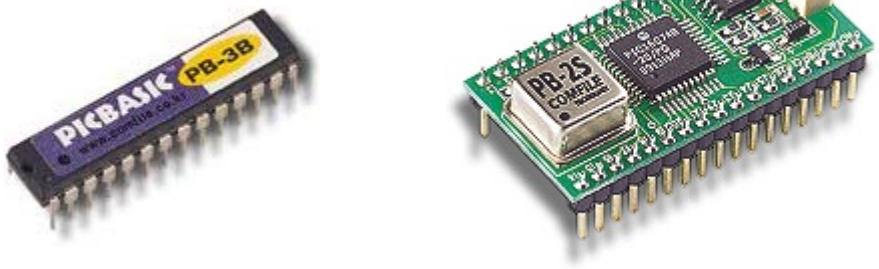
BASIC SINGLE BOARD COMPUTER

PICBASIC

Manuel utilisateur
Vol. 10



"Everything for Embedded Control"

COMFILE
TECHNOLOGY

Copyrights et appellations commerciales

Informations techniques

Ce manuel a été conçu avec la plus grande attention. Tous les efforts ont été mis en oeuvre pour éviter les anomalies. Toutefois, nous ne pouvons garantir que ce dernier soit à 100% exempt de toute erreur. Les informations présentes dans ce manuel sont données à titre indicatif. Les caractéristiques techniques des "PICBASIC", la nature, les possibilités et le nombre de leurs instructions, ainsi que les possibilités de leurs logiciels de programmation et les caractéristiques des modules périphériques associés aux PICBASIC peuvent changer à tout moment sans aucun préavis dans le but d'améliorer la qualité et les possibilités de ces derniers.

Limitation de responsabilité

En aucun cas le Fabricant et LEXTRONIC ne pourront être tenus responsables de dommages quels qu'ils soient (intégrant, mais sans limitation, les dommages pour perte de bénéfice commercial, interruption d'exploitation commerciale, perte d'informations et de données à caractère commercial ou de toute autre perte financière) provenant de l'utilisation ou de l'incapacité à pouvoir utiliser les modules "PICBASIC" et leurs logiciels associés ainsi que leurs platines et modules optionnels associés, même si le Fabricant ou LEXTRONIC ont été informés de la possibilité de tels dommages.

Les modules "PICBASIC" ainsi que leurs platines et modules optionnels associés sont destinés à être utilisés en milieu résidentiel dans les gammes de températures +10 à +50 °C. Les modules "PICBASIC" ainsi que leurs platines et modules optionnels associés ne sont pas conçus, ni destinés, ni autorisés pour être utilisés au sein d'applications militaires, ni au sein d'applications à caractère médical, ni au sein d'applications de détection incendie, ni au sein d'applications d'alarme anti-intrusion, ni au sein d'applications sur ascenseurs, ni au sein d'applications sur machines outils, ni au sein d'applications de commande de feux d'artifices, ni au sein d'applications embarquées dans des véhicules (automobiles, camions, bateaux, scooters, motos, scooters des mers, avions, hélicoptères, ULM, etc...), ni au sein d'applications embarquées sur des maquettes volantes de modèles réduits (type avions, hélicoptères, planeurs, etc...).

De même, les modules "PICBASIC" ainsi que leurs platines et modules optionnels associés ne sont pas conçus, ni destinés, ni autorisés pour expérimenter, développer ou être intégrés au sein d'applications dans lesquelles une défaillance de ces derniers pourrait créer une situation dangereuse pouvant entraîner des pertes financières, des dégâts matériels, des blessures corporelles ou la mort de personnes ou d'animaux. Si vous utilisez les modules "PICBASIC" ainsi que leurs platines et modules optionnels associés volontairement ou involontairement pour de telles applications non autorisées, vous vous engagez à soustraire le Fabricant et LEXTRONIC de toute responsabilité et de toute demande de dédommagement.

En cas de litige, l'entière responsabilité du Fabricant et de LEXTRONIC vis-à-vis de votre recours se limitera exclusivement selon le choix du Fabricant et de LEXTRONIC au remboursement du module "PICBASIC" et/ou de ses platines et modules optionnels associés et/ou de leur réparation et/ou de leur échange. Le Fabricant et LEXTRONIC démentent toutes autres garanties, exprimées ou implicites.

L'utilisateur des modules "PICBASIC" et de ses platines et modules optionnels associés est entièrement et seul responsable des développements logiciels (de l'écriture de son programme en langage BASIC) ainsi que de l'intégration matérielle, des modifications et ajouts de périphériques qu'il effectuera sur les modules "PICBASIC" ainsi que leurs platines et modules optionnels associés. S'agissant de matériel "OEM", Il incombera à l'utilisateur de vérifier que l'application finie complète développée avec les modules "PICBASIC" ainsi que leurs platines et modules optionnels associés soient conformes aux normes de sécurité et aux normes CEM en vigueur.

Tous les modules "PICBASIC" ainsi que leurs platines et modules optionnels associés sont testés avant expédition. Toute inversion de polarité, dépassement des valeurs limites des tensions d'alimentation, courts-circuits, utilisation en dehors des spécifications et limites indiquées dans ce document ou utilisation pour des applications non prévues pourront affecter la fiabilité, créer des dysfonctionnements et/ou endommager les modules "PICBASIC" ainsi que leurs platines et modules optionnels associés sans que la responsabilité du Fabricant et de LEXTRONIC ne puisse être mise en cause, ni que les produits puissent être échangés au titre de la garantie.

Rappel sur l'évacuation des équipements électroniques usagés

Ce symbole présent sur les modules "PICBASIC" ainsi que leurs platines et modules optionnels associés et/ou leurs emballages indique que vous ne pouvez pas vous débarraser de ces produits de la même façon que vos déchets courants. Au contraire, vous êtes responsable de l'évacuation de ces produits lorsqu'ils arrivent en fin de vie (ou qu'ils sont hors d'usage) et à cet effet, vous êtes tenu de les remettre à un point de collecte agréé pour le recyclage des équipements électriques et électroniques usagés. Le tri, l'évacuation et le recyclage séparés de vos équipements usagés permettent de préserver les ressources naturelles et de s'assurer que ces équipements sont recyclés dans le respect de la santé humaine et de l'environnement. Pour plus d'informations sur les lieux de collecte des équipements électroniques usagés, veuillez contacter votre mairie ou votre service local de traitement des déchets.

Note for all residents of the European Union

This symbol on the product or on its packaging indicates that this product must not be disposed of with other household waste. Instead, it is your responsability to dispose of your waste equipement by handing it over to designated collection point for the recycling of waste electrical and electric equipement. The separate collection and recycling of your waste equipement at the time of disposal will help to conserve natural resources and ensure that it is recycled in a manner that protects human health and environnement. For more information about where you can drop off your waste equi-pement for recycling, please contact your local city office or your local hold waste disposal service.

Chapitre 1.

Descriptions des PICBASIC

Préface…

Bien que depuis plus d'une dizaine d'années les réalisations à base de microcontrôleurs soient devenues monnaies courantes, tant au niveau des milieux professionnels que "grand public", force est de reconnaître que leur généralisation au plus grand nombre reste du domaine de l'utopie. La maîtrise d'un langage de haut niveau (assembleur, langage 'C', etc...), la parfaite connaissance des caractéristiques du microcontrôleur choisi ou encore l'investissement de base nécessaire à la réalisation de sa première application sont autant d'obstacles suffisamment importants pour décourager bon nombre d'utilisateurs pourtant intéressés par les immenses possibilités de ces derniers.

Partant de cette constatation certains Fabricants ont, il y a quelques années de cela lancés une gamme complète de petits modules hybrides programmables très facilement en langage BASIC. Véritable révolution pour l'époque, ils ont remporté un franc succès en permettant à tout un chacun de développer très simplement des applications plus ou moins complexes avec un minimum de "connaissance" et de moyen. Néanmoins et malgré leur constante évolution, ces modules sont restés toutefois cantonnés aux milieux "amateurs" de part certaines limitations techniques qui leur sont propres.

C'est dans ce contexte que sont apparus sur le marché des modules hybrides microcontrôlés programmables en BASIC "de nouvelle génération". Fabriqués par le Coréen COMFILE TECHNOLOGY et déjà commercialisés depuis plusieurs années avec succès, ces modules appelés "PICBASIC" sont importés et disponibles en Exclusivité en France par la société LEXTRONIC.

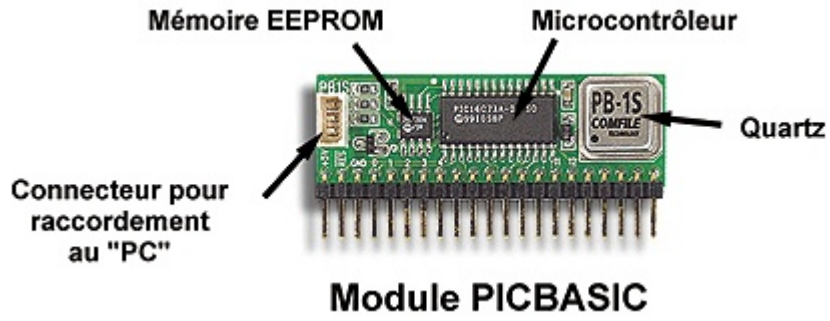
Plus puissants (grâce à leur architecture programme "pseudo multitâche"), plus complets (avec leurs convertisseurs "N/A" intégrés, leur horloge temps réel et leurs instructions dédiées), plus rapides (jusqu'à 56.000 commandes/sec. traitées), plus simples à programmer (de part leur possibilité de débuggage en "mode émulateur"), ces nouveaux modules sont tout naturellement plébiscités tant par les milieux professionnels que "grand public" en raison de leurs tarifs extrêmement compétitifs et leur importante gamme de périphériques divers à l'origine de leur fulgurant succès.

Qu'est-ce qu'un PICBASIC ?

Les PICBASIC sont de petits modules hybrides destinés à prendre place au coeur de vos futures applications afin d'en assurer une gestion "informatique". Programmables en langage BASIC évolué, ils sont disponibles:

- Soit sous la forme de modules hybrides au format S.I.L ou DIL.

Ces modèles livrés pré-assemblés se composent d'un microcontrôleur associé à une mémoire non volatile (EEPROM ou FLASH), à un quartz, à quelques composants et à un connecteur destiné à les relier au PC de programmation. Il ne vous suffira qu'une source d'alimentation +5 V pour les rendre opérationnels.



Module PICBASIC

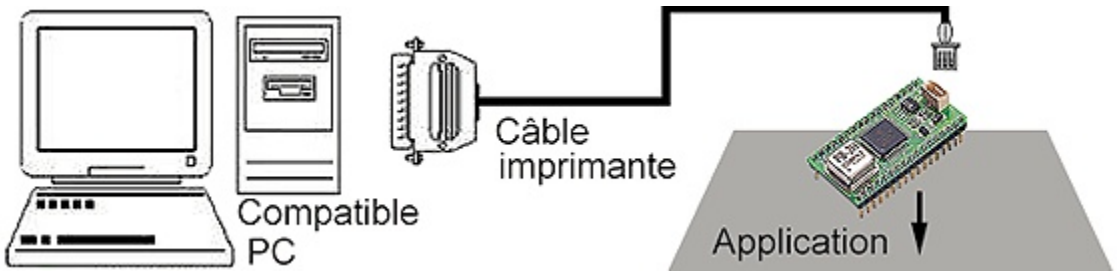
- Soit sous la forme de circuits intégrés 28 ou 40 broches.

