, il est également possible de combiner des chaînes entres-elles.

```
SG = SG + ST                         ' Combinaison de chaînes
```

Il vous faudra aussi dans ce cas vérifier que vous ne dépassez la taille de la variable déclarée.

ATTENTION

A l'inverse des PICBASIC de la série « PB », les variables des PICBASIC de la série « PBM » ne sont pas initialisées à une valeur spécifique lors du RESET. Il vous faut donc impérativement prévoir au début de votre programme une initialisation systématique de toutes les variables déclarées.

Déclaration de tableaux

Vous pouvez également définir des tableaux à une dimension capables de contenir jusqu'à 65535 éléments (de nature différente suivantle type de PICBASIC utilisé).

```
DIM   A(20)  AS  BYTE          ' Définition d'un tableau à 20 éléments
DIM   B(200) AS  INTEGER       ' Tableau Integer
DIM   C(200) AS  LONG          ' Tableau array
DIM   D(200) AS  SINGLE        ' Tableau array
```

Pour la première ligne de déclaration, le paramètre du tableau démarre en 0. Ainsi il sera possible de disposer de 20 éléments en utilisant les positions de 0 à 19. De part la faible capacité de mémoire des PICBASIC de la série « PB » la définition maximale possible d'un tableau correspond à la taille mémoire RAM maxi du PICBASIC. Il ne sera ainsi pas possible de déclarer un tableau supérieur à 96 octets avec un PICBASIC-1S.

Il n'est pas non plus possible de réaliser des déclaratoin similaires à ci-dessous :

```
I = ARRAY1 (K(J))
```

Definition des constantes

Une des possibilités intéressantes des "PICBASIC" réside dans la possibilité de pouvoir attribuer une valeur à une constante. La déclaration se fera à l'aide de l'instruction "**CONST**" en tout début de programme. Dans l'exemple ci-dessous, le module "PICBASIC" dispose d'une LED connectée sur sa broche I/O 2. Lors de la déclaration initiale, on indiquera au "PICBASIC" que le "terme" LED équivaudra à la valeur 2. Dès lors, lorsqu'on voudra allumer la Led en sortie du "PICBASIC", il suffira de faire référence au "mot" LED, ce qui améliorera la lisibilité et la compréhension du programme.

```
10      CONST   LED = 2
20       OUT LED,1    ' Allume la LED
```

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

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

Selon le même principe, il est possible de déclarer d'autres type de « tableaux ».

```
CONST INTEGER DATA1 = (6000, 3000, 65500, 0, 3200)
CONST LONG DATA2 = (12345678, 356789, 165500, 0, 0)
CONST SINGLE DATA3 = (3.14, 0.12345, 1.5443, 0.0, 32.0)
```

Vous pouvez pour les déclarations nécessitant un grand nombre de données continuer la déclaration sur plusieurs lignes.

```
CONST BYTE DATA1 = (31, 25, 102, 34, 1, 0, 0, 0, 0, 0, 65, 64, 34,
            12, 123, 94, 200, 0, 123, 44, 39, 120, 239,
            132, 13, 34, 20, 101, 123, 44, 39, 12, 39)
```

Des données de type chaîne peuvent également être utilisée.

```
CONST BYTE DATA1 = ("I LOVE PICBASIC 2000", 13, 10, 0)
```

La différence entre la déclaration de tableaux et la déclaration de tableaux par le biais de l'instruction « CONST », réside dans le fait que les données des tableaux déclarées avec l'instruction « CONST » **sont enregistrées dans la mémoire « programme » du PICBASIC lors de l'opération de téléchargement.**

Dans le cas des autres type de tableau, ceux-ci sont enregistrés en mémoire SRAM (non conservés après un RESET) à l'inverse des tableaux déclarés par le biais de l'instruction « CONST » qui sont conservés après un RESET.

Contenu	Tableau	Tableau (CONST)
Mémorisation	Mémoire (SRAM)	Mémoire Programme (FLASH ou EEPROM)
Enregistrement	Pendant exécution programme	Pendant le téléchargement
Modification pendant l'éxécution du programme	Oui	Impossible
Utilisation « type »	Stockage de variables « dynamiques »	Stockage de données fixes
Etat après RESET	Initialisation	Conservées

CONST sous « PICBASIC Studio »

L'instruction **CONST** a enfin une dernière utilisation.

Lorsque vous utilisez un ordinateur fonctionnant sous Windows XP™ et que vous travaillez donc avec le logiciel « PICBASIC-Studio », il vous faudra alors déclarer (à la toute première ligne de votre programme) avec quel PICBASIC vous travaillez au moyen de l'instruction CONST. Ceci n'est pas nécessaire si vous utilisez un ordinateur sous Windows 98™.

EXEMPLE

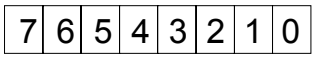
```
CONST DEVICE = 3H        ' Ceci signifiera que votre porgramme sera utilisé sur un PICBASIC-3H
```

En fonction du PICBASIC utilisé, le paramètre 3H pourra être remplacé par 1B, 1S, 2S, 2H, 3B, R1 ou R5.

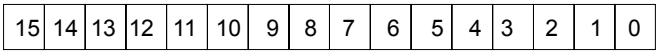
Acces direct aux « Bits »

Il est possible d'accéder directement aux bits d'une variable à l'aide d'un pointeur (le pointeur de Bit est valable de 0 à 31.) La gestion des Bits dépend de chaque type de variables. Pour définir un Bit avec les PICBASIC de la série « PB » il faut utiliser un point (.) tandis qu'avec les PICBASIC de la série « PBM » on utilisera deux points (:). La gestion des bits n'est pas possible avec les variables de type Single.

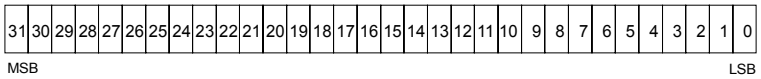
BYTE



INTEGER



LONG



PICBASIC de la série « PBM »

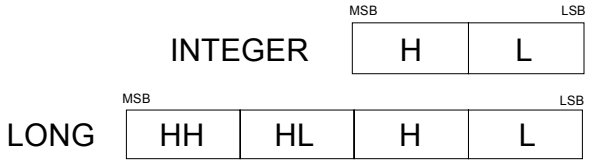
```
DIM I AS BYTE, A AS BYTE
I:7 = 0        ' force 7th bit de la variable Byte I à 0
I:7 = A:2      ' Transmet le 2nd bit de la variable A vers le 7th bit de la variable I
```

PICBASIC de la série « PB »

```
DIM I AS BYTE, A AS BYTE
I.7 = 0        ' force 7th bit de la variable Byte I à 0
I.7 = A.2      ' Transmet le 2nd bit de la variable A vers le 7th bit de la variable I
```

Acces direct aux « Octets »

Il est possible d'accéder directement aux octets de poids faible et fort d'une variable de type Integer/Long en utilisant un pointeur d'octets (non disponible avec les variables de type Single).



La façon d'utiliser le pointeur d'octet diffère également selon que vous utilisez un PICBASIC de la série « PB » ou de la série « PBM ».

Pour la série « PBM » :

Dans le cadre d'une variable de type Integer (2 octets), il est possible de diviser celle-ci en 2 parties (H et L)

```
DIM I AS INTEGER
I:H = 0          ' Place 0 sur l'octet de poids fort de la variable Interger I
I:L = 0          ' Place 0 sur l'octet de poids faible de la variable Interger I
```

Dans le cadre d'une variable de type Long (4 octets), il est possible de diviser celle-ci en 4 parties (HH, HL, H et L)

```
DIM K AS LONG
K:HH = 0         ' Place 0 sur l'octet (haut) de poids fort de la variable Long K
K:HL = 0         ' Place 0 sur l'octet (haut) de poids faible de la variable Long K
K:H = 0          ' Place 0 sur l'octet (bas) de poids fort de la variable Long K
K:L = 0          'Place 0 sur l'octet (bas) de poids faible de la variable Long K
```

Pour la série « PBM » :

On utilise le caractère point (.) en remplacement des 2 points (:) utilisés sur la série « PBM ».

```
I.H = 0
I.L = 0
M = I.H
```

Expressions

Les expressions avec les PICBASIC peuvent être déclarées comme suit (suivant les modèles utilisés) :

```
I = 0              ' Met la variable I à 0
I = I + 1          ' Augmente la variable I d'une unité
I = J * 12 / K     ' Multiplie la variable J par 12 et divise par la variable K
```

Ce tableau montre la liste des opérateurs utilisables (variable suivant le type de PICBASIC).

Opérations arithmétiques (Pour tout type de variable)	Opérations logiques et décalage (Sur variable de type Integer seulement)
+ Adition	AND Opération ET Logique
- Soustraction	OR Opération OU Logique
* Multiplication	XOR Opération XOR Logique
/ Division	<< décalage à gauche
	>> décalage à droite
	MOD reste de la division

Ordre de priorité des opérations : multiplication, division, opération sur bit, addition et soustraction.

```
I = J + 12 * K          ' Multiplication de 12 par K et addition avec J
I = J + I AND &HF       ' Exécute un ET entre I et &HF puis additonne avec J
```

L'écriture d'opérations complexes peut provoquer des erreurs lors de la compilation. Dans ce cas, il est recommander de scinder les opérations en plusieurs parties.

```
I = (J * K) + L / 4   ⇨   I = J * K
                          I = I + L / 4
```

Si l'opération ne nécessite aucun calcul prioritaire, il est possible de saisir une opération via une expression assez longue.

```
I = J * K * L / 4 + 100
```

ATTENTION

Les PICBASIC ne permettent pas l'utilisation d'une fonction à l'intérieur d'une autre fonction.

```
PRINT DEC(ASC(ST))        ' Cette fonction ne marchera pas. .. Elle devra être décomposée comme ci-dessous.

I = ASC(ST)
PRINT DEC(I)
```

Dans le cas des PICBASIC de la série « PBM » vous pouvez diviser les lignes d'instructions trop longues en plusieurs lignes à l'aide du caractère "_". (Ceci n'est pas disponible sur les PICBASIC de la série « PB »)

```
I = TABLE(J,192, 12, 13, 142, 123, 0, 0, 0, 1, 2, 3,_
          234, 192, 14, 90, 100, 200, 0, 0, 0, 1)
```

De plus, il est interdit d'utiliser une expression d'opération au milieu d'une commande.

```
IF  A+2 = 0 THEN K =0        '' Cette fonction ne marchera pas. .. Elle devra être décomposée comme ci-dessous

B = A + 2
IF B=0 THEN K = 0
```

Gestion de valeurs à virgule avec les PICBASIC « PBM »

Pour attribuer une valeur décimale à virgule à une variable de type 'SINGLE' avec les PICBASIC de la série « PBM », il vous faudra utiliser une variable 'STRING' de passage associée à l'instruction 'VALSNG'.

```
DIM   VALEUR AS STRING*16
      DIM   NB AS SINGLE

      VALEUR = "1234,5678"
      NB = VALSNG(VALEUR)
```

En cas de calculs ou de comparaisons entre une variable de type 'SINGLE' avec des variables de type 'BYTE', 'INTEGER' ou 'LONG', il vous faudra impérativement procéder à une conversion préalable de ces dernières à l'aide de l'instruction sous peine d'obtenir un résultat erroné.

Gestion de valeurs à virgule avec les PICBASIC « PB »

Pour utiliser des nombres à virgule à l'aide des PICBASIC de la série « PB », vous devrez utiliser des astuces de programmation.