Ces circuits nécessiteront quelques composants externes (2 résistances, 1 diode, 3 condensateurs, un quartz et un connecteur destiné à les relier au PC de programmation) ainsi qu'une source d'alimentation +5 V pour les rendre opérationnels.

Principe de programmation

Les PICBASIC se programment très facilement en langage “BASIC” par l’intermédiaire d’un compatible PC et d’un puissant logiciel de développement (”PICBASIC-LAB” ou ”PICBASIC Studio” suivant la version du système d’exploitation que vous utilisez) qui transformera vos instructions “BASIC” en codes spécifiques, lesquels seront alors transférés dans la mémoire du “PICBASIC” par le biais d’un cordon de liaison spécial préalablement raccordé au port imprimante (ou USB) de votre ordinateur.

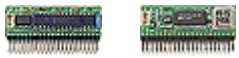
Une fois le PICBASIC ainsi “chargé”, ce dernier pourra être déconnecté du "PC" pour devenir autonome afin de réaliser votre programme par le biais de son microcontrôleur qui récupérera un à un les codes transférés pour les “traduire” en “action” adéquat.



Aperçu de la gamme

Les PICBASIC se déclinent en 4 catégories (répartie en 2 gammes « PB » et « PBM »):

- Les "PICBASIC-1B" et "PICBASIC-1S" qui se présentent sous la forme d’un hybride au format "S.I.L" (Gamme « PB »)



- Les "PICBASIC-2S" et "PICBASIC-2H" qui se présentent sous la forme d’un hybride au format "D.I.L" (Gamme « PB »).

- Les "PICBASIC-3B" et "PICBASIC-3H" qui se présentent sous la forme de circuit intégré au format "D.I.L" (Gamme « PB »).

- Les "PBM-R1" et "PBM-R5" au format "D.I.L" encapsulés dans un boîtier plastique (Gamme « PBM »).

Les 6 premiers modules peuvent se programmer sous environnement Windows98™ ou WindowsXP™. Les 2 derniers modules se programment uniquement sous environnement WindowsXP™.

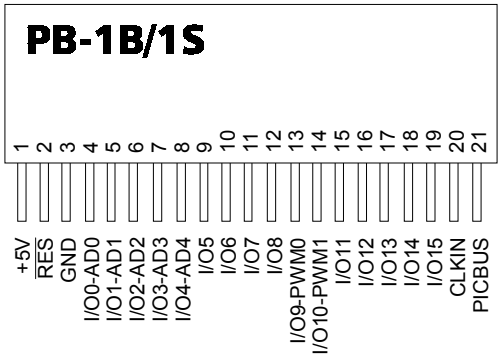
Spécifications techniques

	PB-1B	PB-1S	PB-2S	PB-2H	PB-3B	PB-3H	PBM-R1	PBM-R5
Gamme	PB	PB	PB	PB	PB	PB	PBM	PBM
Mémoire prog.	2 K	4 K	8 K	16 K	4 K	4 K	32 K	64 K
Mémoire RAM	96 octets	96 octets	96 octets	96 octets	79 octets	79 octets	8 K	32 K
Ports E/S	16	16	27	27	21	29	34	34
CPU	PIC16C73	PIC16C73	PIC16C74	PIC16C74	PIC16F876	PIC16F877	PIC16F877	PIC16F877
Fréq. quartz	4.19MHz	4.19MHz	4.19MHz	20MHz	20MHz	20MHz	20MHz	20MHz
Vitesse *	13.1 fois	13.9 fois	13.9 fois	3.1 fois	1 fois	1 fois	1.4 fois	1.4 fois
EEPROM Pour données							8 K	32 K
Nb de broches	21	21	34	34	28	40	40	40
Conv. A/N (Résolution)	5 (8 bit)	5 (8 bit)	8 (8 bit)	8 (8 bit)	5 (10 bit)	8 (10 bit)	8 (10 bit)	8 (10 bit)
A/N 12 bits								2 canaux
Sorties PWM (Résolution)	2 (8bit)	2 (8bit)	2 (8bit)	2 (8bit)	2 (8bit)	2 (8bit)	2 (10bit)	2 (10bit)
Buffer RS232	-	-	-	-	-	-	Dispo	Dispo
Horloge RTC	-	-	-	-	-	-	-	Dispo

- Ce paramètre donne une indication sur la vitesse d'exécution des PICBASIC. Ce dernier prend comme référence les modèles de PICBASIC les plus rapides que sont les PICBASIC-3B et PICBASIC-3H. Ainsi, le PICBASIC-1B est 13.1 fois plus lent que les PICBASIC-3B et PICBASIC3H.

Brochage des PICBASIC

Modèles « PICBASIC-1B (PB-1B) / PICBASIC-1S (PB-1S) »



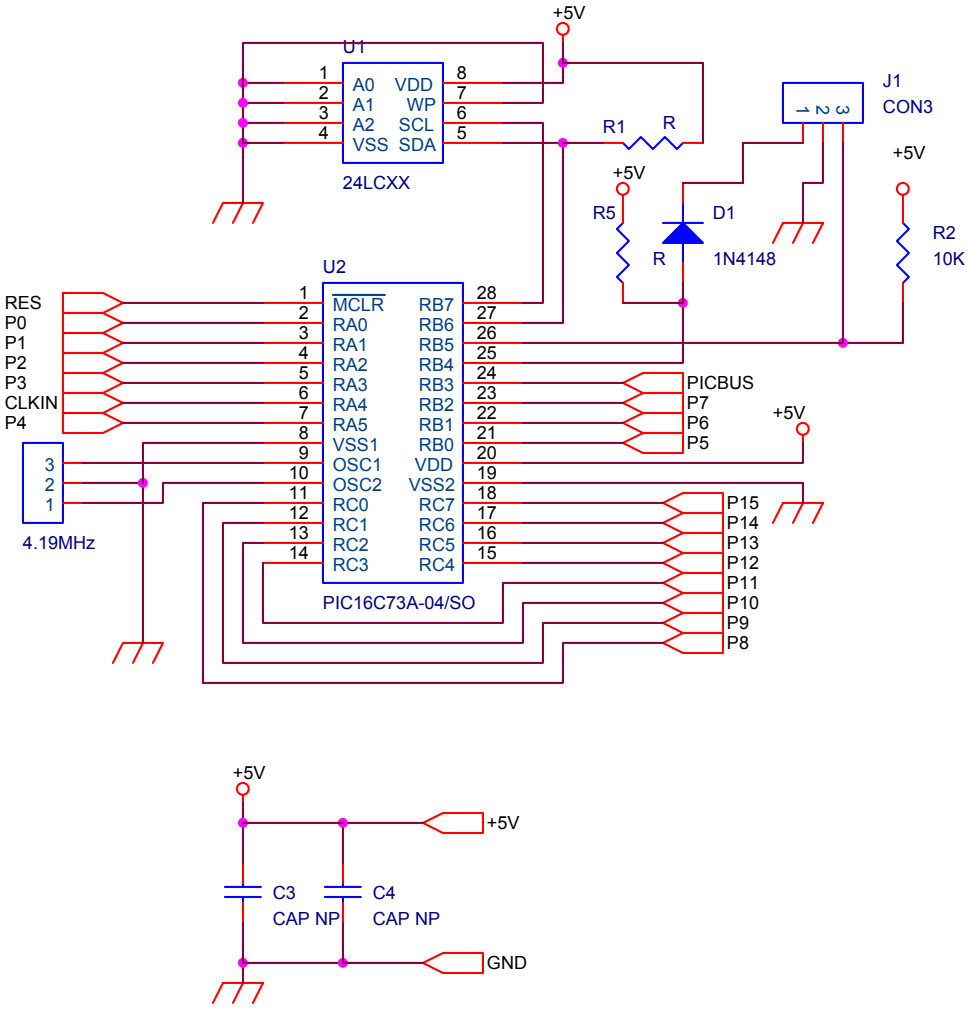
Broche N°	Description		Bloc	Niveau	Fonctions (*)
1	+5V	Alimentation 5V			
2	/RES	Reset, 5V			
3	GND	Masse			
4	I/O0/ AD0	Port 0	Bloc 0	TTL	Conv. A/N
5	I/O 1/ AD1	Port 1	Bloc 0	TTL	Conv. A/N
6	I/O 2/ AD2	Port 2	Bloc 0	TTL	Conv. A/N
7	I/O 3/ AD3	Port 3	Bloc 0	TTL	Conv. A/N
8	I/O 4/ AD4	Port 4	Bloc 0	TTL	Conv. A/N
9	I/O 5	Port 5	Bloc 0	TTL	
10	I/O 6	Port 6	Bloc 0	TTL	
11	I/O 7	Port 7	Bloc 0	TTL	
12	I/O 8	Port 8	Bloc 1	ST	
13	I/O 9/ PWM0	Port 9	Bloc 1	ST	Port PWM
14	I/O 10/ PWM1	Port 10	Bloc 1	ST	Port PWM
15	I/O 11	Port 11	Bloc 1	ST	
16	I/O 12	Port 12	Bloc 1	ST	
17	I/O 13	Port 13	Bloc 1	ST	
18	I/O 14	Port 14	Bloc 1	ST	
19	I/O 15	Port 15	Bloc 1	ST	
20	CLKIN	Entrée compteur		ST	
21	PICBUS	Port LCD			

- ST = Trigger de Schmitt
- TTL = Niveau d'entrée TTL

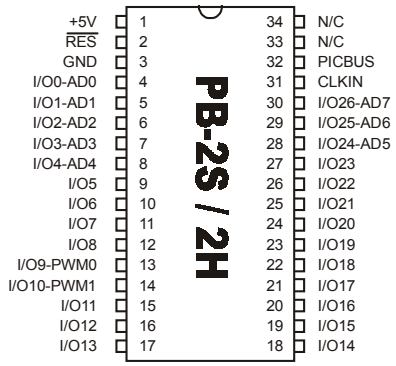
Les ports avec niveau d'entrée TTL permettent d'interpréter un niveau HAUT pour une tension supérieur à 1.4 V et un niveau BAS pour une tension inférieure à 1.3 V. Les entrées avec trigger de Schmitt permettent d'interpréter un niveau HAUT pour une tension supérieur à 3.4 V et un niveau BAS pour une tension inférieure à 3.3 V.

(*) Les fonctions de conversion A/N ne sont disponibles que sur le « PICBASIC-1S »

Schéma théorique des « PICBASIC-1B (PB-1B) / PICBASIC-1S (PB-1S) »



Modèles « PICBASIC-2S (PB-2S) / PICBASIC-2H (PB-2H) »

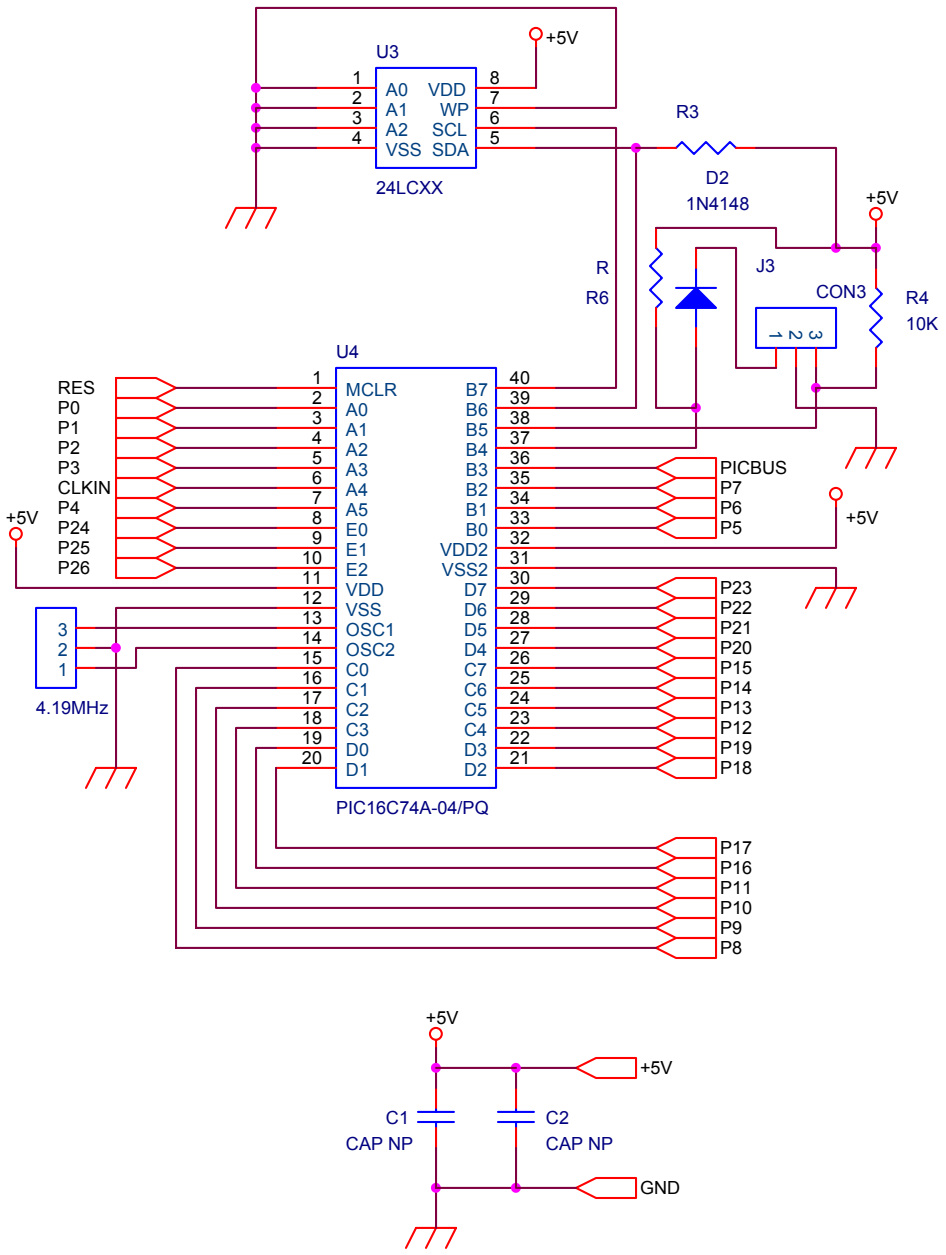


Broche N°	Description		Bloc	Niveau	Fonctions
1	+5V	Alimentation 5V			
2	/RES	Reset, 5V			
3	GND	Masse			
4	I/O0/AD0	Port 0	Bloc 0	TTL	Conv. A/N
5	I/O1/AD1	Port 1	Bloc 0	TTL	Conv. A/N
6	I/O2/AD2	Port 2	Bloc 0	TTL	Conv. A/N
7	I/O3/AD3	Port 3	Bloc 0	TTL	Conv. A/N
8	I/O4/AD4	Port 4	Bloc 0	TTL	Conv. A/N
9	I/O5	Port 5	Bloc 0	TTL	
10	I/O6	Port 6	Bloc 0	TTL	
11	I/O7	Port 7	Bloc 0	TTL	
12	I/O8	Port 8	Bloc 1	ST	
13	I/O9/PWM0	Port 9	Bloc 1	ST	Port PWM
14	I/O10/PWM1	Port 10	Bloc 1	ST	Port PWM
15	I/O11	Port 11	Bloc 1	ST	
16	I/O12	Port 12	Bloc 1	ST	
17	I/O13	Port 13	Bloc 1	ST	
18	I/O14	Port 14	Bloc 1	ST	
19	I/O15	Port 15	Bloc 1	ST	
20	I/O16	Port 16	Bloc 2	ST	
21	I/O17	Port 17	Bloc 2	ST	
22	I/O18	Port 18	Bloc 2	ST	
23	I/O19	Port 19	Bloc 2	ST	
24	I/O20	Port 20	Bloc 2	ST	
25	I/O21	Port 21	Bloc 2	ST	
26	I/O22	Port 22	Bloc 2	ST	
27	I/O23	Port 23	Bloc 2	ST	
28	I/O24/AD5	Port 24	Bloc 3	ST	Conv. A/N
29	I/O25/AD6	Port 25	Bloc 3	ST	Conv. A/N
30	I/O26/AD7	Port 26	Bloc 3	ST	Conv. A/N
31	CLKIN	Entrée compteur		ST	
32	PICBUS	Port LCD			
33, 34	N/C	Non connectés			

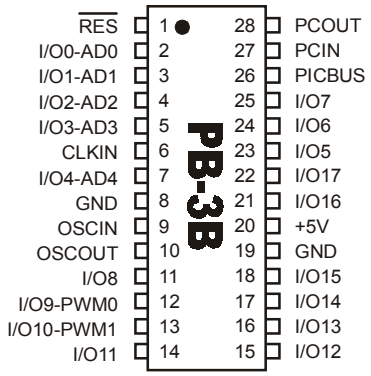
- ST = Trigger de Schmitt
- TTL = Niveau d'entrée TTL

Les ports avec niveau d'entrée TTL permettent d'interpréter un niveau HAUT pour une tension supérieur à 1.4 V et un niveau BAS pour une tension inférieure à 1.3 V. Les entrées avec trigger de Schmitt permettent d'interpréter un niveau HAUT pour une tension supérieur à 3.4 V et un niveau BAS pour une tension inférieure à 3.3 V.

Schéma théorique des « PICBASIC-2S (PB-2S) / PICBASIC-2H (PB-2H) »



Modèle « PICBASIC-3B (PB-3B) »



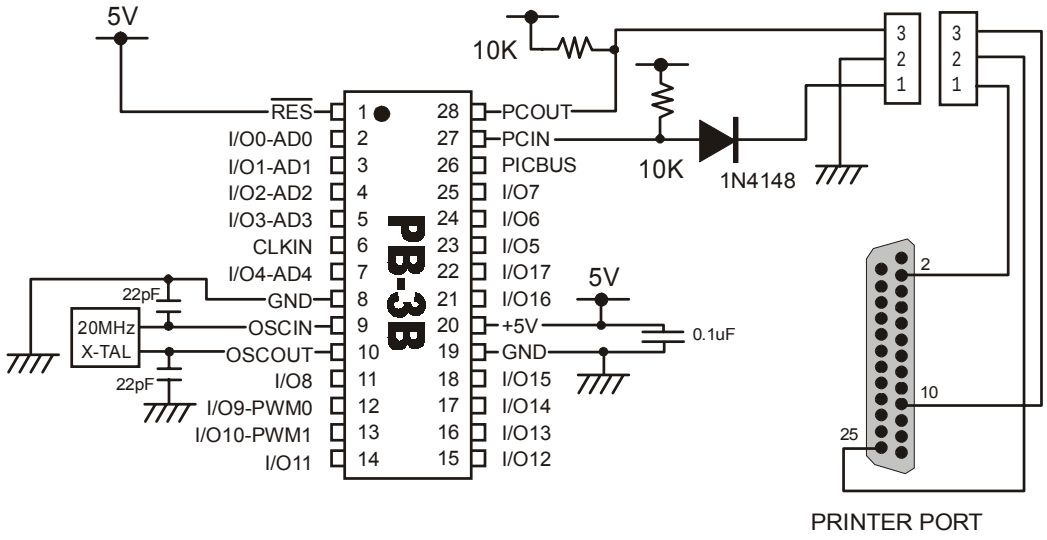
Broche N°	Description		Bloc	Niveau	Fonctions
1	/RES	Reset			
2	I/O0-AD0	Port 0		TTL	Conv. A/N
3	I/O1-AD1	Port 1		TTL	Conv. A/N
4	I/O2-AD2	Port 2		TTL	Conv. A/N
5	I/O3-AD3	Port 3		TTL	Conv. A/N
6	CLKIN	Entrée compteur			
7	I/O4-AD4	Port 4		TTL	Conv. A/N
8	GND	Masse			
9	OSCIN	Connexion Quartz			
10	OSCOUT	Connexion Quartz			
11	I/O8	Port 8	Bloc 1	ST	
12	I/O9-PWM0	Port 9	Bloc 1	ST	Port PWM
13	I/O10-PWM1	Port 10	Bloc 1	ST	Port PWM
14	I/O11	Port 11	Bloc 1	ST	
15	I/O12	Port 12	Bloc 1	ST	
16	I/O13	Port 13	Bloc 1	ST	
17	I/O14	Port 14	Bloc 1	ST	
18	I/O15	Port 15	Bloc 1	ST	
19	GND	Masse			
20	+5V	Alimentation 5V			
21	I/O16	Port 16		ST	Interrupt.
22	I/O17	Port 17		ST	
23	I/O5	Port 5		ST	
24	I/O6	Port 6		ST	
25	I/O7	Port 7		ST	
26	PICBUS	Port LCD			
27	PCIN	Connexion PC (IN)			
28	PCOUT	Connexion PC (OUT)			

- Toutes les broches (sauf $V_{DD,}$ V_{SS} et RESET) sont des ports d'E/S.
- /RES (circuit de RESET interne) doit être relié à V_{DD} (+5 V)
- Le PICBASIC-3B ne dispose que d'un seul bloc (les bloc 0 et bloc 2 sont ignorés).
- OSCIN et OSCOUT doivent être reliés à un quartz de 20 MHz.

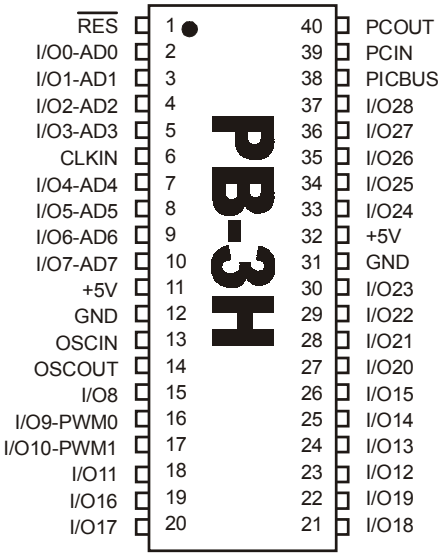
- ST = Trigger de Schmitt
- TTL = Niveau d'entrée TTL

Les ports avec niveau d'entrée TTL permettent d'interpréter un niveau HAUT pour une tension supérieur à 1.4 V et un niveau BAS pour une tension inférieure à 1.3 V. Les entrées avec trigger de Schmitt permettent d'interpréter un niveau HAUT pour une tension supérieur à 3.4 V et un niveau BAS pour une tension inférieure à 3.3 V.

Comme indiqué précédemment, le « PICBASIC-3B » nécessite quelques composants externes additionnels pour être exploité. Les 2 résistances et la diode devront être câblés le plus près possible du PICABSIC. Le schéma donné ci-dessous montre comment le relier au port imprimante d'un PC afin de pouvoir le programmer (le schéma du raccordement de la prise Sub-D25 broches n'est valable que si votre ordinateur dispose d'un système d'exploitation de type Windows 98™ - Voir la chapitre 4 « Les câbles de téléchargement » ci-après pour plus d'infos).