Par exemple il ne vous sera pas possible d'origine de multiplier 200 par 3.14 avec les PICBASIC de la série « PB ». Dans ce cas, il vous faut avoir recours à une méthode de « mise à l'échelle » pour obtenir le même résultat (ou une approximation de ce résultat plus exactement). Voir l'exemple ci-dessous pour plus d'infos.

```
' La multiplication de 200 * 3.14 donne 628
'
SET PICBUS HIGH
LCDINIT
DIM I AS INTEGER
I = 200
I = 200 * 3
I = 200 * 1 / 10 + I
I = 200 * 4 / 100 + I
LOCATE 0,0
PRINT DEC(I)                    ' Affiche le résultat sur le LCD.
10    GOTO 10
```

Dans le même esprit, lorsque vous voulez convertir une valeur issue d'une entrée de conversion analogique/numérique récupérée sous la forme d'une valeur entre (0~255) et que vous voulez avoir l'équivalent sous une valeur comprise entre (0~1000), il vous faut avoir recours à l'opération : 1000 / 256 = 3.90625

Ce qui revient à multiplier la valeurde la mesure « A/N » par 3.90625. Le programme ci-dessous montre comment faire :

```
SET PICBUS HIGH
LCDINIT
DIM I AS INTEGER
DIM J AS INTEGER
10    I = ADIN(0)
J = I * 3
J = I * 9 / 10 + J
LOCATE 0,0
PRINT DEC(I)          ' Valeur de la conversion « A/N » surla ligne du dessus.
LOCATE 0,1
PRINT DEC(J)          ' Résultat de la conversion sur la ligne du dessous.
GOTO 10
```

Parce que l'on utilise uniquement 3.9 comme chiffre multiplicateur, une erreur apparaît dans le résultat de la conversion (le maximum affichable sera alors de 994). Toutefois ces astuces sont très pratiques avec les PICBASIC de la séie « PB ».

Description détaillée des instructions

Lors de la description détaillée des instructions des PICBASIC, vous pourrez voir apparaître des petits « logo » à droite de l'instruction. Voici la description de ces derniers :

PBM : Instruction utilisable seulement avec les PICBASIC de la série « PBM » (PBM-R1 / PBM-R5)

PB : Instruction utilisable seulement avec les PICBASIC de la série « PB » (PICBASIC 1B / 1S / 2S / 2H / 3B / 3H)

Si aucun logo n'est présent, c'est que l'instruction peut être utilisée avec TOUS les PICBASIC.

NOTE :
Lors de la description détaillée des instructions, nous ne faisons pas systématiquement apparaître les déclarations des variables utilisées dans les petits programmes donnés en exemples.

De même, nous ne faisons pas apparaître la déclaration CONST DEVICE = 3B (ou 1B, 1S, 2S, 2H, 3B, R1 ou R5) nécessaire en début de programme si vous utilisez le logiciel de programmation « PICBASIC-Studio » sous Windows XP™.

ADIN()

*Variable Integer = ADIN (***port***)

CONVERSION « A/N »

Port est une variable/contante de type Integer relative à une entrée de conversion « A/N ».

EXPLICATION

Cette instruction permet de connaître la valeur de la tension analogique présente sur une broche de conversion « analogique / numérique » particulière du module "PICBASIC" (**sauf le "PICBASIC-1B"**). La valeur à lire doit être comprise entre 0 et + 5 V (pour des valeurs plus élevées, il sera nécessaire d'avoir recours à des ponts diviseurs à l'aide de résistances, en s'assurant toujours que la tension ne dépasse jamais + 5 V, sous peine de destruction du port d'entrée du "PICBASIC" (non pris en compte par la garantie). Le paramètre (**Port**) correspond à la broche du module qui recevra la valeur à mesurer. Se référer aux brochages des PICBASIC pour connaître les pattes bénéficiant d'une fonction de conversion analogique/numérique et pouvant être utilisées pour cette mesure (voir ci-dessous). Le nombre obtenu en résultat d'une conversion est directement proportionnel à la valeur de la tension d'entrée: Dans tous les cas, il est impératif que les fils de connexions entre le signal analogique à mesurer et l'entrée du port de conversion du PICBASIC ne dépassent quelques cm afin de ne pas être perturbé par des parasites externes.

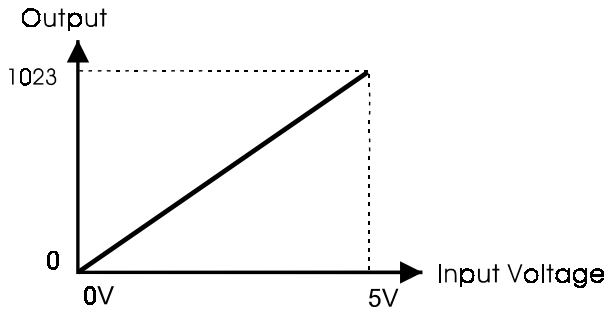
Avec les PICBASIC-1S / 2S / 2H les convertisseurs bénéficient d'une résolution sur 8 bits:
Pour 0 V en entrée -> on obtient le nombre 0
Pour 2.5 V en entrée -> on obtient le nombre 128
Pour 5.0 V en entrée -> on obtient le nombre 255.

EXEMPLE

```
I = ADIN(0)              ' Récupère le résultat de la conversion dans la variable I
SEROUT 8,93,0,0,[I]      ' Envoi la valeur sur le port série
```

Avec les PICBASIC-3B / 3H / PBM-R1/ PBM-R5 : les convertisseurs bénéficient d'une résolution sur 10 bits:
Pour 0 V en entrée -> on obtient le nombre 0
Pour 2.5 V en entrée -> on obtient le nombre 512
Pour 5.0 V en entrée -> on obtient le nombre 1023

*** Signifie que le résultat de cette instruction deviendra une donnée de type « Integer »**



INFORMATIONS COMPLEMENTAIRES

Le module PBM-R5 dispose de 2 entrées de conversion « A/N » supplémentaires dotées d'une résolution de 12 bits. Lorsque vous utilisez cette instruction sur les ports 32 et 33, vous récupèrerez une valeur comprise entre 0 et 4095.

```
DIM K AS INTEGER         ' Déclaration de la variable K en Integer (16 bits)
K = ADIN(32)             '  Récupère la valeur de la conversion sur 12 bits dans la variable K
```

Attention : les ports 32 et 33 ne peuvent être utilisés qu'en tant qu'entrées de conversion « analogique / numérique » (les entrées ne peuvent pas être utilisées comme des entrées / sorties tout-ou-rien standards).

ADKEYIN ()

Variable Byte = ADKEYIN (*port*)

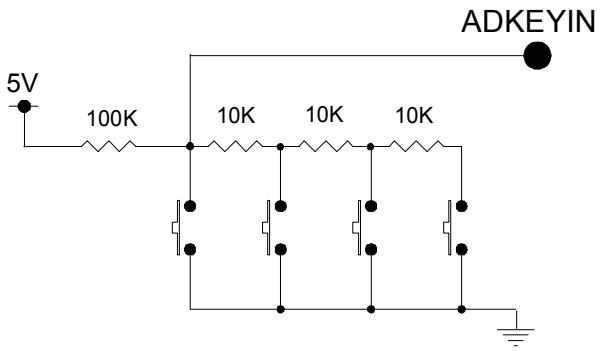
Gestion de touches via entrées de conversion « A/N »

Port est une variable de type Byte comprise entre 0 et 33 ou le numéro d'une entrée de conversion "A/N".

EXPLICATION

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-dessous (nous disposons également d'une petite platine prête à l'emploi sous la référence « ADKEY BOARD »). Le principe de fonctionnement est en fait très simple et repose sur celui des ponts diviseurs. Lorsque le bouton-poussoir le plus près de la broche du "PICBASIC" est sollicité, l'entrée se retrouve avec une tension de 0 V en entrée, dès lors l'instruction "ADKEYIN" retourne la valeur "1". Lorsque le second bouton-poussoir est sollicité à son tours, on se retrouve avec une tension de 450 mV et la valeur "2" est retournée. La troisième touche génère 830 mV et retourne la valeur "3" enfin la quatrième touche génère environ 1,15V et retourne la valeur "4" et ainsi de suite... Il est possible de gérer jusqu'à 10 touches (associée chacune à une résistance) par broche dédiée à une conversion analogique / numérique (il vous sera ainsi possible de gérer entre 50 et 80 touches au maximum suivant le type de module "PICBASIC utilisé). Attention, il ne sera pas possible d'utiliser cette instruction avec le **PICBASIC-1B** (car il ne dispose pas d'entrée de conversion « A/N »). En absence de sollicitation des boutons-poussoirs, la valeur retournée est "0". Lors du montage, vérifiez à toujours utiliser une tension de référence de +5V sous peine de destruction de l'entrée du "PICBASIC" - non prise en compte dans la garantie). On notera enfin que si plusieurs touches sont sollicitées en même temps, seule la valeur de la touche la plus "basse" chronologiquement sera retournée.

EXEMPLE




```
10   I = ADKEYIN(0)          ' récupère la valeur de la touche sur le port 0.
     PRINT HEX(I)            ' Affiche la valeur de la touche sur l'écran LCD
     GOTO 10
```

INFORMATIONS COMPLEMENTAIRES

Il n'est pas possible d'utiliser l'instruction « ADKEYIN » sur les ports de conversion « A/N » 12 bits du « PBM-R5 ».

Si des valeurs de touches erronées sont détectées ou que des touches ne sont pas détectées, il vous faudra ajuster légèrement la valeur de la résistance « talon » de 100 KΩ. Ce symptôme est du à une erreur de conversion « A/ N » pouvant survenir sous l'influence de perturbations externes, signaux parasites, etc…

A ce titre, il est impératif que les fils de connexions entre tous les boutons poussoirs et le PICBASIC ne dépassent pas quelques cm afin d'éviter que le PICBASIC ne soit perturbé par des parasites externes.

BCD ()

Variable Integer = BCD (*Valeur*)

Conversion BCD

Valeur est une constante comprise entre 0 et 65535 ou une variable de type Byte / Integer.

EXPLICATION

Cette instruction permet de convertir le format d'un nombre en sa formulation «Décimal Codé en Binaire».

Si la valeur à convertir dépasse 2 chiffres (>99), il vout faudra déclarer les variables en INTEGER.
Si la valeur est supérieure à 9999, l'instruction BCD ne pourra plus être utilisée.

En cas contraire une erreur interviendra dans la conversion. Si par exemple vous essayez de convertir 12345 en BCD dans une variable de type INTEGER, vous n'obtiendrez que 2345.

Avant conversion BCD (Décimal)	Après conversion BCD (Décimal)	Après conversion BCD (Héxadécimal)
1	1	&H01
2	2	&H02
3	3	&H03
4	4	&H04
5	5	&H05
6	6	&H06
7	7	&H07
8	8	&H08
9	9	&H09
10	16	&H10
11	17	&H11
12	18	&H12
13	19	&H13
14	20	&H14
15	21	&H15

EXEMPLE

```
DIM A AS INTEGER, I AS INTEGER
A = 13
I = BCD(A)              ' Conversion valeur de la variable A en BCD
LOCATE 0,0
PRINT DEC(I)            ' Affiche le résultat sur un écran LCD
```

BEEP

BEEP *port*

Génération d'un bip sonore

Port est une constante de valeur comprise entre 0 et 31 ou une variable de type "Byte".

EXPLICATION

Cette instruction génère un signal carré de fréquence 2 KHz pendant quelques milli-secondes 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-dessous). Le paramètre "Port" indique la broche où sera connecté le buzzer. Les PICBASIC « PBM-R1 / PBM-R5 » ne peuvent utiliser que les ports 0 à 15. Le buzzer devra être câblé au plus près de la broche du PICBASIC (quelques cm de fil max.).

EXEMPLE

```
20   IF KEYIN(1)=1 THEN GOTO 20      ' Attend la sollicitation d'une touche connectée sur le port 1 (le port passe alors à 0)
     BEEP 10                         ' Génère un bip sonore lorsque la touche est sollicitée.
     GOTO 20
```

BREAK

BREAK

Break

EXPLICATION

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 Window' afin que vous puissiez vérifier les valeurs de toutes les variables et éventuellement tester votre programme en mode "pas-à-pas" (voir explications détaillées aux chapitres 3 et 6). 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 est impératif (une fois le programme complètement mis au point) de supprimer toutes les instructions 'Break' de son programme.

ATTENTION

Si vous placez une instruction BREAK dans une boucle sans fin très rapide, vous générerez des arrêts de programmes à répétition (pouvant être à l'origine d'erreurs de fonctionnement du PICBASIC). Placez donc toujours les instructions BREAK de telle sorte qu'elles ne soient pas appelées de façon trop répétitive (l'idéal est d'avoir recours à cette instruction suite à la sollicitation d'une entrée particulière de votre PICBASIC par exemple). Si vous avez en revanche placé l'instruction BREAK dans une boucle sans fin rapide, utilisez la commande du menu ERASE ALL dans le programme du PICBASIC du PC pour sortir de la boucle.

BUSOUT

BUSOUT *Valeur*, [*Valeur*], ...

Transmission de codes spécifiques sur le PICBUS

Valeur est une constante comprise entre 0 et 255 ou une variable de type Byte.

EXPLICATION

La broche "PICBUS" de chaque module "PICBASIC" est spécialement conçue pour piloter des afficheurs à commandes séries spéciaux (appelés « ELCDxxx ») par le biais d'instructions spécifiques (locate, PRINT, etc…) qui envoient une des d'ordres à ces derniers.

L'utilisateur a également la possibilité de piloter ces afficheurs séries en utilisant l'instruction "BUSOUT".

EXEMPLE

```
10        BUSOUT &HA1; &H00; &H00; &HA2; &H41; &H42; &H43; &H44; &H00
20        LOCATE 0,0
30        PRINT "ABCD"                ' Consultez la documentation des afficheurs à commande série pour plus d'infos
```

La ligne 10 provoque le même résultat que les lignes 20 et 30. Les signaux générés par les instructions spécialisées pour la gestion des afficheurs séries ou l'instruction "BUSOUT" sont de type série (5 V / niveau TTL). Le débit 4800 ou 19200 Bds est fonction de la déclaration faite par les instructions "PICBUS HIGH" ou "PICBUS LOW".

BYTEIN ()

Variable Byte = BYTEIN (*port block*)

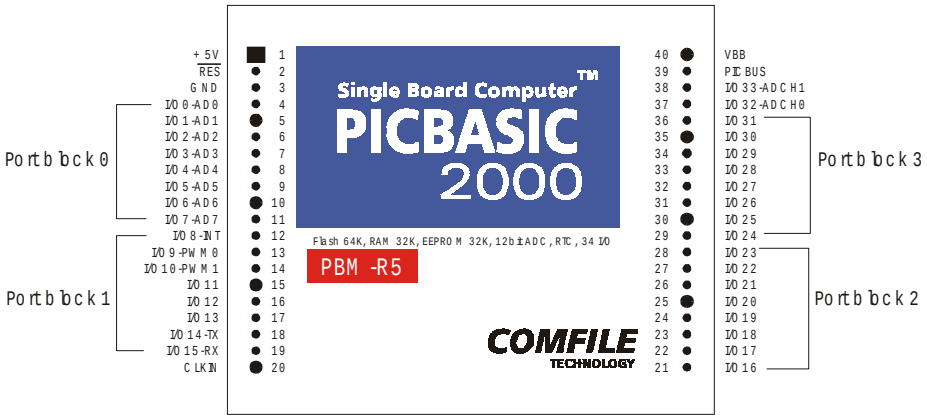
Gestion d'entrées sur 8 bits

Port block est un numéro de bloc (0 à 3) ou une variable de type Byte.

EXPLICATION

Cette instruction permet de récupérer la valeur de 8 entrées du module "PICBASIC" dans un "mot binaire" 8 bits dont chaque bit est l'image de chacune des entrées. Il est ainsi possible de gérer de 1 à 4 blocs de 8 bits différents (donc 4 x 8 entrées max.) suivant le modèle de PICBASIC utilisé.

EXEMPLE




```
DIM I AS BYTE                ' I est une variable de type Byte
I = BYTEIN(0)                ' Lecture de l'état du block 0
```

ATTENTION

Aucun des ports présent au sein d'un block ne pourra être exploité en sortie si vous utilisez l'instruction BYTEIN (en effet, tous les ports du block seront automatiquement configurés en entrées).

BYTEOUT

BYTEOUT *port block*, *Valeur*

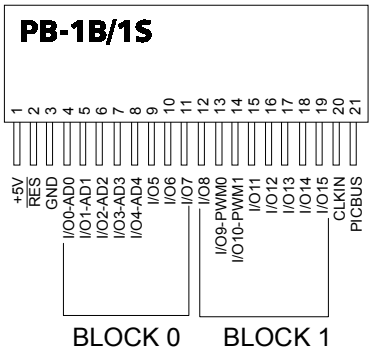
Sortie d'une valeur sur 8 bits

Port block est un numéro de bloc (0 à 3) ou une variable de type Byte.
Valeur est une contante (0 ~ 255) ou une variable de type Byte.

EXPLICATION

Cette instruction « sort » une ***Valeur*** 8 bits sur le ***port block***. Tous les ports d'un ***port block*** utilisés avec l'instruction BYTEOUT seront automatiquement configurés en sortie.

EXEMPLE




```
I = &HAB
BYTEOUT 0,I              ' Sort le contenu de la variable I sur le port block 0
```

CAPTURE ()

Variable Integer = CAPTURE (*port, target*)

Capture d'impulsion

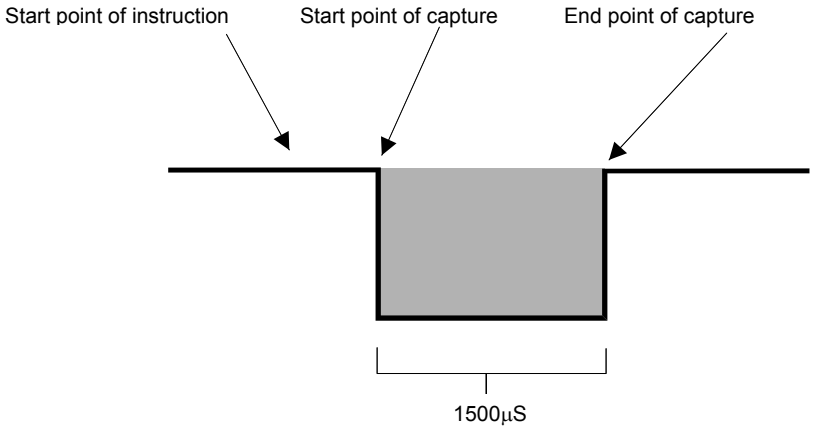
Port est le port de la mesure pouvant être défini par une constante (0~3) ou une variable de type Byte.
Target est une contante (0 ou 1 seulement) – Ce paramètre ne peut pas être utilisé avec une variable.

EXPLICATION

Permet de mesurer la durée d'une impulsion d'un signal extérieur (niveau haut "1" ou bas "0" selon "Target") sur une broche (Port) du "PICBASIC". Le nombre récupéré est soumis à un facteur de réduction. Ainsi pour les "PICBASIC-1B / 1S / 2S", il doit être multiplié par "20" pour obtenir la valeur réelle - pour les "PICBASIC-2H/3B/3H", il doit être multiplié par "4" et par "7" pour les "PBM-R1 / PBM-R5". Ceci veut dire qu'il ne sera pas possible de mesurer des largeurs d'impulsions supérieures à 1,31 s pour les PICBASIC-1B / 1S / 2S et supérieur à 0,26 s pour les "PICBASIC-2H/3B/3H" (le calcul est très simple: Nombre max= 65535 (car déclaré en INTEGER) * 20 = 1310700 ms et 65535 x 4 = 262140ms).

EXEMPLE

```
I = CAPTURE(0,0)          'Mesure la largeur de l'impulsion basse sur le port "I/O O"
```



Dans l'exemple ci-dessus, le calcul de la durée de l'impulsion commence lorsque le niveau bas est détecté <Start point of instruction> et se termine lors de la détection du niveau logique haut <end point of instruction>. Si la détection du niveau haut <end point of capture> intervient en dehors de la largeur maximale mesurable, l'instruction retournera la valeur 0, sans attendre le retour au niveau haut <end point of instruction>.

	PB-1B/1S/2S	PB-2H	PB-3B/3H	PBM-R1/R5
Largeur maxi. mesurable	1.31 sec	0.26 sec	0.26 sec	0.4 sec
Valeur retorunée	Interval capturé/ 20	Interval capturé / 4	Interval capturé / 4	Interval capturé / 7

Le fil de raccordement ramenant le signal d'impulsion à la broche du PICBASIC devra être le plus court possible et ne pas excéder quelques cm.

CONST DEVICE

CONST DEVICE = *device*

Declararation d'une constante ou du type de PICBASIC utilisé

Cette instruction a 2 usages :

1) Lorsque vous utilisez un ordinateur fonctionnant sous Windows XP™, il vous faudra alors déclarer (au tout début de vore programme) avec quel PICBASIC vous travaillez au moyen de l'instruction CONST. Ceci n'est pas nécessaire si vous utilisez un ordinateur sous Windows 98™.

EXEMPLE

```
CONST DEVICE = 3H        ' Ceci signifiera que votre porgramme sera utilisé sur un PICBASIC-3H
                           3H sera remplacé par 1B, 1S, 2S, 2H, 3B, R1 ou R5 suivant le type de PICBASIC utilisé.
```

2) Cette instruction permet également de remplacer une constante par une expression au sein de votre programme afin d'en faciliter la lecture et la mise en oeuvre.

EXEMPLE

```
10      CONST  BUZZER  =  5
20      BEEP BUZZER
```

Dans cet exemple, la sortie "I/O 5" est reliée à un buzzer. Afin de simplifier la lisibilité du programme et après l'exécution de l'instruction CONST, il suffira d'écrir le mot "BUZZER" pour désigner en fait la sortie "I/O 5".

COUNT ()

Integer variable = COUNT (*parametre*)

Entrée de comptage

Parametre est une constante (0 ou 1) 0 = Maintenu, 1= RAZ Celle-ci ne peut pas être remplacée par une variable

EXPLICATION

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** (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 ou 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) - Respectez impérativement le niveau d'entrée maximal de +5V sous peine de destruction de l'entrée "CLKIN" (non pris en compte dans la garantie). Le compteur est géré sur 16 bits (jusqu'à 65535). En cas de dépassement du compteur, celui-ci repasse à zéro.

EXEMPLE

```
I = COUNT(0)
```

La variable I contiendra le nombre d'impulsions comptabilisées depuis la mise sous tension du module. Si l'instruction COUNT est de nouveau appelée, le nombre des nouvelles impulsions comptabilisées s'ajoutera à celui de celles déjà comptabilisées.

```
I = COUNT(1)
```

La variable I contiendra le nombre d'impulsions comptabilisées depuis la mise sous tension du module et le compteur interne d'impulsions est automatiquement remis à zéro. Si l'instruction COUNT est de nouveau appelée, le nombre des nouvelles impulsions comptabilisées pourra alors être connu.

Le fil de raccordement ramenant le signal des impulsions à comptabiliser sur la broche du PICBASIC devra être le plus court possible et ne pas excéder 1 à 2 cm.

CSRON

CSRON

Active l'apparition du curseur d'un afficheur LCD à commandes séries

EXPLICATION

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

CSROFF

CSROFF

Désactive l'apparition du curseur d'un afficheur LCD à commandes séries

EXPLICATION

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

CLS

CLS

Efface le contenu d'un écran LCD à commandes séries

EXPLICATION

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

DELAY

DELAY *val*

Delais en mS

mS est une constante/variable de type Byte/Integer permettant de définir une temporisation exprimée en millisecondes.

EXPLICATION

Cette instruction permet de générer une temporisation dont la durée est fonction de la valeur de "Val". Cette durée exprimée en milli-secondes peut s'étendre de 1 à 255 ms (ou jusqu'à 65535) suivant le type de PICBASIC.

A noter que cette instruction ne dispose pas d'une extrême précision.

EXEMPLE

```
DELAY 20      ' Effectue une temporisation de 20 ms.
```

FREQOUT

FREQOUT *port*, *valeur*

Output desired frequency

Port est une constante de type Byte (9 ou 10).
Valeur est une constante de type Byte comprise entre (1 - 253).

EXPLICATION

Cette instruction permet de générer un signal rectangulaire sur une broche spécifique (**Port**) du "PICBASIC" de fréquence pré-définie par (Valeur). Seules les broches "PWM0" (I/O 9) et "PWM1" (I/O10) des modules "PICBASIC" peuvent être affectées à cette tâche. La valeur de la fréquence de sortie peut être déterminée à l'aide des tableaux de correspondance ci-dessous. A noter que le signal rectangulaire est généré en tâche de fond et que le "PICBASIC" peut réaliser d'autres instructions tout en continuant de délivrer son signal de sortie. L'instruction PWMOFF permet de stopper le signal.