Modèle « PICBASIC-3H (PB-3H) »



Broche N°	Description		Bloc	Niveau	Fonction
1	/RES	Reset			
2	I/O0-AD0	Port 0	Bloc 0	TTL	Conv. A/N
3	I/O1-AD1	Port 1	Bloc 0	TTL	Conv. A/N
4	I/O2-AD2	Port 2	Bloc 0	TTL	Conv. A/N
5	I/O3-AD3	Port 3	Bloc 0	TTL	Conv. A/N
6	CLKIN	Entrée compteur			
7	I/O4-AD4	Port 4	Bloc 0	TTL	Conv. A/N
8	I/O5-AD5	Port 5	Bloc 0	TTL	Conv. A/N
9	I/O6-AD6	Port 6	Bloc 0	TTL	Conv. A/N
10	I/O7-AD7	Port 7	Bloc 0	TTL	Conv. A/N
11	+5V	Alimentation 5V			
12	GND	Masse			
13	OSCIN	Connexion Quartz			
14	OSCOUT	Connexion Quartz			
15	I/O8	Port 8	Bloc 1	ST	
16	I/O9-PWM0	Port 9	Bloc 1	ST	Port PWM
17	I/O10-PWM1	Port 10	Bloc 1	ST	Port PWM
18	I/O11	Port 11	Bloc 1	ST	
19	I/O16	Port 16	Bloc 2	ST	
20	I/O17	Port 17	Bloc 2	ST	
21	I/O18	Port 18	Bloc 2	ST	
22	I/O19	Port 19	Bloc 2	ST	
23	I/O12	Port 12	Bloc 1	ST	
24	I/O13	Port 13	Bloc 1	ST	
25	I/O14	Port 14	Bloc 1	ST	
26	I/O15	Port 15	Bloc 1	ST	
27	I/O20	Port 20	Bloc 2	ST	
28	I/O21	Port 21	Bloc 2	ST	
29	I/O22	Port 22	Bloc 2	ST	
30	I/O23	Port 23	Bloc 2	ST	
31	GND	Masse			

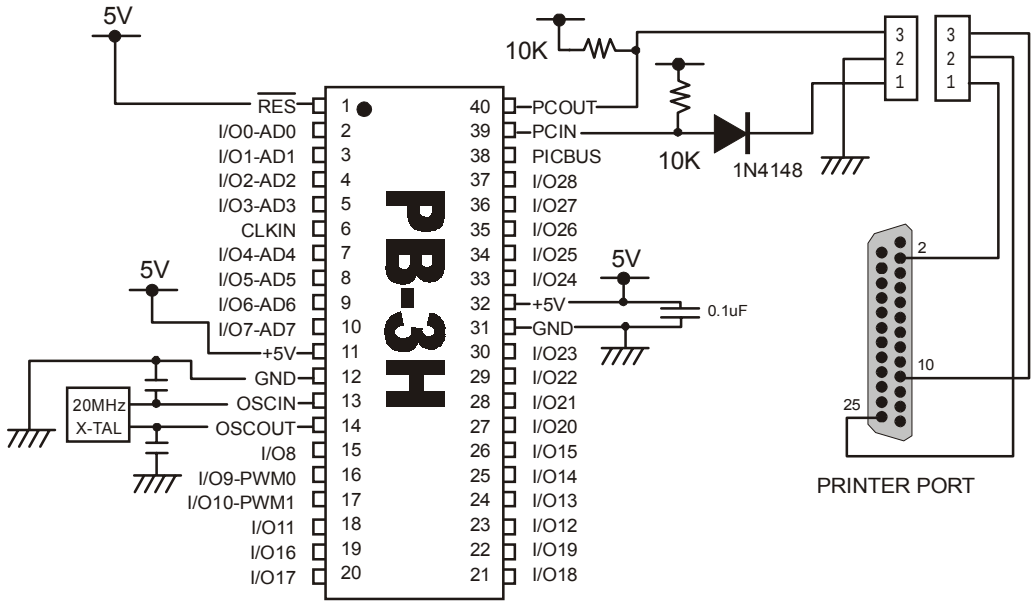
32	+5V	Alimentation 5V			
33	I/O24	Port 24		ST	Interrupt.
34	I/O25	Port 25		ST	
35	I/O26	Port 26		ST	
36	I/O27	Port 27		ST	
37	I/O28	Port 28		ST	
38	PICBUS	Port LCD			
39	PCIN	Connexion PC (IN)			
40	PCOUT	Connexion PC (OUT)			

- Toutes les broches (sauf $V_{DD,}$ V_{SS} et RESET) sont des ports d'E/S.
- /RES (circuit de RESET interne) doit être relié à V_{DD} (+5 V)
- OSCIN et OSCOUT doivent être reliés à un quartz de 20 MHz.

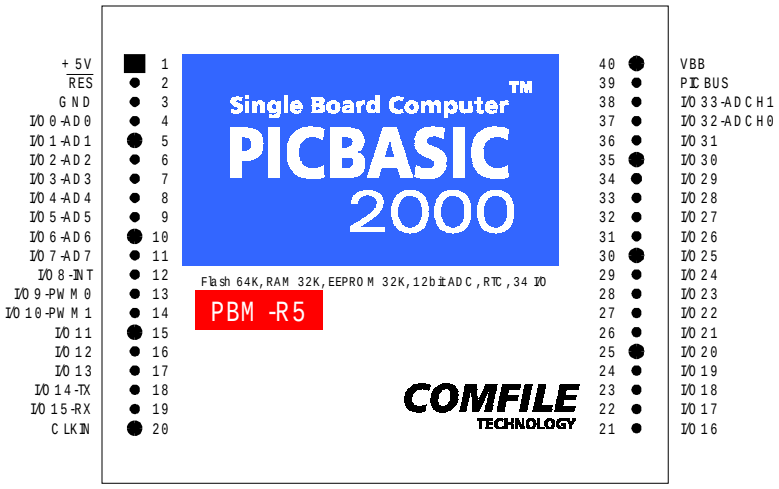
- ST = Trigger de Schmitt
- TTL = Niveau d'entrée TTL

Les ports avec niveau d'entrée TTL permettent d'interpréter un niveau HAUT pour une tension supérieur à 1.4 V et un niveau BAS pour une tension inférieure à 1.3 V. Les entrées avec trigger de Schmitt permettent d'interpréter un niveau HAUT pour une tension supérieur à 3.4 V et un niveau BAS pour une tension inférieure à 3.3 V.

Comme indiqué précédemment, le « PICBASIC-3H » nécessite quelques composants externes additionnels pour être exploité. Les 2 résistances et la diode devront être câblés le plus près possible du PICABSIC. Le schéma donné ci-dessous montre comment le relier au port imprimante d'un PC afin de pouvoir le programmer (le schéma du raccordement de la prise Sub-D25 broches n'est valable que si votre ordinateur dispose d'un système d'exploitation de type Windows 98™ - Voir la rubrique « Les câbles de téléchargement » ci-après pour plus d'infos).



Modèles « PICBASIC-R1 (PBM-R1) / PICBASIC-R5 (PBM-R5) »



Pin	Description		Bloc	Niveau	Fonction
1	+5V	Alimentation 5V			
2	/RES	Reset, 5V			
3	GND	Masse			
4	I/O0/AD0	Port 0	Bloc 0	TTL	Conv. A/N 10 bits
5	I/O1/AD1	Port 1	Bloc 0	TTL	Conv. A/N 10 bits
6	I/O2/AD2	Port 2	Bloc 0	TTL	Conv. A/N 10 bits
7	I/O3/AD3	Port 3	Bloc 0	TTL	Conv. A/N 10 bits
8	I/O4/AD4	Port 4	Bloc 0	TTL	Conv. A/N 10 bits
9	I/O5/AD5	Port 5	Bloc 0	TTL	Conv. A/N 10 bits
10	I/O6/AD6	Port 6	Bloc 0	TTL	Conv. A/N 10 bits
11	I/O7/AD7	Port 7	Bloc 0	TTL	Conv. A/N 10 bits
12	I/O8 / INT	Port 8	Bloc 1	ST	Interrupt.
13	I/O9 / PWM0	Port 9	Bloc 1	ST	Port PWM 10 bits
14	I/O10 / PWM1	Port 10	Bloc 1	ST	Port PWM 10 bits
15	I/O11	Port 11	Bloc 1	ST	
16	I/O12	Port 12	Bloc 1	ST	
17	I/O13	Port 13	Bloc 1	ST	
18	I/O14 / TX	Port 14	Bloc 1	ST	RS232C (émission)
19	I/O15 / RX	Port 15	Bloc 1	ST	RS232C (Réception)
20	CLKIN	Entrée compteur		ST	
21	I/O16	Port 16	Bloc 2	ST	
22	I/O17	Port 17	Bloc 2	ST	
23	I/O18	Port 18	Bloc 2	ST	
24	I/O19	Port 19	Bloc 2	ST	
25	I/O20	Port 20	Bloc 2	ST	
26	I/O21	Port 21	Bloc 2	ST	
27	I/O22	Port 22	Bloc 2	ST	
28	I/O23	Port 23	Bloc 2	ST	
29	I/O24	Port 24	Bloc 3	ST	
30	I/O25	Port 25	Bloc 3	ST	

31	I/O26	Port 26	Bloc 3	ST	
32	I/O27	Port 27	Bloc 3	ST	
33	I/O28	Port 28	Bloc 3	ST	
34	I/O29	Port 29	Bloc 3	ST	
35	I/O30	Port 30	Bloc 3	ST	
36	I/O31	Port 31	Bloc 3	ST	
37	I/O32/ADCH0	Port 32	Conv. A/N		Conv. A/N 12 bits (*)
38	I/O33/ADCH1	Port 33	Conv. A/N		Conv. A/N 12 bits (*)
39	PICBUS	Port LCD			
40	VBB	Alim capa RTC (*)	Capacité		

- Toutes les broches (sauf $V_{DD,}$ V_{SS} et RESET) sont des ports d'E/S.
- /RES (circuit de RESET interne) doit être relié à V_{DD} (+5 V)
- OSCIN et OSCOUT doivent être reliés à un quartz de 20 MHz.
- Les ports I/O 32/33 sont uniquement utilisables (avec le PBM-R5) comme entrées de conversion A/N.
- La broche 40 du « PICBASIC-R5 » (V_{BB}) sert à la recharge de la super capacité de 0,1 F intégrée permettant la sauvegarde de l'horloge RTC. Si le « PICBASIC-R5 » n'est pas alimenté pendant plus de 6 mois, il vous faudra appliquer une tension sur cette entrée afin de procéder à la recharge de la capacité (voir description des instructions liées à l'horloge RTC du « PICBASIC-R5 pour plus d'infos).

- ST = Trigger de Schmitt
- TTL = Niveau d'entrée TTL

Les ports avec niveau d'entrée TTL permettent d'interpréter un niveau HAUT pour une tension supérieur à 1.4 V et un niveau BAS pour une tension inférieure à 1.3 V. Les entrées avec trigger de Schmitt permettent d'interpréter un niveau HAUT pour une tension supérieur à 3.4 V et un niveau BAS pour une tension inférieure à 3.3 V.