EXEMPLE:

```
10        FREQOUT  9,191   ' Géénère un signal de l'ordre de 1 KHz sur le port 9 d'un PICBASIC-1S
20        DELAY    255
30        PWMOFF   9
```

PB-1B/1S/2S

Parameter	Frequency
1	258Hz
10	267.5Hz
20	278.7Hz
40	304.7Hz
80	374Hz
100	421.8Hz
191	***1KHz***
200	1.17KHz
230	2.5KHz
253	21.9KHz

PB-2H/3B/3H

Parameter	Frequency
1	1.227KHz(2H), 19.6KHz(3B/3H)
10	1.274KHz(2H), 20.3KHz(3B/3H)
20	1.324KHz(2H), 21.19KHz(3B/3H)
40	1.439KHz(2H), 23.15KHz(3B/3H)
80	1.77KHz(2H), 28.4KHz(3B/3H)
100	2KHz(2H), 32.5KHz(3B/3H)
153	***3KHz(2H), 48.5KHz(3B/3H)***
200	5.58KHz(2H), 89.28KHz(3B/3H)
230	12KHz(2H), 192.3KHz(3B/3H)
253	104KHz(2H), 1600KHz(3B/3H)

La valeur maximale de **valeur** est 253 (si vous utilisez 254 ou 255, aucune fréquence ne sera générée).

ATTENTION

Il n'est pas possible d'utiliser l'instruction FREQOUT simultanément sur les ports 9 et 10. Il n'est pas non plus possible d'utiliser les instructions de génération de signaux PWM en même temps que l'instruction FREQUOUT (l'inverse est également vrai, si vous utilisez une instruction PWM, vous ne pourrez alors pas utiliser l'instruction FREQUOUT en même temps).

Les valeurs de fréquences indiquées dans le tableau peuvent être soumises à de légères variations.

FOR...NEXT

FOR *variable1*=*val1* TO *val2* [STEP–increment]...NEXT *val3*

Boucle FOR…NEXT

Variable est une variable de type Byte/Integer.
Val1 est une constante de type Byte/Integer correspondant à la valeur de départ.
Val2 est une constante de type Byte/Integer correspondant à la valeur de fin.
Val3 est une constante de type Byte (-128~+127)/ représentant le pas devariation (utilisable uniquement sur série « PBM »)

EXPLICATION

Cette instruction permet de réaliser plusieurs fois de suite certaines actions comprises entre les instructions "FOR" et "NEXT" de votre programme. Le nombre de "répétition" de ces actions sera déterminé par les valeurs de **Val1** et **Val2**. Par exemple : “FOR I=0 TO 50” provoquera l’exécution des commandes/instructions entre FOR et NEXT 51 fois.

EXEMPLE 1:

```
10        DIM  I AS  BYTE
20        FOR  I  =  0   TO   5
30          BEEP  4
40        NEXT  I
```

EXEMPLE 2:

```
10        DIM  I  AS  BYTE
20        DIM  J  AS  BYTE
30        FOR   I =  0   TO   5
40          FOR  J =  0   TO   4
50            BEEP 4
60          NEXT  J
70        NEXT  I
```

Pour une meilleure "lisibilité" de votre programme, il est conseillé de décaler légèrement les instructions qui se trouvent entre "FOR" et "NEXT" vers la droite.

Pour les PICBASIC de la série « PBM », il est possible d'adjoindre un pas (**Val3**) de comptage (ou de décomptage).

EXEMPLE 3:

```
10   FOR  I  =  0   TO   50   STEP 3
              .
          .
50   NEXT  I                          ' La variable I va prendre successivement les valeurs 0, 3, 6 ...


60   FOR  I  =  50   TO   0   STEP -3
              .
          .
90   NEXT  I                          ' La variable I va prendre successivement les valeurs 50, 47, 44 ...I
```

GOTO

GOTO *ligne*

Saut inconditionnel

Ligne est l'endroit où le programme doit poursuivre son exécution

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

EXEMPLE 1:

```
10        DIM  I  AS   BYTE
20        I  =  I  +  1
30        GOTO 20
```

Exemple 2:

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

INFORMATIONS ADDITIONELLES
Il existe 2 sortes d'identifiants pour les sauts inconditionnels :

- Les N° de ligne
- Les étiquettes

Les N° de ligne sont en quelque-sorte des étiquettes constituées de chiffres.

```
10        PRINT  DEC(I)
          GOTO 10
```

Les étiquettes permettent d'améliorer la lisibilité de vos programmes en utilisant des « noms ».
Une étiquette doit commencer par une lettre de l'alphabet et terminer par :

```
          GOSUB DELAY10
          ;
DELAY10:  FOR  I =0  TO  10
            NEXT I
            RETURN
```

Vous pouvez utiliser des étiquettes de 255 caractères max.

Exemples d'étiquettes valides	Exemples d'étiquettes non valides
NOKEY:	123AB: Non car commence par un nombre
DELAY_RTN:	IN: Non car il s'agit du même nom qu'une instruction
ACCIN_PROC:	

GOSUB...RETURN

GOSUB *ligne*, RETURN

Appel d'une sous-routine

Ligne est l'endroit où le programme doit poursuivre son exécution

EXPLICATION

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").

EXEMPLE 1:

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

100       DELAY     255
110       RETURN
```

INFORMATIONS ADDITIONELLES

A noter qu'il est également possible imbriquer plusieurs sous-routines les unes dans les autres jusqu'à concurrence de 6 max. (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). Le nombre de sous-routines pouvant être imbriquées les unes dans les autres peut donc être plus limité suivant l'occupation de la RAM par le PICBASIC. Il est donc impératif de faire des essais successifs en cas d'imbrications multiples afin de vérifier que le PICBASIC n'est pas en dépassement sans quoi ce dernier pourra être amené à faire des actions incohérentes et aléatoires.

EXEMPLE 2:

```
10        CONST   BUZZER  =  5
20        GOSUB    100
30        GOTO     20

100       BEEP   BUZZER
110       GOSUB 200
120       RETURN

200       DELAY  255
210       RETURN
```

Il est également impératif de vérifier que chaque instruction GOSUB est bien relayée par l'instruction RETURN associée, sans quoi des erreurs dans la gestion de la pile de retour des GOSUB interviendront (sans que ceci ne vous soit annoncé lors de la compilation du programme). Ceci aura pour conséquence de générer des actions incohérentes et aléatoires de la part du PICBASIC.

```
        ' Exemple de gestion correct
        GOSUB   1000
        :
1000    I = I + 1       ' Sous-Routine
        RETURN
```

```
        ' Exemple de gestion non correct  (le GOSUB n'est pas relayé par RETURN)
                GOSUB 1000
        2000
                :
1000            I = I + 1       ' Sous-Routine
                GOTO 2000
```

IF...THEN

IF *expression* THEN...ELSE...

Action conditionnée

EXPLICATION
Cette instruction permet de réaliser des “actions” en fonction de tests et de conditions définies par vos soins.

EXEMPLE 1:

```
10        DIM  I  AS  BYTE
20        IF   I  >  5   THEN   GOTO 50          ' Continu l'exécution du programme si I > 5
```

EXEMPLE 2:

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

EXEMPLE 3:

```
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          ENDIF
```

EXEMPLE 4:

```
10        IF  I < 10   THEN
20          PRINT “I < 10”
30        ELSE   IF   I < 80   THEN
40             PRINT “10 < I <80”
50          ELSE
60            PRINT “I > 80”
70        END IF
```

On notera que les décalages vers la droite des instructions internes à la boucle IF ... THEN / ENDIF ne sont pas absolument nécessaires, mais conseillés du fait qu'ils participent à une plus grande clarté et lisibilité du programme.

EXPRESSIONS CONDITIONNELLES
De même il est important de signaler que les tests peuvent également faire l'objet de conditions plus étendues:

EXEMPLES 5:

```
IF I<>5 THEN    ‘Si “I” différent de 5 alors...
IF I>=5 THEN    ‘Si “I” supérieur ou égal à 5 alors...
IF I>5 THEN     ‘SI “I” supérieur à 5 alors...
IF I<=5 THEN    ‘Si “I” inférieur ou égal à 5 alors...
IF I<5 THEN     ‘SI “I” inférieur à 5 alors...
IF I=5 THEN     ‘SI “I” égal à 5 alors...
```

Il est également d'utiliser des conditions additionnelles du type “AND” / “OR” (“ET” / “OU”)

```
IF I<5 AND I>10 THEN    ‘Si “I” inférieur à 5 et “I” supérieur à 10 alors...
IF I=5 OR I=10 THEN     ‘Si “I” égal à 5 ou “I” égal à 10 alors...
```

NOTE #1
Il n'est pas possible d'utiliser des parenthèses pour délimiter des conditions au sein d'une instruction "IF" – ces parenthèses ne seront pas « traitées » lors de la compilation du programme.

Si pour écrivez par exemple ce programme :

```
IF ((A > 1) AND (A < 10)) OR ((B > 10) AND (B < 20)) THEN
      K = K + 1
END IF
```

En fait le programme sera interprété comme ci-dessous.

```
IF A > 1 AND A < 10 OR B > 10 AND B < 20 THEN
      K = K + 1
END IF
```

Dans ce cas de figure, il vous faudra séparer les tests conditionnels en 2 tests distincts

NOTE #2
Il est impératif que les conditions de comparaison se fassent sur des données de même type sans quoi le PICBASIC ne pourra pas fonctionner correctement.

NOTE #3
Il n'est pas possible d'effectuer des comparaisons entre des chaînes de caractères.

```
DIM ST1 AS STRING * 12
DIM ST2 AS STRING * 12
IF ST1 = ST2 THEN GOTO 100
```

Ceci ne fonctionne pas !

IN ()

Variable Integer = IN (*port*)

Gestion des entrées

Port est une constantes (0~31).

EXPLICATION

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".

EXEMPLE:

```
10        DIM  I  AS  BYTE
20        I  =  IN(0)                    ‘ Récupère le niveau logique de l’entrée "I/O 0" dans la variable “I”.
```

INFORMATIONS ADDITIONNELLES

Chacune des broches d’"E/S" 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. Dans ces conditions, il conviendra d’être extrêmement vigilant avec le type de signaux appliqués sur ces broches et le type de dispositifs pilotés par ces broches. Ceci est d’autant plus vrai lors des premières phases d’utilisation ou pour le besoin de vos tests, pendant lesquels vous serez amené à changer souvent le rôle de vos “broches”.

Dans le cas des ports de type « TTL », une tension supérieure à 1,4 Vcc sera considérée comme une valeur "1" (et comme une valeur "0" pour une tension inférieure à 1,3 Vcc).

Dans le cas des ports de type « Trigger de Schmitt », une tension supérieure à 3,4 Vcc sera considérée comme une valeur "1" (et comme une valeur "0" pour une tension inférieure à 3,3 Vcc).

Avant d’appliquer une quelconque tension (+ 5V ou masse) sur une des broches du PICBASIC, vérifiez IMPERATIVEMENT que cette broche ai bien été configurée en ENTREE. Dès lors, ne reliez aucune tension (+ 5V ou masse) sur les ports du PICBASIC configurés en sorties (sous peine de court-circuit et de destruction de ces derniers).

Si certaines broches du PICBASIC ne sont pas utilisées pour les besoins de votre application, configurez tout de même impérativement ces dernières en SORTIE et placez ces dernières au niveau logique « 0 ». Remettez à jour l’état de toutes les broches des PICBASIC régulièrement (même celles non utilisées) au sein de la « boucle » principale de votre programme (ne vous contentez pas d’une simple configuration au début du programme).

REFERENCE

Tous les ports passent à l’état HAUT (haute impédance) à la mise sous tension du PICBASIC.

NOTE

Dans tous les cas, il est impératif que les fils de connexions des signaux appliqués sur les entrées du PICBASIC ne dépassent pas quelques 3 cm. Il faudra également impérativement vérifier que la tension ne dépasse pas +5 Vcc sur les entrées du PICBASIC afin d’éviter tout dysfonctionnement et/ou destruction du PICBASIC (non pris en compte par la garantie). Suivant l’environnement dans lequel est utilisé le PICBASIC des circuits d’anti-parasitage devront être utilisés pour éviter toute perturbation sur les entrées du PICBASIC pouvant entraîner des disfonctionnements ou une destruction de ce dernier.

KEYIN ()

Variable Integer = KEYIN (*port*, [*Param*])

Gestion de touches

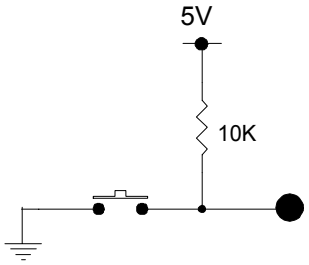
Port est une constante (0~31)
Param est une constante exprimant un delais en mS (0~255). La valeur par defaut est de 20 mS.

EXPLICATION

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, donné ci-dessous peut en effet être la source de rebonds pouvant “fausser”’ le niveau logique de lecture du programme. Pour y remédier, le paramètre (**Param**) va fixer une durée comprise entre 1 et 100 ms pendant laquelle le "PICBASIC" va attendre un niveau logique stable avant de le valider.

EXEMPLE:

```
10        DIM  I  AS  BYTE
20        I  =  KEYIN(0,25)
```



(Figure #1) Entrée test pour BP

(Figure 2#2) Génération de rebond par le poussoir

NOTE

Le schéma ci-dessus ne vaut que pour des essais. Il est dans tous les cas impératif que les fils de connexions des signaux du bouton-poussoir appliqués sur les entrées du PICBASIC ne dépassent pas quelques cm. Il faudra également impérativement vérifier que la tension ne dépasse pas +5 Vcc sur les entrées du PICBASIC afin d’éviter tout dysfonctionnement et/ou destruction du PICBASIC (non pris en compte par la garantie). Suivant l’environnement dans lequel est utilisé le PICBASIC des circuits d’anti-parasitage devront être utilisés pour éviter toute perturbation sur les entrées du PICBASIC pouvant entraîner des disfonctionnements ou une destruction de ce dernier.

KEYDELAY

Variable Integer = KEYDELAY (*Instruction*, *Param1*, *Param2*, *Param3*)

Gestion de touche

Instruction peut être ADKEY() ou KEYIN() ou PADIN().
Param1 est une constante donnantle retour de l'information (normallement 0) lorsqu'il n'y a pas de saisie.
Param2 est une constante (0~255) correspondant à une valeur d'attente.
Param3 est une constante (0~255) correspondant à une valeur de répétition.

EXPLICATION

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 ADKEY, 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 remplacée par ADKEY, KEYIN ou PADIN. Cette instruction n'est pas disponible sur les PBM-R1 / PBM-R5.

EXEMPLE:

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

            ' Action suite à l'action sur une touche

100       GOTO 20
```

LCDINIT

LCDINIT

Initialisation d'un LCD à commandes séries de type « ELCDxxx »

EXPLICATION

Cette instruction doit impérativement être exécutée au démarrage du programme si vous désirez piloter un afficheur LCD à commande série de type « ELCDxxx » à l'aide du port "PICBUS" du "PICBASIC" afin que l'afficheur s'initialise correctement

LOCATE

LOCATE *Val1*, *Val2*

Positionnement sur afficheur LCD à commandes séries

Val1 est une constante (0~255) ou une variable de type Byte.
Val2 est une constante (0~255) ou une variable de type Byte.

EXPLICATION

Cette instruction permet, si vous pilotez un afficheur LCD à commande série de type « ELCDxxx » 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.

La position (0,0) correspond au point le plus en haut à gauche. Dans le cas d'un afficheur de type 2 lignes de 16 caractères, vous pourrez utilisez les valeurs 0 à 15 pour **val1** et 0 à 1 pour **val2**.

EXEMPLE

```
LOCATE 10,0
PRINT "ABC"
```

ON...GOTO

ON *Var* GOTO *ligne1*, *ligne2*, *ligne3*, ...

Branchement conditionnel

Var est une variable de type Byte (La valeur maximale est 127)
Ligne1, Ligne2, Ligne3... sont des étiquettes ou des N° de lignes où leprogramme doit aller poursuivre son exécution.

EXPLICATION
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

                ' Quand I=0,  on continue l'exécution du programme à la ligne 100.
                ' Quand I=1, on continue l'exécution du programme à la ligne 200.
                ' Quand I=2, on continue l'exécution du programme à la ligne 300.
                ' Quand I ne correspond à aucune de ces valeurs, on continue l'exécution du programme à la ligne suivante.
```

ON INT ()=x GOSUB

ON INT (*port*) = *val* GOSUB *ligne*

Interruption sur Port

Port est une variable de type Byte ou un numéro de port (0~31) capable de recevoir une interruption.
Val est une constante (0 ou 1).
Ligne est une ligne ou étiquette correspondant à la sous-routine devant être appelée.

EXPLICATION

Lorsqu'un niveau logique Haut ou Bas est détecté sur le **port** d'un PICBASIC (autre que celui de la série « PBM »), cette instruction appelle la sous-routine spécifiée par ***Ligne***. Si ***Val*** est à 0, un niveau Bas sera détecté. A l'inverse, si ***Val*** est à 1, un niveau HAUT sera détecté. Cette instruction doit être utilisée une seule fois au début de votre programme (il ne faut pas l'utiliser plusieurs fois au sein de votre programme).

EXEMPLE

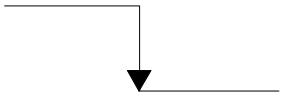
```
ON INT(0) = 0 GOSUB 20     ' Pendant l'éxécution du programme, si port 0 est au niveau bas le programme ira en ligne 10.
10 GOTO 10

20  OUT 1,1
    RETURN
```

A PROPOS DES INTERRUPTIONS SUR FRONTS MONTANT/DESCENDANT...

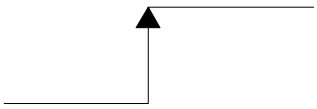
Selon le même principe il est également possible de générer une interruption sur un front (montant ou descendant) en utilisant un port spécial du PICBASIC. Pour les PICBASIC 1B/1S/2S/2H, il s'agira du port 5. Pour les PICBASIC-3B, il faudra utiliser le port 16 et le port 24 pour les PICBASIC-3H. Il suffira d'indiquer le N° du port entre parenthèse et le type de front à détecter (voir syntaxe ci-dessous).

EXEMPLE

```
ON INT(5) = 0 GOSUB 10    ' Si un front descendant survient sur P5  le programme ira en ligne 10.
```



```
ON INT(5) = 1 GOSUB 10     'Si un front montant survient sur P5  le programme ira en ligne 10.
```



ON TIMER () GOSUB

ON TIMER (*Param*) GOSUB *Ligne*

Time interrupt

Param est une constante (0~20) qui indique l'intervalle des interruptions.
Ligne est un N° ou une étiquette qui sera appelée lors de la détection de l'interruption.

EXPLICATION

Cette puissante instruction 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**. L'intervalle de temps défini par la valeur (**Param**) se fait grâce à une table de correspondance (données ci-dessous) en fonction du type de module utilisé.

EXEMPLE:

```
10        DIM  I  AS  BYTE
20        ON  TIMER(1)  GOSUB 100

' Programme principal.

100       I=I+1
110       RETURN
```

Interval	1B/1S/2S	2H/3B/ 3H	PBM-R1/R5
0	0.5 second	0.105 second	0.10 second
1	1 second	0.210 second	0.21 second
2	2 second	0.420 second	0.42 second
3	3 second	0.630 second	0.63 second
4	4 second	0.840 second	0.84 second
5	5 second	1.05 second	1.05 second
6	6 second	1.26 second	1.26 second
7	7 second	1.47 second	1.47 second
8	8 second	1.68 second	1.68 second
9	9 second	1.89 second	1.89 second
10	10 second	2.10 second	2.10 second

INFORMATION COMPLEMENTAIRE

Il est impératif que la durée d'exécution totale de la sous-routine appelée soit plus courte que la durée définie par **Param** dans l'instruction ON TIMER, sans quoi le programme de la sous-routine n'aura cesse d'être appelé en continu ou pourra également générer des disfonctionnements dans le déroulement du programme. De même, évitez de faire appel à des instructions telles que SEROUT, SERIN, PRINT dans la sous-routine appelée par "ON TIMER".

OUT

OUT *port*, *Val*

Sortie sur un port

Port est une constante (0~31) ou une variable de type Byte qui représente un N° de port.
Val est une constante (0 ou 1) ou une variable de type Byte.

EXPLICATION

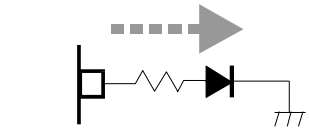
Cette instruction permet de faire passer une broche “I/O” (Port) dès lors configurée en sortie à un niveau logique donné (**Val**).

EXEMPLE:

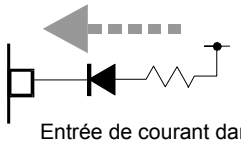
```
10      OUT 0,1     ‘ Place la broche “I/O 0” au niveau logique ‘1’ (+ 5 V)
20      OUT 0,0     ‘ Place la broche “I/O 0” au niveau logique ‘0’ (0 V)
```

ETAT DU PORT EN SORTIE...

Lorsqu’il est utilisé en sortie, un port de PICBASIC peut générer ou absorber jusqu’à 20 mA env. Ce qui est suffisant pour allumer directement une Led.

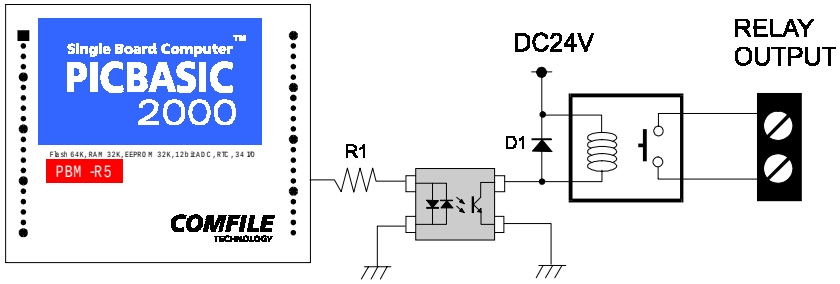


Sortie de courant d’un Port



Entrée de courant dans un port

Toutefois les 20 mA ne seront pas suffisant pour piloter un relais. Dans ce cas vous devrez utiliser un montage à transistor ou à optocoupleur comme ci-dessous. Dans tous les cas, le fil de sortie du PICBASIC ne devra pas depasser quelques cm (il ne faudra pas par exemple piloter des Leds déportées sur des grands fils).



OUTSTAT ()

OUTSTAT (*port*)

Verification de l'état d'un port

Port est une constante (0~31) ou une variable de type Byte qui représente un N° de port.

EXPLICATION

Cette instruction permet de connaître l'état logique d'une broche "I/O" (**Port**) configurée en sortie. Ceci peut être très utile, si on utilise une broche du "PICBASIC" tout en ayant la possibilité de connaître son état.

EXEMPLE:

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

INFORMATION COMPLEMENTAIRE

Rappel des actions réalisées par les différentes instructions sur les ports :

-IN() : Configure un port en entrée et récupère le niveau logique de ce port
-OUT() : Configure un port en sortie et change son état logique.
-OUTSTAT() : Configure un port en sortie et lecture du contenu interne du buffer de sortie

PADIN ()

PADIN (*block*)

Gestion de clavier

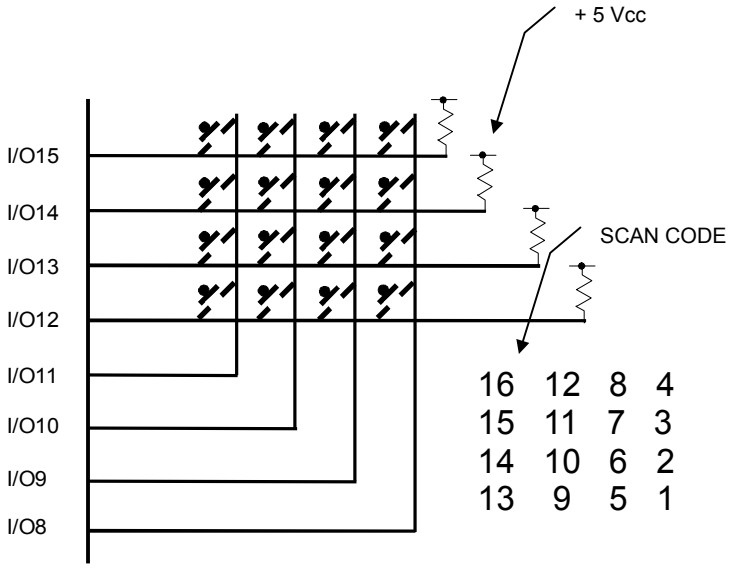
Bloc est une constante (0~3) de type Byte représentant le n° du block à utiliser.

EXPLICATION

Cette instruction permet de gérer automatiquement un clavier 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)
```



Dans les cas des PICBASIC « PBM-R1/R5 », l’instruction PADIN pourra être utilisée avec les blocs 1, 2 et 3. Pour tous les autres PICBASIC, il faudra utiliser le block 1.