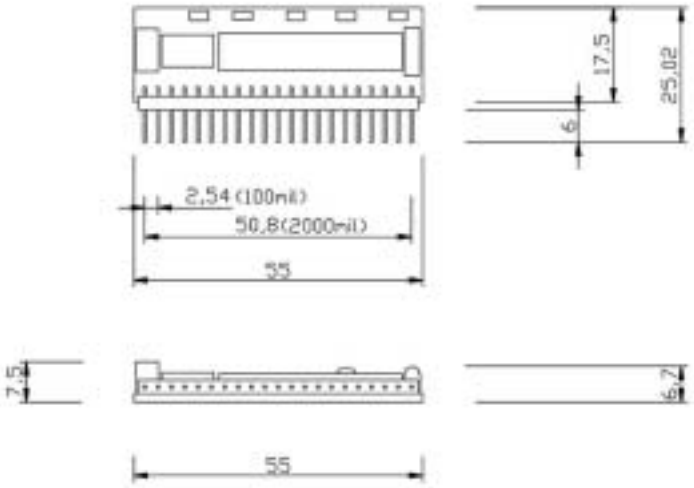
* Fonctions uniquement présentent sur le « PBM-R5 » (ne pas utiliser ces broches sur le «PBM-R1»).

Caractéristiques électriques des « PICBASIC »

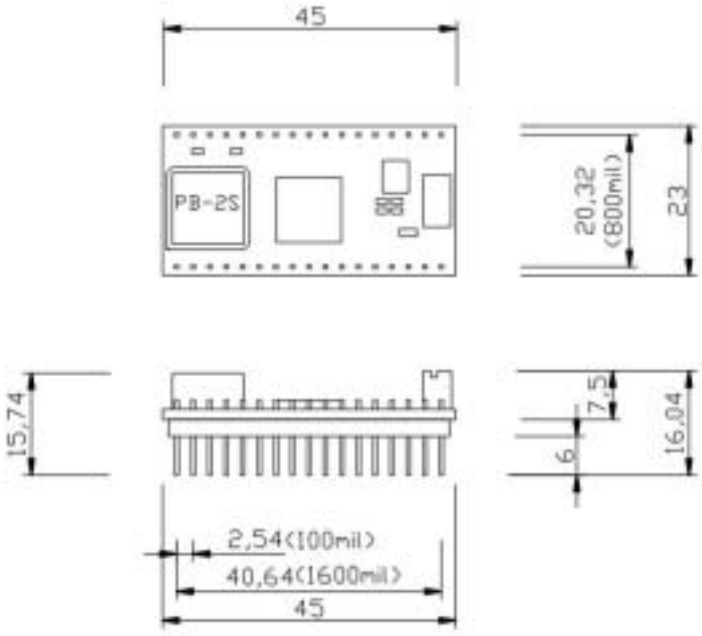
Tension V_{DD}	: 4.75 Vc ~ 5.5 Vcc
Cosommations moyennes	: PB-1B, 1S, 2S : 7 mA env.
	: PB-2H : 15 mA env.
	: PB-3B, 3H : 6 mA env.
	: PBM-R1, R5 : 50 mA env.
Températures de stockage	: -40 °C ~ 125 °C
Températures fonctionnement	: +10 °C ~ + 50 °C
Courant de sortie des ports	: 25 mA
Courant max admissible V_{SS}	: 300 mA
Courant max admissible V_{DD}	: 250 mA

Dimensions des PICBASIC

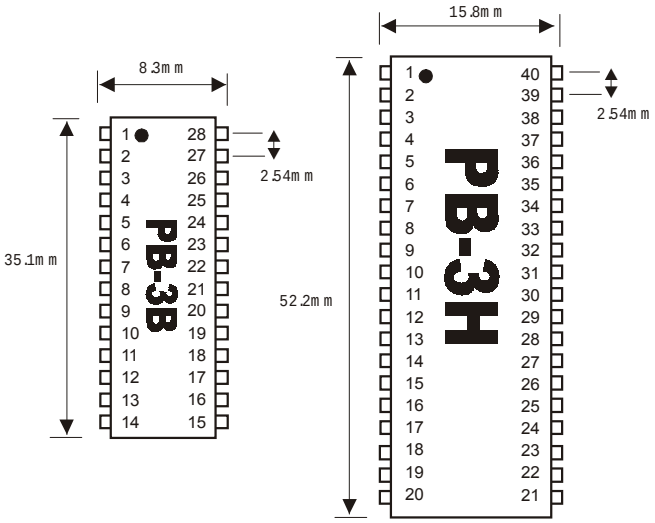
PB-1B / PB-1S



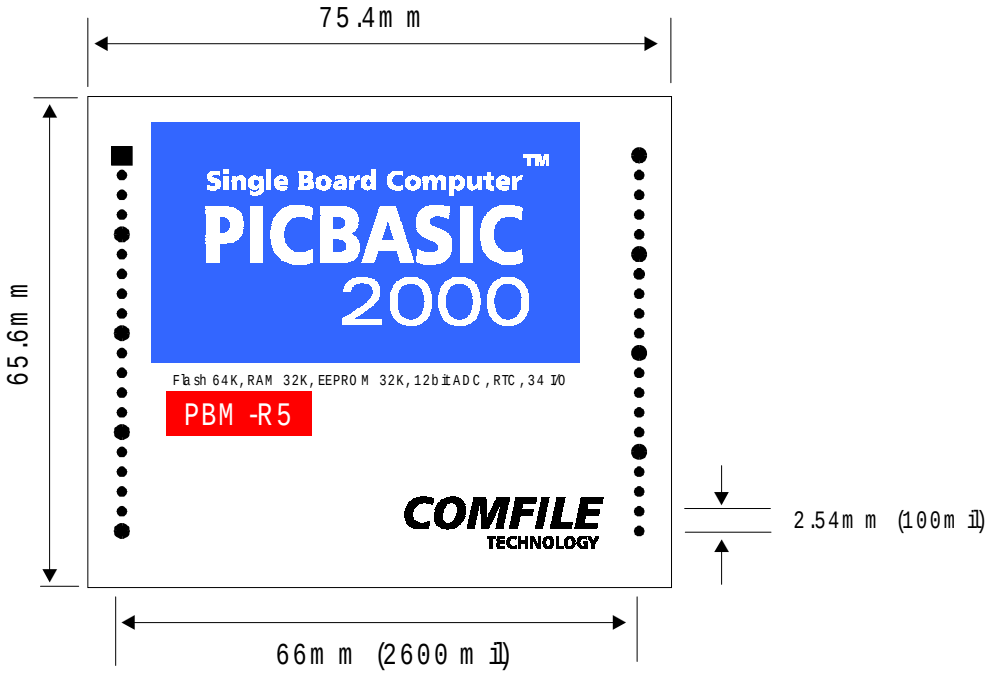
PB-2S / PB-2H



PB-3X


15.8mm
8.3mm
2.54mm
35.1mm
52.2mm
2.54mm
PB-3B
PB-3H

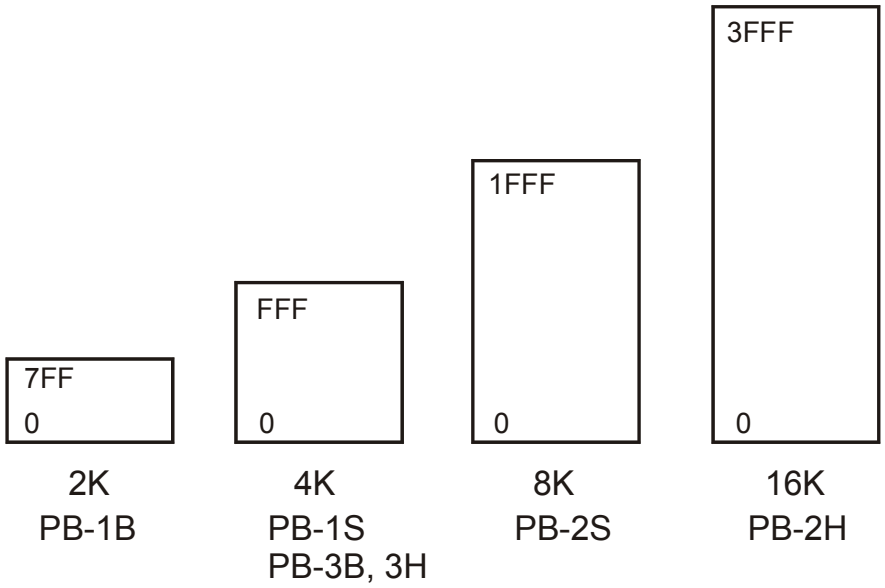
PBM-Rx


75.4mm
65.6mm
Single Board Computer™
PICBASIC
2000
Flash 64K, RAM 32K, EEPROM 32K, 12bit ADC, RTC, 34 I/O
PBM-R5
COMFILE
TECHNOLOGY
2.54mm (100mil)
66mm (2600mil)

Organisation mémoire des PICBASIC

Modèles « PB-1B / 1S / 2S / 2H / 3B / 3H »

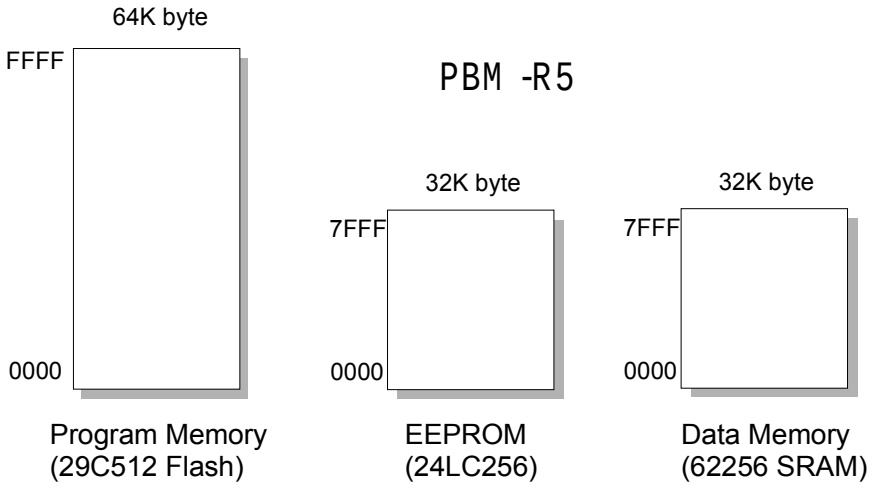
Les PICBASIC « PB-1B / 1S / 2S / 2H » disposent d'une mémoire programme EEPROM de 2 K à 16 K (cette mémoire peut également être utilisée pour stocker des données). Leur mémoire EEPROM est reliée au microcontrôleur via une liaison sériel de type I2C™. Dans le cas des PICBASIC « PB-3B / 3-H », la mémoire EEPROM est directement intégrée au microcontrôleur (ce qui explique que ces derniers soient plus rapides). Pour tous les modèles de PICBASIC, votre programme débutera à l'adresse 0 (une fois votre programme chargé dans le PICBASIC, il vous sera possible d'utiliser la mémoire restante pour stocker des données non volatile).



Tous les PICBASIC disposent également d'une mémoire volatile de type SRAM de 96 octets (79 octets pour les « PB-3B / PB-3H »). Cette mémoire SRAM est directement intégrée dans les microcontrôleurs afin de pouvoir stocker les variables. A chaque coupure d'alimentation, les données présentent dans la mémoire SRAM seront effacées.