Les valeurs des résistances de tirage (devant toutes être appliquées au + 5 Vcc) devront être comprises entre 5 et 10 Kohms. La longueur des fils reliant le clavier au PICBASIC ne devra pas depasser quelques cm afin que ce dernier ne soit pas perturbé par des parasites pouvant se propager le long des fils de liaison.

PEEK ()

PEEK (*Adr*)

Lecture de registres internes

Adr est une constante représentant une adresse de registre (0~&HFF)

EXPLICATION

Cette instruction permet de lire directement dans les registres internes du microcontrôleur PIC qui assure la gestion du PICBASIC (PIC16C73 ou PIC16C74 ou PIC16F876 ou PIC16F877 suivant les modèles). La valeur de l’octet se trouvant à l’adresse (**Adr**) peut ainsi être connue.

EXEMPLE

```
10        DIM  I  AS  BYTE
20        I  =  PEEK(&H85)
```

POKE

POKE *Adr*, *Val*

Ecriture dans un registre

Adr est une constante représentant une adresse de registre (0~&HFF)
Val est une constante ou une variable de type Byte (90~255).

EXPLICATION

Cette instruction permet “d’écrire” la valeur (**Val**) directement à l’adresse mémoire (**Adr**) du microcontrôleur PIC qui assure la gestion du PICBASIC (PIC16C73 ou PIC16C74 ou PIC16F876 ou PIC16F877 suivant les modèles).

EXEMPLE

```
10      POKE &H5,0          Force la donnée présente à l’adresse &H5 à zéro.
```

NOTE

Cette instruction peut avoir des effets directs sur le fonctionnement du PICBASIC. Nous recommandons aux personnes qui ne sont pas familiarisées avec les microcontrôleurs PIC de ne pas utiliser cette instruction sans quoi des disfonctionnements pourraient intervenir sur votre application.

PLAY

PLAY *port*, *Val1 Val2 Val3 ...*

Génération de musique

Port est une constante (0~31) ou une variable de type Byte indiquant le N° d’un Port.

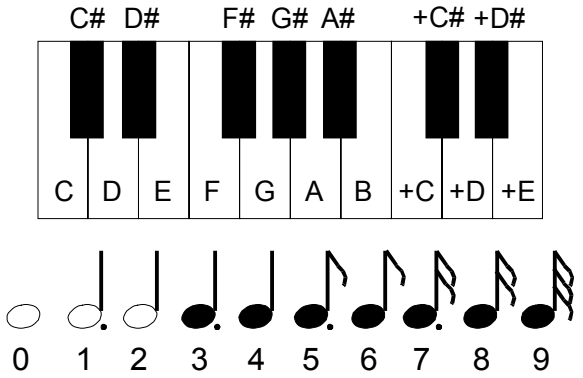
EXPLICATION

Cette instruction permet la génération de notes musicales sur une broche du module "PICBASIC" (qui devra être connectée à un buzzer sans oscillateur). Le paramètre "**Port**" indique la broche où sera connectée le buzzer. Les paramètres "Val1 Val2 Val3..." indiquent la valeur des notes conformément au "clavier" ci-contre (la lettre indique la note et le chiffre, la durée de la note). Sur les PICBASIC de la série « PBM » (PICBASIC-R1 / PICBASIC-R5), vous ne pourrez utiliser que les ports 0 à 15 pour restituer les notes de musiques sur le buzzer. Le buzzer devra être câblé au plus près de la broche du PICBASIC (quelques cm de fils max.).

EXEMPLE

```
10      PLAY 5,”C5C7D4C4F4E2C5C7D4C4G4F2C5”
20      PLAY 5,”C7+C4A4F4E4D2A#5A#7G4E4G4F4”
```

Génère la musique "Joyeux anniversaire...".



LORSQUE VOUS INDIQUEZ C4 :

2 paramètres sont utilisés pour définir la tonalité. Le premier paramètre est une lettre de l’alphabet (A ~ G) qui détermine l’interval de la tonalité et l’autre paramètre est un chiffre qui détermine la durée de la tonalité. Ainsi, “C4” désigne un Do 1/4 de note.

LORSQUE VOUS INDIQUEZ +C#5,

4 paramètres sont utilisés pour définir la tonalité. “#” désigne la tonalité haute par demi-ton et “+” désigne la tonalité haute par un octave. Ainsi+C#5 désigne une tonalité plus haute qu’un Do (d’un demi-ton et d’un octave). Sa longeur est 5. (1/4 de note.)

Exemple : C5E5G5+C1
Joue un : do, mi, sol, do.

PRINT

PRINT *Texte* [*Val1*, *Val2*, ...]

Affichage sur écran LCD

Texte est une chaîne de caractères de type “ABC123”
Val1, Val2 est une constante (0~255) ou une variable de type Byte

EXPLICATION

Cette instruction permet d’afficher des messages sur un afficheur « OEM » LCD Comfile Technology de la série « Elcdxxx » (à commandes séries) 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").

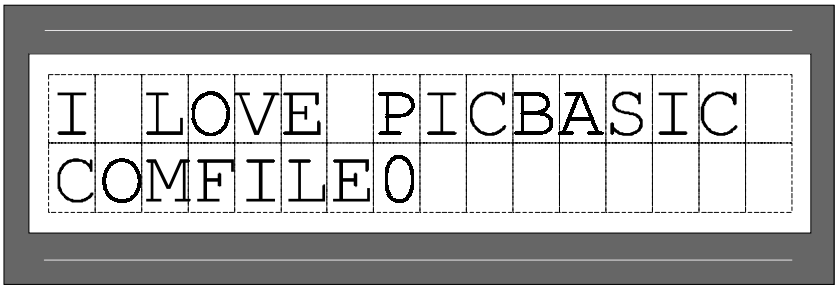
EXEMPLE1

```
10  SET PICBUS HIGH                    10  SET PICBUS HIGH                    10  SET PICBUS HIGH
20  LCDINIT                            20  LCDINIT                            20  LCDINIT
30  LOCATE 0,0          OU             30  LOCATE 0,0          OU            30  LOCATE 0,0
40  PRINT “LEXTRONIC-2001"             40  PRINT “LEXTRONIC-"                 40  PRINT “LEXTRONIC-200",&H31
                                       50  PRINT “2001”.
```

Il est également possible d’afficher des valeurs décimales ou héxadécimales en utilisant les instructions de conversion telles que DEC ou HEX.

EXEMPLE2

```
SET PICBUS HIGH
LCDINIT
LOCATE 0,0
PRINT “I LOVE PICBASIC”
PRINT “COMFILE”,&H30
```



EXEMPLE sur un afficheur « ELCD162 »

PRINT DEC

PRINT DEC (*Var, Param1, Param2*)

Affichage sur écran LCD

Var est une constante ou une variable de type Byte
Param1 est une constante (1~5) de type Byte
Param2 est une constante (0 ou 1) de type Byte

EXPLICATION
Cette instruction permet d'afficher la valeur décimale d'un nombre selon plusieurs formats possibles sur un afficheur « OEM » LCD Comfile Technology de la série « Elcdxxx » (à commandes séries) 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 (Param1) détermine le nombre de caractères que devra occuper le nombre à l'écran (1 à 5) - Si (Param1) est inférieur au nombre de chiffres qui compose le nombre à afficher, ce dernier sera alors affiché en partie. Le paramètre (Param2) permet de déterminer si le nombre doit être précédé de "0" à la place des caractères non utilisés (Param2 = 0 -> affichage de "0", Param2 = 1 -> Affichage d'espace). En absence de paramètre (Param1 et Param2), le "PICBASIC" affichera automatiquement le nombre sur 5 caractères, sans "0" devant.

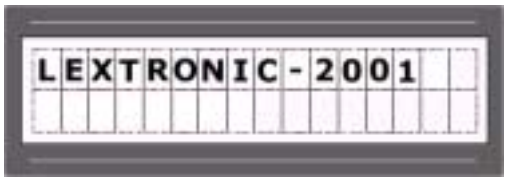
EXEMPLE 1:

```
10        SET PICBUS HIGH
20        DIM   I  AS   BYTE
30        I  =  12
40        LCDINIT
50        LOCATE 0,0
60        PRINT DEC(21,5,0)
70        LOCATE 0,1
80        PRINT DEC(I,4,1)              'Ce programme affiche le message de l'afficheur ci-dessus:
```

Il est également possible de réaliser des combinaisons afin d'afficher plusieurs types de données/textes sur une même ligne.

EXEMPLE 2:

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



PRINT HEX

PRINT DEC (*Var, Param1, Param2*)

Affichage sur écran LCD

Var est une constante ou une variable de type Byte
Param1 est une constante (1~5) de type Byte
Param2 est une constante (0 ou 1) de type Byte

EXPLICATION
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 « OEM » LCD Comfile Technology de la série « Elcdxxx » (à commandes séries) préalablement connecté sur le port "PICBUS" du "PICBASIC".

EXEMPLE:

```
10        DIM  I  AS   BYTE
20          SET PICBUS HIGH
30        I  =  10
40        LCDINIT
50        LOCATE 0,0
60        PRINT HEX(I,4,0)
```

PULSE

PULSE *port*

Génération d'impulsion

Port est une constante (0~31) ou une variable de type Byte indiquant un N° de port.

EXPLICATION
Cette instruction permet de générer des impulsions (positives ou négatives) de durée fixe de l'ordre de 2 à 3 µs 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.

EXEMPLE 1:

```
10        OUT 2,0          ' Place le port au niveau logique bas
20        PULSE 3          ' Génère une impulsions positive de l'ordre de 2 à 3 µs
```

EXEMPLE 2:

```
10        OUT 2,1          ' Place le port au niveau logique haut
20        PULSE 3          ' Génère une impulsions négative de l'ordre de 2 à 3 µs
```

PWM

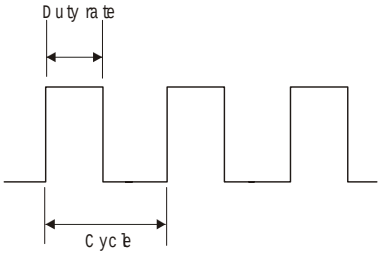
PWM *port*, *Val1*, *Val2*

Génération de signaux PWM

Port est une constante (9 ou 10) ou une variable de type Byte.
Val1 est une constante ou une variable de type Byte (pour les « PBM » elle est de type Integer)
Val 2 est une constante/Variable de type Byte (uniquement utilisable sur la série « PBM »).

EXPLICATION

Cette instruction permet de générer sur une des broches du "PICBASIC" (Port), un signal rectangulaire de fréquence fixe (voir tableau ci-dessous) mais dont le rapport cyclique est variable et fonction de **Val1** (duty rate). **Val 1** est ajustable de 1 à 255 (ou de 1 à 1023 sur la série « PBM »). 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".



Le paramètre **Val2** (cycle) peut être ajusté uniquement que sur la série des PICBASIC « PBM » - Si cette valeur est omise, elle sera d’office à 255. Le tableau ci-dessous donne la relation entre **Val2** (cycle) et la génération du signal PWM. La valeur du rapport cyclique variera onc également en fonction de la valeur de **Val2** (*cycle)*. La valeur de **Val1** (duty rate) ne doit pas dépasser le quadruple de la valeur de **Val2** (cycle). En cas contraire, le port passera à l’état haut. Gardez également à l’esprit que le paramètre **Val2** (cycle) affecte les 2 ports PWM (9 et 10) en même temps. Il n’est pas possible d’utiliser des valeurs différentes pour les 2 ports. Modifiez donc la valeur de Val2 avec précaution.

Valeur cycle	Fréquence	Gamme Duty rate
10	28.4KHz	0~40
100	3KHz	0~400
255	1.22KHz	0~1023

Dans le cas des autres PICBASIC (série PB), vous ne pouvez pas changer la valeur de **Val2** (cycle). Celle-ci est figée à la valeur 255. La fréquence de chaque modèle est indiquée dans le tableau ci-dessous.

1B/1S/2S	2H	3B/3H
Fréquence 256Hz Résolution 8 bits	Fréquence 1.22 KHz Résolution 8 bits	Fréquence 19.53 KHz Résolution 8 bits

Si vous désirez générer un signal de fréquence différente sur la série des PICBASIC de type « PB », vous devrez utiliser l’instruction FREQOUT. En revanche, il ne sera pas possible de générer un signal "PWM" sur une sortie et un signal avec l'instruction "FREQOUT" sur une autre sortie en même temps.