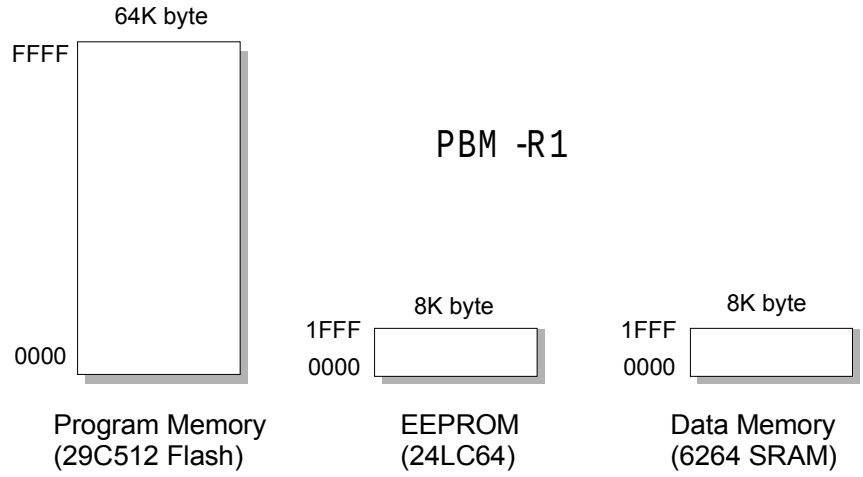
Modèle « PBM-R5 »

Les PICBASIC « PBM-R5 » intègrent une mémoire FLASH 29C512 (64 K) dédiée à la sauvegarde de votre programme, ainsi qu'une mémoire SRAM 62256 (32 K) et une mémoire EEPROM 24LC256 (32 K). Les « PBM-R5 » disposent d'une horloge temps réel (RTC) et de 2 entrées supplémentaires de conversion « A/N » 12 bits.



Modèle « PBM-R1 »

Les PICBASIC « PBM-R1 » intègrent une mémoire FLASH 29C512 (64 K) dédiée à la sauvegarde de votre programme, ainsi qu'une mémoire SRAM 6264 (8 K) et une mémoire EEPROM 24LC64 (8 K). Contrairement au « PBM-R5 », les PICBASIC « PBM-R1 » ne disposent pas d'horloge temps réel (RTC), ni d'entrées de conversion A/N 12 bits.



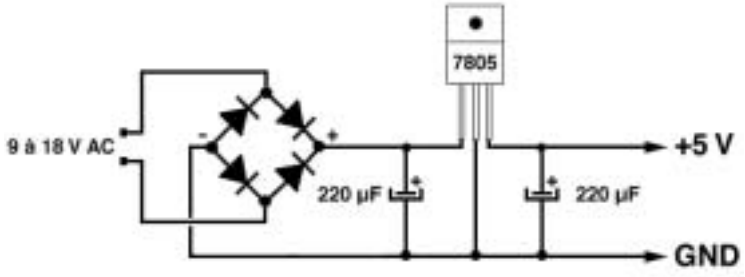
A l'inverse des autres modèles de PICBASIC, la mémoire SRAM des « PBM-R1 / PBM-R5 » n'est pas initialisée au moment de leur mise sous tension ou d'un RESET (des valeurs aléatoires peuvent donc être présentent). Il conviendra donc si nécessaire de réaliser une initialisation des données au sein de votre programme BASIC.

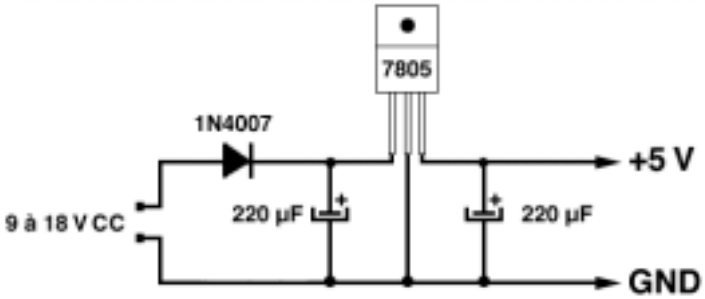
Chapitre 2.

Intégration des PICBASIC

Alimentation des PICBASIC

Tous les modules “PICBASIC” doivent impérativement être alimentés sous une tension de + 5 V (voir schéma type préconisé dans le cadre d’une platine de test). L’utilisation du pont redresseur peut être remplacé par une simple diode de protection contre les inversions de polarité, si la tension d’entrée est continue. Enfin le 7805 peut être remplacé par un 78L05 (plus petit), si la consommation totale de l'application n'excède pas 100 mA. Placez également impérativement un condensateur de découplage de 0,1 uF en parallèle sur l’alimentation du PICBASIC et au plus près de celui-ci.





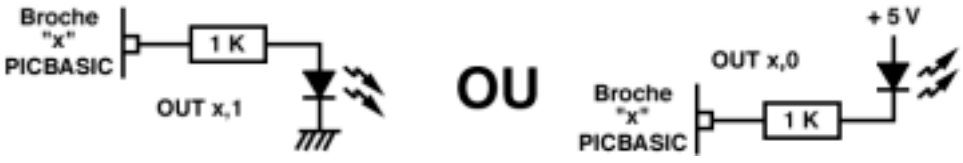
Reset des PICBASIC

La broche "RESET" des modules "PICBASIC" doit simplement être reliée au +5 V (avec une connexion au plus court).

Pilotage de dispositifs externes

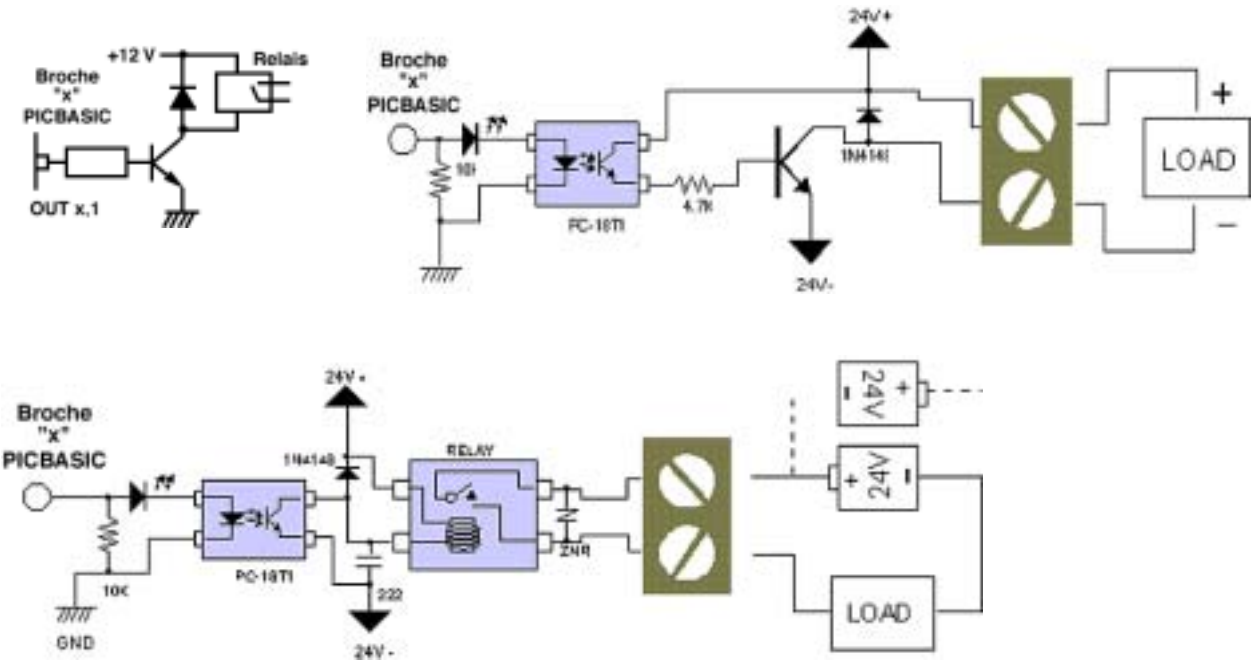
(schémas donnés à titre indicatif)

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



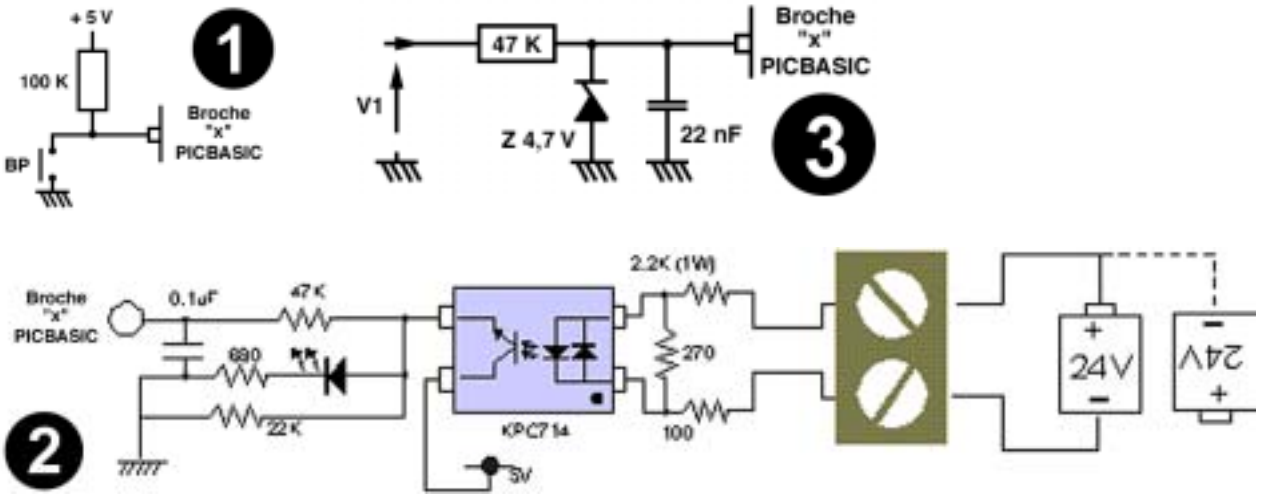
Attention toutefois à ne pas allumer plus de 6 ou 8 leds à la fois (utilisez dans ce cas des transistors d’interface).

Si la consommation des dispositifs à piloter devait dépasser les 25 mA, il conviendra alors d’avoir impérativement recours à l'utilisation d’un relais, d’un optocoupleur ou d’une combinaison des deux (voir exemples de schémas ci-dessous).



Gestion des broches configurées en entrée (schémas donnés à titre indicatif)

La "lecture" de contacts externes par les broches des modules "PICBASIC" est très simple. Dans le cadre de boutons-poussoirs ou d'interrupteurs, il suffira de réaliser le schéma ci-dessous (1). La longueur des câbles reliant les boutons-poussoirs au "PICBASIC" ne devra pas excéder 2 à 3 centimètres (il est également conseillé d'adjoindre un condensateur de 47 nF un parallèle sur le bouton-poussoir). Ce type de schéma convient pour la réalisation de clavier de saisie par exemple.



Si vous devez par contre relier des contacts déportés sur une plus grande longueur, il conviendra d'utiliser une interface adéquate (avec des optocoupleurs par exemple) afin d'éviter que des parasites ne "remontent" par les câbles et ne provoquent des perturbations ou dans certains cas extrêmes n'endommages les broches du "PICBASIC" – voir schéma (2). Cette condition est d'autant plus importante si le PICBASIC est exploité en environnement perturbé ou en présence d'éléments perturbateurs potentiels (moteurs, bobines, etc…).Si vous devez interfacer les entrées tout-ou-rien du "PICBASIC" avec des tensions supérieures à + 5 Vcc (avec des longueurs de câbles très courtes de l'ordre de 2 à 3 centimètres), vous pourrez utiliser le schéma (3). Dans le cadre de mesures de valeurs analogiques dont la tension maximale serait supérieure à + 5 V, il conviendra d'avoir recours à l'utilisation d'un pont diviseur en s'assurant toujours que la tension en entrée du "PICBASIC" ne dépasse jamais les + 5 Vcc.

Précautions d'usages :

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".