EXEMPLE

```
10   I = ADIN(0)     ‘ Mémorise la valeur de la tension présente sur l’entrée du PICBASIC
     PWM 9, I        ‘ Génère un signal PWM variable surle port 9
GOTO 10
```

PWMOFF

PWMOFF port

Stoppe la génération d'un signal PWM

Port est une constante (9 ou 10) ou une variable de type Byte.

EXPLICATION

Cette instruction permet de désactiver le signal "PWM" (initialement généré par l'instruction "PWM"). A la mise sous tension les ports 9 et 10 ne génèrent aucun signal PWM.

EXEMPLE

```
10        PWM  9,191
20        DELAY      255
30        PWMOFF   9                    ' Stoppe le signal PWM
```

RND (0)

RND (*0*)

Génération valeur pseudo-aléatoire

EXPLICATION

Cette instruction permet de générer une suite de chiffres "peusdo" aléatoires (la séquence est la même à chaque mise sous tension du PICBASIC).

EXEMPLE

```
10        DIM I      AS BYTE
20        I=RND(0)
30          SOUND 2,I,3                 ' Joue une série de notes pseudo-aléatoires
40          GOTO 20
```

SERIN

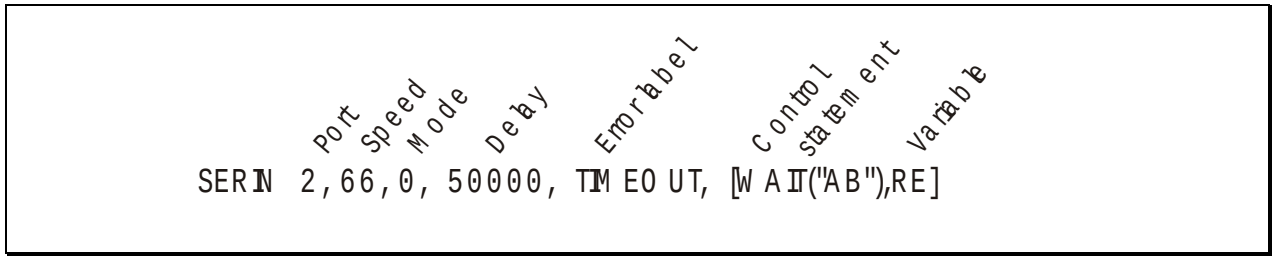
SERIN *port*, *Param1*, *Mode*, *Param2*, *Adress* [*Condition*, *var1*]

Réception de données séries

Port est une constante de type Byte représentant un N° de Port (sur la série PBM on ne peut utiliser que les ports 0 ~ 15)
Param1 est une constante qui détermine la vitesse de communication de la liaison série.
Mode est une constante (0 ou 1) permettant d'inverser la polarité des données reçues.
Param2 est une constante représentant une durée à partir de laquelle si aucune donnée n'est reçue, le PICBASIC continuera l'exéution de son programme à la ligne **Adress**. Si **Adress** est absent, le programme continuera à la ligne suivante.
Adress est un N° de ligne ou une étiquette utilisée si aucune donnée n'est reçue dans un laps de temps donné.
Condition permet d'inclure des conditions lors de la réception des données telles que WAIT, UNTIL et SKIP. UNTIL et SKIP ne sont pas disponibles avec les PICBASIC de la série « PBM ».
Var1 est une variable de type Byte servant à recevoir les données.

EXPLICATION

Cette instruction permet d'attendre la réception de données sous forme série (8 bits, 1 stop, sans parité). 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-après. Durant cette phase, le module "PICBASIC" ne pourra pas effectuer d'autres tâches 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 "Adress". Le paramètre (**Mode**) permet d'inverser la polarité des données reçues. En temps normal, il doit être mis à "0".



La vitesse de communication définie par Param1 est différente pour chaque modèle de PICBASIC (voir table de correspondance ci-dessous). Cette table peut aussi être utilisée pour l'instruction SEROUT.

Baud rate	Valeur Param1 (1B/1S/2S)	Valeur Param1 (2H/3B/3H)	Valeur Param1 (PBM-R1/R5)
300			3260
600			1620
1200			810
2400	138		400
4800	66	207	196
9600	30	103	93
19200	11	47	40
38400			14

EXEMPLE

```
SERIN  2, 196, 0, 5000, TIMEOUT, [I]      ' Attend réception de données sur le port 2 à 4800 bds avec un « PBM-R1 ».
                                          ' Saute à l'étiquette TIMEOUT si aucune donnée n'arrive dans les 5 secondes.
```

Rappelez-vous enfin que les niveaux logiques présents sur les ports des PICBASIC sont de 0 / 5 Vcc. Si vous devez raccorder le PICBASIC à un PC ou à tout autre dispositif au travers d'une liaison RS232 « standard », il vous faudra intercaler un composant MAX-232 additionnel (à câbler au plus près du PICBASIC) entre le port utilisé avec l'instruction SERIN et le port série du PC.

RECEPTION DE PLUSIEURS OCTETS.....
Vous pouvez utiliser un tableau ou une chaîne pour recevoir plusieurs données en utilisant le caractère “~”.

```
SERIN 2, 93, 0, 50000, 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).

```
SERIN  2, 93, 0, 50000, TIMEOUT, [ST~5]
```

Dans cet exemple le "PICBASIC" va attendre 5 données et transférer celles-ci dans la variable de type chaîne ST – Le dernier octet reçu doit être un « 0 » (ce qui signifie pour le PICBASIC une fin de chaîne). Vérifiez que la taille de la chaîne déclarée par l’instruction DIM en début de programme ne soit pas dépassé.

INFORMATIONS COMPLEMENTAIRES

Réception avec condition #1: WAIT
La condition « WAIT » pemet d’inclure une notion de « filtrage » ou « d’adressage » lors de la réception de vos données. En utilisant cette dernière, le programme attendra 2 octets particuliers, puis mémorisera la suite des données au sein d’une variable.

```
SERIN 2, 66, 0, 50000, TIMEOUT, [WAIT(“AB”),I]
```

Cette instruction attend pendant 5 secondes la suite de caractères "AB" sous forme de données série à 4800 bds (avec un PICBASIC-1B) 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".

Réception avec condition#2: UNTIL (non utilisable avec les PICBASIC de la série « PBM »)
La condition « UNTIL » permet lors de la réception d’une suite de données de stopper la réception si un caractère spécifique est reconnu.

```
SERIN 2, 66, 0, 50000, TIMEOUT, [UNTIL(“*”),A(0)~10]
```

Cette instruction attend pendant 5 secondes la réception de données sous forme de données série à 4800 bds (avec un PICBASIC-1B) sur la broche "I/O 2". Si des données arrivent, elles seront stockées dans les variables A(0), A(1), A(2)… Si parmis les données reçues le programme détecte le caractère "*", la réception est stoppée.

Réception avec condition #3: SKIP (non utilisable avec les PICBASIC de la série « PBM »)
La condition « SKIP » permet lors de la réception d’une suite de données de ne pas prendre en compte un caractère spécifique.

```
SERIN  2, 66, 0, 50000, TIMEOUT, [SKIP(“R”),A(0)~10]   ‘During receiving 10 bytes to array A, character R will be skipped.
```

Cette instruction attend pendant 5 secondes la réception de données sous forme de données série à 4800 bds (avec un PICBASIC-1B) sur la broche "I/O 2". Si des données arrivent, elles seront stockées dans les variables A(0), A(1), A(2)… Si parmis les données reçues le programme détecte le caractère "R", ce dernier ne sera pas mémorisé.

ASTUCE POUR UTILISER UNE RECEPTION SUR INTERRUPTION AVEC LES PICBASIC DE LA SERIE « PB »
Le programme ci-dessous vous permettra de quitter automatiquement le « cours » de votre programme principal dès qu’une données sera reçue sur la broche d’interruption d’un PICBASIC de la série PB (PICBASIC-1B par exemple).

```
ON INT(5)=0 GOSUB RS232_INT
:
:  ‘ Programme principal
:

RS232_INT:
SERIN 5,11,0,1000,TIMEOUT,[BF]
TIMEOUT:
RETURN
```

Lors de la détection d’un front descendant sur le port 5 (Bit de start), le programme principal ira exécuter la sous-routine d’interruption **RS232_INT** , laquelle attendra la réception des données séries. A ce moment il faut que le système annexe qui envoi des données au PICBASIC puisse temporairement retarder l’envoi des données car le PICBASIC mettra plusieurs mS entre le moment ou il détecte le bit de start et où il est prêt à recevoir des données. Vous pouvez également essayer d’envoyer un premier octet « non exploitable » à 00 afin que le PICBASIC ait le temps de se synchroniser (mais il faut faire des essais en fonction du type de système qui envoi des données au PICBASIC).

SEROUT

SEROUT *port*, *Param1*, *Mode*, *Interval*, [*Var1*]

RS232C transmission

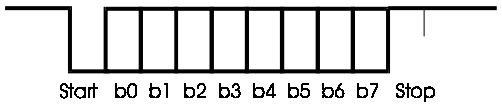
Port est une constante de type Byte représentant un N° de Port (sur la série PBM on ne peut utiliser que les ports 0 ~ 15)
Param1 est une constante qui détermine la vitesse de communication de la liaison série.
Mode est une constante (0 ou 1) permettant d'inverser la polarité des données envoyées.
Interval est une constante permettant d'intercaler des temporisations (en mS) entre les données envoyées.
Var1 est une variable de type Byte servant à envoyer les données.

EXPLICATION

Cette instruction permet de transmettre des données sous forme série (8 bits, 1 stop, sans parité). Une fois exécutée, la broche (**Port**) du "PICBASIC" transmettra la ou les données (**Var1**) à une vitesse définie par (**Paramètre1**), selon la correspondance du tableau donné ci-après. Le paramètre (**Mode**) permet d'inverser la polarité des données envoyées. En temps normal, il doit être mis à "0". Le paramètre (**Interval**) permet d'instaurer une temporisation (en ms) entre chaque caractère émis (la valeur à mettre par défaut est « 1 »). Les données à envoyer (**Var1**) peuvent être de type "BYTE" ou chaîne. Si vous essayez d'envoyer des données de type "INTEGER", seuls les 8 bits de poids faibles seront transmis.

```
                Port  Speed  Mode  Interval  Data
SEROUT 3,66,0,1,[&HA0]
```

La figure ci-dessous montre la forme du signal série.



En résumé, SEROUT offre des possibilités similaires à l'instruction PUT sur les PICBASIC de la série « PBM » (mais sans gestion matérielle de la transmission – C'est à dire que durant la transmission de données avec SEROUT, le PICBASIC ne peut pas faire autre chose). Par contre l'avantage de SEROUT est de pouvoir travailler avec n'importe quel port, de pouvoir inverser les données et disposer de temporisation entre les données envoyées.

La vitesse de communication définie par **Param1** est différente pour chaque modèle de PICBASIC (voir table de correspondance ci-dessous). Cette table peut aussi être utilisée pour l'instruction SEROUT.

Baud rate	Valeur Param1 (1B/1S/2S)	Valeur Param1 (2H/3B/3H)	Valeur Param1 (PBM-R1/R5)
300			3260
600			1620
1200			810
2400	138		400
4800	66	207	196
9600	30	103	93
19200	11	47	40
38400			14

EXEMPLE

```
SEROUT 3, 196, 0, 1, [&HA0]
' Envoi en série l'octet &HA0 sur le port 3 (à 4800 bds via un PBM-R1, sans inversion de polarité et avec interval de 1 ms)

SEROUT 1, 93, 0, 1, ["PICBASIC 2000",13,10]
' Envoi en série une chaîne de caractères (PICBASIC ainsi que les octets 13 et 10 sur le port 3 (à 9600 bds via un PBM-R1,
  sans inversion de polarité et avec interval de 1 ms)
```

INFORMATIONS COMPLEMENTAIRES

Lorsque vous désirez envoyer seulement des codes ASCII, utilisez les instructions de conversion telles que DEC, HEX… pour convertir les digits en code ASCII.

```
SEROUT 1, 93, 0, 1, [DEC(I)]        ' Envoi la valeur décimale de la variable I
SEROUT 1, 93, 0, 1, [HEX(I)]        ' Envoi la valeur héxadécimale de la variable I
```

Lorsque vous envoyez un variable de type String, tout le contenu de la variable est envoyé.