Correctement utilisé, votre PICBASIC vous permettra de réaliser d'innombrables quantités de montages et d'applications dont vous ne pouvez même pas imaginer la puissance. Toutefois il vous faut impérativement garder à l'esprit que le PICBASIC n'est rien d'autre qu'un microcontrôleur et au même titre qu'avec tout autre microcontrôleur il vous faut respecter certaines règles de bases afin d'éviter qu'il ne rendre l'âme !

1) Ne jamais alimenter les PICBASIC sous une tension supérieur à +5 Vcc - Ne jamais inverser la polarité d'alimentation.

2) Si vous appliquez des tensions issues de capteurs ou de dispositifs extérieurs sur les PICBASIC:

- Vérifiez toujours que ces tensions soient égales ou inférieures à + 5 Vcc.
- Coupez **en PRIORITÉ** l'alimentation des capteurs externes **AVANT** de couper celle du PICBASIC afin d'éviter qu'une tension soit toujours présente sur le PICBASIC alors que ce dernier n'est plus alimenté (sans quoi le PICBASIC serait endommagé).
- Selon la même recommandation que ci-dessus, vérifiez que vous ne disposez pas de condensateurs de forte valeur reliés sur les entrées des PICBASIC, lesquels pourront stocker une tension qui viendra alors se décharger dans le PICBASIC lorsque vous couperez les alimentations.

3) Les fils ramenant des tensions sur les entrées de conversion « A/ N » du PICBASIC ne doivent jamais quelques cm.

4) Lorsque vous utilisez les ports du PICBASIC en entrées, n'utilisez jamais de grand fils pour y raccorder des boutons-poussoirs et

autres capteurs sans avoir recours à un circuit de mise en forme et de protection (circuit RC avec zener de protection ou opto-coupleur – voir ci-après). Si pour vos tests vous n'utilisez pas de protection de ce type, limitez la longueur de vos fils à 3 - 4 cm afin d'éviter les phénomènes de "latch-up" ou de destruction par électricité statique.

5) Utilisez impérativement des composants de protection et d'anti-parasitage suffisament adaptés lorsque vous pilotez des charges inductives (moteurs ou autres exemple) afin d'éviter que des courants induits ne perturbent ou ne détruisent le PICBASIC.

6) Découplez rigoureusement l'alimentation du PICBASIC (au plus près de celui-ci).

7) 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).

8) Passez toujours par un montage à transistor ou à opto-coupleur pour alimenter et piloter un dispositif consommant plus d'une vingtaine de milli-ampère.

9) 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).

10) Comme TOUT microcontrôleur, les PICBASIC sont sensibles à l'électricité statique. Ces derniers devront donc être manipulés (et soudés) avec les précautions qui s'imposent afin d'éviter leur destruction ou leur fragilisation.

11) Ne rallongez JAMAIS le câble de téléchargement des PICBASIC.

12) Utilisez impérativement un composant « MAX232 » si vous reliez un port série du PICBASIC au port RS232 d'un PC.

13) Ne déconnectez JAMAIS le câble de programmation et/ou ne coupez JAMAIS l'alimentation du PICBASIC lorsque ce dernier est en cours de programmation.

En cas de non respect des limites et des conditions d'utilisations indiquées dans ce manuel, la fiabilité et la durée de vie des modules PICBASIC sera remise en cause (sans que la responsabilité de Lextronic puisse être mise en cause, ni que l'échange du module PICBASIC ne puisse être pris en charge au titre de la garantie).

Chapitre 3.

Les logiciels de programmation

Développement avec les « PICBASIC » :

Le développement avec les PICBASIC nécessite l'utilisation d'un logiciel adapté préalablement installé sur votre ordinateur. Ce logiciel intègre:

- Un éditeur (qui vous pemrettra d'écrire votre programme BASIC)
- Un module de téléchargement qui vous permettra de transférer votre programme au sein du PICBASIC par l'intermédiaire d'un câble spécial (via le port parallèle ou USB du PC)
- Un module d'émulation (qui vous permettra tant que le module est relié au PC d'excécuter le programme pas-à-pas, de vérifier l'état de toutes les variables crées, de les modifier...

Suivant le type de module PICBASIC que vous utilisez et suivant la version du système d'exploitation dont vous disposez, il existe 2 versions du logiciel de développement.

Le logiciel "**PICBASIC-LAB**" permet :

- La programmation des "PICBASIC" série 1 / 2 / 3 (à savoir les PICBASIC-1B / 1S / 2S / 2H / 3 B / 3H) sous environnement Windows™ 98/Me/Se à l'aide d'un câble de programmation parallèle.

Le logiciel "**PICBASIC-STUDIO**" permet :

- La programmation des PICBASIC-1B / 1S / 2S / 2H / 3 B / 3H / PBM-R1 / PBM-R5) sous environnement Windows™ XP à l'aide d'un câble de programmation parallèle.

Ou

- La programmation des PICBASIC-2H / 3 B / 3H / PBM-R1 / PBM-R5) sous environnement Windows™ XP à l'aide d'un câble de programmation USB.

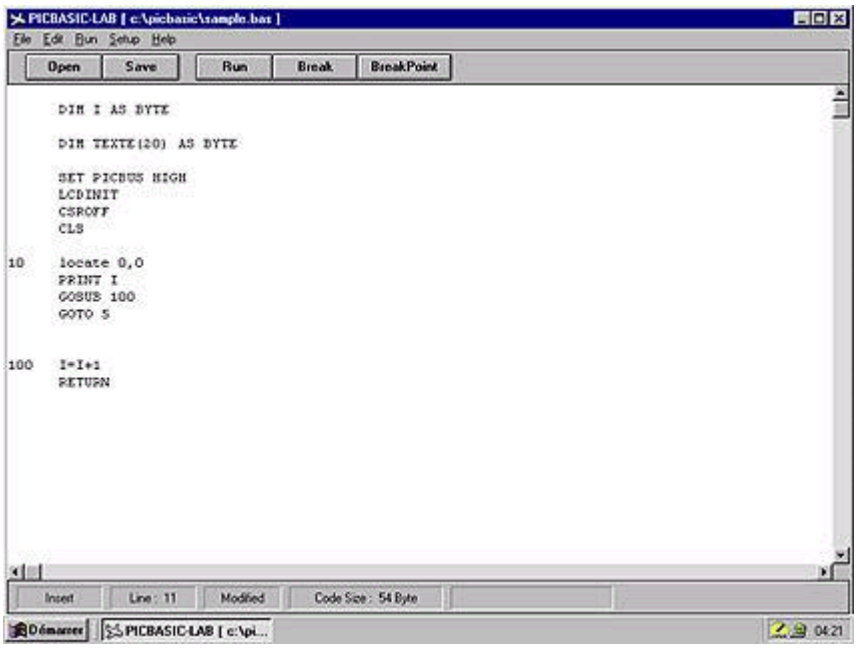
Vous trouverez ci-dessous un descriptif détaillé des 2 versions de logiciels.

Consultez la feuille livrée avec les PICBASIC pour prendre connaissance de la procédure d'installation des logiciels sur votre PC.

Le logiciel « PICBASIC-LAB »

Ce logiciel permet la programmation des "PICBASIC" série 1 / 2 / 3 (à savoir les PICBASIC-1B / 1S / 2S / 2H / 3 B / 3H) sous environnement Windows™ 98/Me/Se. il n'est **PAS COMPATIBLE** avec Windows 2000/XP™. Bien que doté d'une apparence "sobre" et "dépouillée" il n'en demeure pas moins un formidable outil de mise au point extrêmement performant, convivial et simple à maîtriser. Il devra bien évidémement être associé à un cordon de programmation spécifiques aux PICBASIC (raccordé au port imprimante de votre PC – Consultez le chapitre 4 pour plus d'infos).

L'éditeur du « PICBASIC-LAB »



Cet éditeur dispose de différents menus qui vous permettrons entre autre de charger et sauvegarder vos programmes sur le disque-dur du PC, d'effacer le contenu de la mémoire du module "PICBASIC" relié au PC, de vérifier la syntaxe de votre programme (en indiquant les lignes présentant une erreur), de configurer le N° du port imprimante utilisé et la vitesse de dialogue de ce dernier vis-à-vis des performances de votre PC, de visualiser le nom et la date de modification du programme stocké sur le PICBASIC relié au PC, de visualiser l'occupation de la mémoire "EEPROM" et "RAM" de votre programme, etc...

Description des différents "menus":

MENU « FILE »

"**New**" -> Création d'un nouveau programme (efface celui présent à l'écran).

"**Open**" -> Chargement d'un programme depuis le disque-dur du "PC" (également accessible par touche "F3" ou par "icône"sur l'écran de base).

"**Save**" -> Sauvegarde le programme à l'écran sur le disque-dur (également accessible par touche "F2" ou par "icône" sur l'écran de base).

"**Save As...**" -> Sauvegarde le programme à l'écran sur le disque-dur de votre "PC" en le nommant sous un autre nom que celui d'origine.

"**All Clear FLASH Program Memory**" -> Efface le contenu du"PICBASIC".

Lorsque vous sauvegardez plusieurs programmes, le nom des 4 derniers programmes utilisés apparaît au bas de ce menu. Ceci vous permet de les "rappeler" très rapidement.

"**Exit**" -> Permet de sortir du programme.

MENU « RUN »

"**Run**" -> Transfert le programme à l'écran dans le "PICBASIC" et exécute le programme (fonction également accessible par la touche "F5").

"**Break**" -> Ordonne au "PICBASIC" connecté au PC de stopper l'exécution de son programme et passe en mode "DEBUG" (voir ci-après) - fonction également accessible par la touche "F6").

"**Insert Break Point**" -> Insère dans l'éditeur de texte à l'endroit du curseur une instruction "BREAK".

"**View Object Code...**" -> Visualise dans une fenêtre le code "objet" généré par le programme BASIC (fonction également accessible par la touche "F8").

"**View Memory Usage Status...**" -> Visualise l'occupation de la mémoire EEPROM et de la mémoire RAM de votre programme.

"**View Dowload Profile...**" -> Visualise dans une fenêtre le code "objet" du code du "PICBASIC" connecté au "PC".

"**Read Flash Program Memory...**" -> Visualise le nom, la date de modification du programme chargé dans le "PICBASIC" connecté au PC.

"**Syntaxe Check...**" -> Permet de vérifier la syntaxe de votre programme (fonction également accessible par la touche "F9").

MENU « SETUP »

"**FONT**" -> Permet la modification des "fontes".

"**USE KSC5601**" -> Non utilisé.

"**COMMUNICATION STATUS**" -> Permet de connaitre l'état de la communication entre le "PC" et le module "PICBASIC".

"**PORT SETUP**" -> Permet de modifier les paramètres de communication entre le "PC" et le module "PICBASIC".

MENU « HELP »

"**ABOUT PICBASIC-LAB...**" -> Verifie la version logiciel de l'interpréteur du module "PICBASIC" connécté au PC.

"**UPGRADE HISTORY**" -> Retrace l'historique des mises à jours du logiciel.