```
DIM ST AS STRING * 16
ST = "PICBASIC 2000"
SEROUT 1, 93, 0, 1, [ST]    '" PICBASIC 2000" est envoyé.
```

Rappelez-vous enfin que les niveaux logiques présents sur les ports des PICBASIC sont de 0 / 5 Vcc. Si vous devez raccorder le PICBASIC à un PC ou à tout autre dispositif au travers d'une liaison RS232 « standard », il vous faudra intercaler un composant MAX-232 additionnel (à câbler au plus près du PICBASIC) entre le port utilisé avec l'instruction SEROUT et le port série du PC.

SERVO

SERVO *Port*, *Val*

Gestion d'un servomoteur

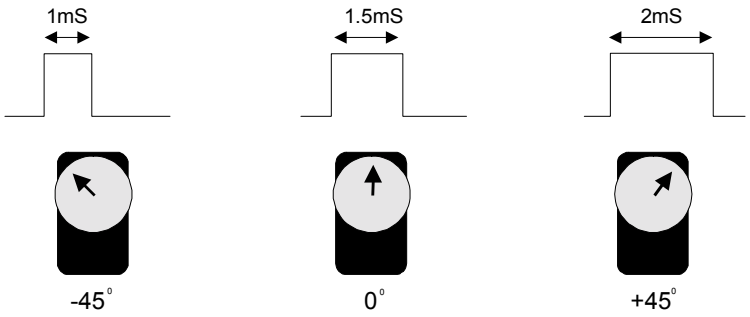
Port est une constante de type Byte représentant un N° de Port
Val est une constante ou une variable de type Byte comprise entre 0~255.

EXPLICATION

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**) et fonction du type de "PICBASIC" utilisé. A noter que les impulsions générées devront être espacées de 10 à 15 ms chacune.

Pour une impulsion de l'odre de 1 mS le servomoteur se positionne sur –45 °.
Pour une impulsion de l'odre de 1,5 mS le servomoteur se positionne sur 0 °.
Pour une impulsion de l'odre de 2 mS le servomoteur se positionne sur +45 °.

NOTA : De part le manque de précision des servomoteurs, les valeurs indiquées peuvent différer d'un modèle à l'autre.



Il n'est pas possible d'utiliser l'instruction SERVO avec les PICBASIC de la série « PBM ». Pour piloter un servomoteur avec les PICBASIC de la série « PBM » vous devrez utiliser l'instruction PULSE.

EXEMPLE

Exemple de positionnement du servomoteur sur position –45° avec un PB-1B/1S/2S via le Port 0.

```
DASI:   SERVO 0, 333          ' Generation impulsion 1mS sur Port 0
        DELAY 10
        GOTO DASI
```

Tableau permettant l'utilisation de l'instruction SERVO en fonction des PICBASIC.

	PB-1B / 1S / 2S	PB-2H, 3B, 3H
Unité	3 uS	0.8 uS
Génération pulse de 1mS	SERVO 0, 333	SERVO 0, 1250

INFORMATIONS COMPLEMENTAIRES

Il est impératif que la génération des impulsions soit continue et sans interruption (avec un espacement de durée fixe). Dans certains cas, il pourra être difficile de réaliser cet impératif. C'est pourquoi, vous pouvez avoir recours à un module optionnel "SMC" (voir page 4) qui vous permettra de piloter 8 servomoteurs via une commande série. 8 cartes "SMC" peuvent ainsi être pilotées afin de pouvoir gérer jusqu'à 64 servomoteurs !

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 produits lors de la rotation du moteur. De plus le fil de liaison reliant le port du PICBASIC à l'entrée de commande du servomoteur devra être le plus court possible.

SET PICBUS

SET PICBUS *HIGH/LOW*

Détermine la vitesse de communication du port PICBUS des PICBASIC

EXPLICATION

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

```
10      SET PICBUS HIGH          ‘ Configure le “PICBUS” à 19200 bps
20      SET PICBUS LOW           ‘ Configure le “PICBUS” à 4800 bps
```

SHIFTIN ()

SHIFTIN (*Port 1*, *Port2*, *Param*, *Bit*)

Communication bi-filaire

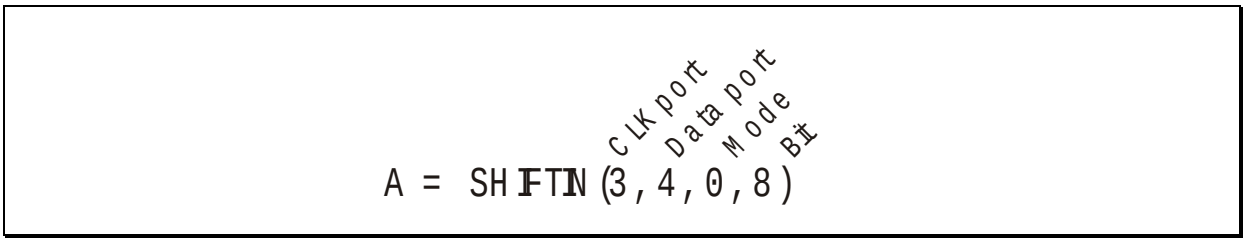
Port1 est une constante de type Byte indiquant le N° d’un Port (assurant la génération du signal de sortie d’horloge)
Port2 est une constante de type Byte indiquant le N° d’un Port (assurant la génération du signal d’entrée des données)
Param

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 est une constante indiquant le nombre de bits 8 ~ 16 bit (par defaut 8 bits)

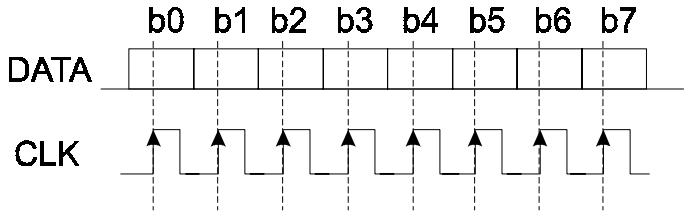
EXPLICATION

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 (**Param**) permet de définir le mode de lecture (voir syntaxe ci-dessus). Le paramètre (Bit) permet de définir le nombre de bits à lire (8 ou 16).



EXEMPLE

```
I = SHIFTIN(3,4,0)      ' Le port3 st utilisé en tant que signal d’horloge. Le Port4 reçoit les donées en entrée.
                        ' Le mode est 0 et le résultat est stocké dans I.
```



8SHIFTOUT

SHIFTOUT *Port1*, *Port2*, *Param*, *data*, *bit*

Serial output

Port1 est une constante de type Byte indiquant le N° d'un Port (assurant la génération du signal de sortie d'horloge)
Port2 est une constante de type Byte indiquant le N° d'un Port (assurant la génération du signal des données)
Param
0 = LSB prioritaire
1 = MSB prioritaire
2 = MSB prioritaire avec génération d'un signal 'ACK' (convient pour le pilotage de composant I2C™).
Bit est une constante indiquant le nombre de bits 8 ~ 16 bit (par defaut 8 bits)

EXPLICATION

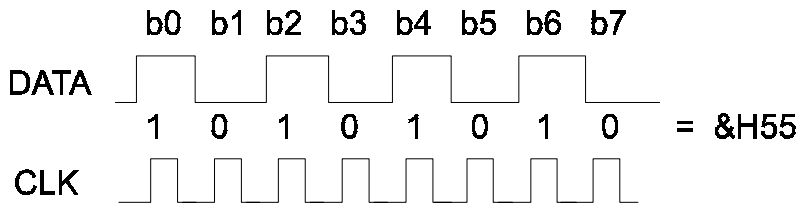
Cette instruction permet de "communiquer" très facilement avec la plupart des composants à adressage série 2 fils (type I2C™, SPI™...). Son exécution génère un signal d'horloge de synchronisation sur la sortie (**Port1**) du "PICBASIC", tout en venant "écrire sériellement" les données présentent sur l'entrée (**Port2**). Le paramètre (Param) 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).

```
                    Clk port
                       Data port
                          Mode
                             Data
                                Bit
SHIFTOUT 3,4,0,TE,8
```

Configurez **Param** avec la valeur 2 pour une communication de type I2C™ (avec ACK après chaque réception de 8 bits).

EXEMPLE

```
SHIFTOUT 0,1,0,&H55          ' Le port 0 est l'horloge, le port 1 est pour les données, Le mode 0 est sélectionné
```



SOUND

SOUND *port*, *Val1*, *Val2*, [, ...]

Generate sound

Port est une constante ou une variable de type Byte indiquant le N° d'un Port
Val1 est une constante/variable de type Byte.
Val2 est une constante/variable de type Byte.

EXPLICATION

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 (**Por**t) du "PICBASIC". Il est possible de cumuler plusieurs tonalités les unes à la suite des autres. Pour se faire, il vous faudra raccorder un buzzer sans oscillateur sur le port du PICBASIC (le fil de raccordement entre le buzzer et le port du PICBASIC devra être le plus court possible).

Lorsque les valeurs 233 ~ 139 sont utilisées, on obtient approximativement les notes do, re, mi, fa, sol, la (sur un octave)
Plus ***Val2*** est grand, plus la note sera générée longtemps (lorsque ***Val2*** est à 16, les notes correspondent environ à 1/4 note).

Sur les PICBASIC de la série « PBM », il vous faut utiliser les ports 0~15 pour la génération des sons.

EXEMPLE

```
10      SOUND   1,239,10,159,10
20      GOTO  10                    ' Génère 2 sonorités de suite (type "pompier") en boucle
```

STEPOUT

STEPOUT *Port1*, *Var1*, *Var2*, *Port2*, *Var3*

Gestion d'un moteur pas-à-pas

Port1 est une constante ou une variable de type Byte indiquant le N° d'un Port
Var1 est une constante (1~255) ou une variable de type Byte définissant l'interval des impulsions
Var2 est une constante (1~65535) ou une variable de type Integer
Port est une constante ou une variable de type Byte indiquant le N° d'un Port (utilisé pour stopper le moteur pas-à-pas)
Var3 est une constante (0 ou 1)

EXPLICATION

Cette instruction (uniquement disponible sur les PICBASIC de la série « PB ») permet en association avec des interfaces spécialisées, de piloter très facilement des moteurs pas-à-pas. Ces interfaces (non livrées) 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-dessous et dont le nombre généré est directement fonction de (**Var2**).

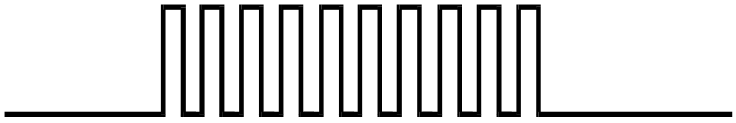
Val1 (1~255)	1B,/1S/ 2S	2H/3B/ 3H
1	11.6 KHz	60.9 KHz
2	10.3 KHz	53.2 KHz
5	7.6 KHz	38.47 KHz
10	5.3 KHz	26.31 KHz
20	3.3 KHz	16.13 KHz
50	1.54 KHz	7.463 KHz
100	0.819 KHz	3.937 KHz
200	0.423 KHz	2.024 KHz
255	0.333 KHz	1.597 KHz

EXEMPLE 1:

```
10        STEPOUT  2, 50, 10
```

Cette ligne va générer 10 impulsions de fréquence 1,54 KHz si vous l'utilisez sur un "PICBASIC-1B".
Si le moteur pas-à-pas est un modèle de type 1,8 °, il effectuera une rotation de 18 °.

```
10        STEPOUT  2, 20, 10          ' Cette instruction générera une fréquence plus élevée.
                                      ' Le moteur tournera plus vite.
```



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 1,54 KHz et stoppe ces dernières si le niveau logique présent sur "I/O 3" tombe à "0".

TABLE ()

Table (*Var*, *Val1*, *Val2*, *Val3*, ...)

Table de correspondance

Var est une variable de type Byte. Sa valeur ne doit pas dépasser 127.
Val1, Val2, Val3... sont des constantes (0~255)

EXPLICATION

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... Si I a une valeur supérieur à 4 alors J = 0.

TOGGLE

TOGGLE *port*

Inverse le niveau logique d'une sortie

Port est une constante comprise entre (0~31) et représentant un N° de port.

EXPLICATION

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.

EXEMPLE

```
10          DIM   I   AS   BYTE
20          OUT    0,1
30          FOR   I=0 TO 7
40              TOGGLE  0
50           NEXT       I
60           OUT        0,0
```

'Dans cet exemple, la broche "I/O 0" va changer plusieurs fois d'état avant de passer définitivement au niveau logique "0".