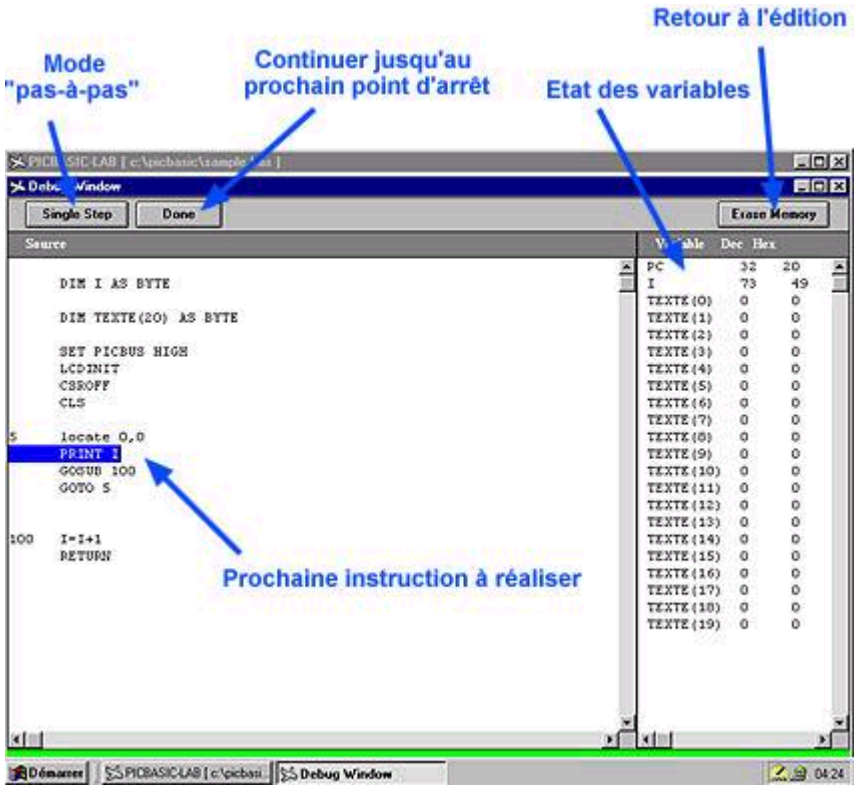
Commandes de l'éditeur de texte:

Les commandes de l'éditeur du logiciel "PICBASIC-LAB" s'apparentent à la plupart des éditeurs traditionnels. Elles sont accessible depuis le menu "Edit". On notera ci-dessous les commandes les plus intéressantes:

CTRL + Z -> Annule les dernières saisies un-à-un.
CTRL + X -> Efface le texte sélectionné par la souris (en vidéo-inverse) et le sauve dans le buffer interne de l'éditeur.
CTRL + V -> Recopie et affiche le texte mémorisé dans le buffer de l'éditeur.
CTRL + C -> Copie le texte sélectionné par la souris (en vidéo-inverse) et le sauve dans le buffer interne de l'éditeur.
F3 -> Chargement d'un programme depuis le disque-dur du PC.
F2 -> Sauvegarde du programme en cours sur le disque-dur du PC.

La fenêtre « DEBUG » du « PICBASIC-LAB »

Une fois votre programme écrit, il vous suffira (alors que le module PICBASIC est correctement alimenté par son application et relié au PC via le câble spécifique) de "télécharger" ce dernier en sollicitant simplement une icône adéquate (**Run**) du programme "PICBASIC-LAB". A ce stade, le module "PICBASIC" devient opérationnel et réalise les opérations de votre programme de façon autonome. Vous conserverez toutefois "la main" au niveau de l'éditeur du "PICBASIC-LAB" et vous pouvez modifier en même temps votre programme puis le télécharger à nouveau dans le module "PICBASIC" en sollicitant encore l'icône (**Run**).



Néanmoins à l'inverse de la plupart des produits concurrents, Il vous sera possible de disposer d'un très puissant système de "débuggage" en sollicitant simplement l'icône "**Break**" (ou la touche "**F6**"). A ce moment, le "PICBASIC" va interrompre son programme et s'arrêter sur la dernière instruction qu'il était en train d'exécuter. De même une nouvelle fenêtre va s'afficher sur l'écran de votre compatible "PC". Cette fenêtre appelée "**Debug Window**" est scindée en 2 parties.

La partie de gauche affiche votre programme où la prochaine instruction à réaliser apparaît en vidéo inverse (bleue).

La partie de droite affiche l'ensemble des variables utilisées dans votre programme avec les valeurs de chacune d'entres elles (en décimal et hexadécimal).

A ce stade, Vous allez pouvoir exécuter **une par une** les instructions de votre programme en visualisant les effets sur votre module "PICBASIC" et surtout l'évolution des valeurs de chaque variable de votre programme sur l'écran du "PC". Pour ce faire, cliquez à plusieurs reprises sur l'icône "**Single step**". Comme vous pouvez le constater, vous pouvez suivre très facilement l'évolution de votre programme grâce à la ligne de la prochaine instruction à réaliser qui s'affichera en vidéo inverse (bleue).

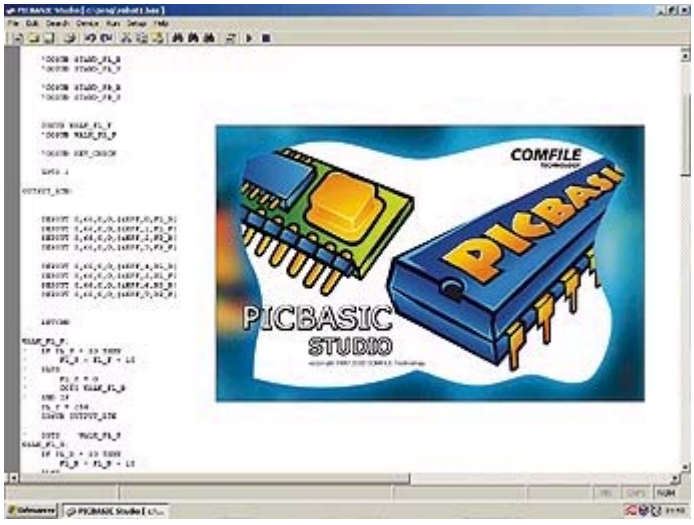
Une fois vos tests et essais effectués, cliquez sur l'icône "**Erase memory**" afin de revenir à la fenêtre de l'éditeur. A ce stade, le module "PICBASIC" est toujours "stoppé". Pour lui permettre d'effectuer à nouveau son programme il faut alors de nouveau cliquer sur l'icône "**Run**". Vous pouvez également en insérant une ou plusieurs instructions "**Break**" au sein de votre listing, demander au PICBASIC de stopper automatiquement l'exécution de son programme dès qu'il rencontre cette instruction et de vous afficher la fenêtre "**Debug Window**" sur le "PC". A ce stade, à chaque fois que vous cliquerez sur l'icône "**Done**", le programme reprendra le fil de son exécution pour s'arrêter à chaque fois après une instruction "**Break**". Vous pouvez également choisir à loisir d'exécuter le programme instruction par instruction (icône "**Single step**") ou revenir à l'éditeur (icône "**Erase Memory**").

Avec ces possibilités, le module "PICBASIC" ainsi relié au "PC" s'apparente à une véritable sonde d'émulation dont la puissance vous permettra de mettre très rapidement au point vos programmes. Il vous sera ainsi possible d'adjoindre de nombreuses instructions "**Break**" afin d'exécuter "normalement" les portions de programmes qui ne posent pas de problème et s'arrêter pour contrôler plus précisément les "passages" à "débuguer".

Configuration de base nécessaire au fonctionnement du "PICBASIC-LAB"

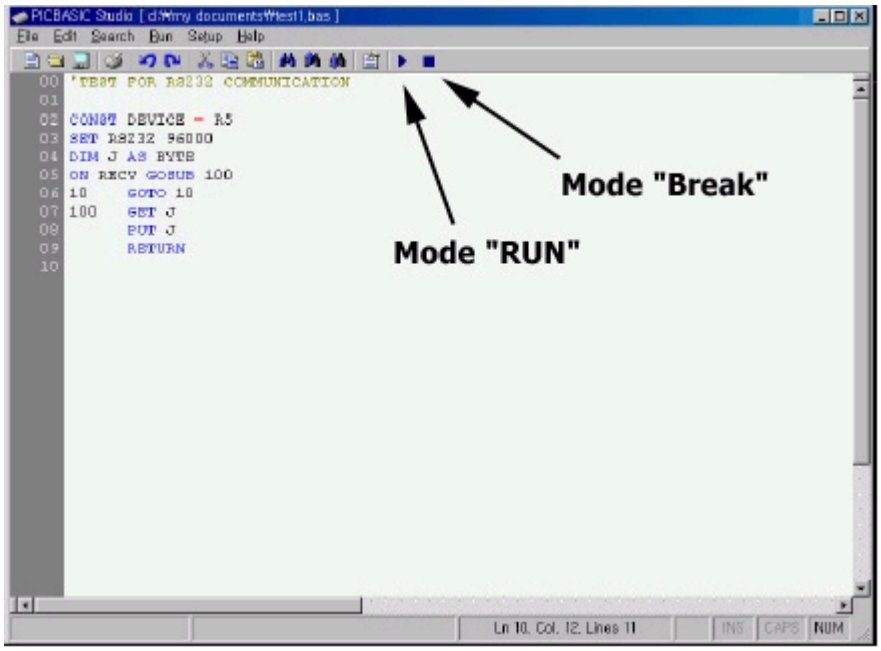
- Compatible PC (Pentium minimum).
- Windows™ 95/98/Me.
- RAM 16 Mo.
- 5 Mo de libre sur votre disque dur.
- Lecteur de CD-rom.
- Câble de programmation "noir moulé".

Le logiciel « PICBASIC-STUDIO »



Ce logiciel permet la programmation des PICBASIC-1B / 1S / 2S / 2H / 3 B / 3H / PBM-R1 / PBM-R5) sous environnement WindowsXP™. Il n'est **PAS COMPATIBLE** avec Windows 2000/98/Me/Se™. Ce dernier offre un formidable outil de mise au point extrêmement performant, convivial et simple à maîtriser. Il devra bien évidémement être associé à un cordon de programmation spécifiques aux PICBASIC (raccordé au port imprimante ou USB de votre PC – Consultez le chapitre 4 pour plus d'infos).

L'éditeur du « PICBASIC-STUDIO »

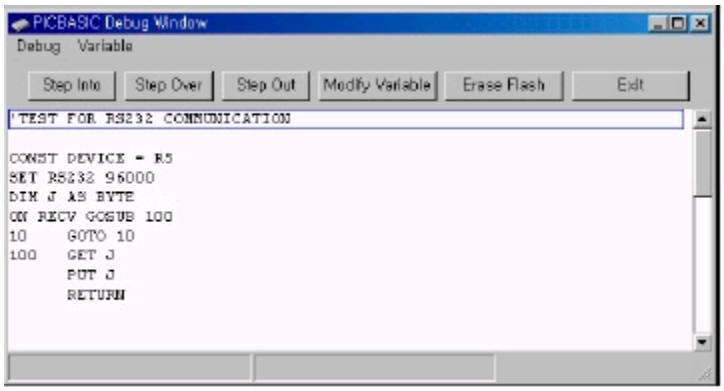


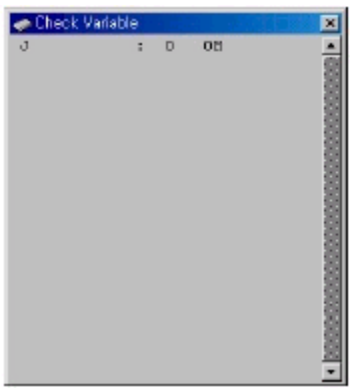
Cet éditeur dispose de différents menus qui vous permettrons entre autre de charger et sauvegarder vos programmes sur le disque-dur du PC, d'effacer le contenu de la mémoire du module "PICBASIC" relié au PC, de vérifier la syntaxe de votre programme (en indiquant les lignes présentant une erreur), de configurer le N° du port imprimante utilisé et la vitesse de dialogue de ce dernier vis-à-vis des performances de votre PC, de visualiser le nom et la date de modification du programme stocké sur le PICBASIC relié au PC, de visualiser l'occupation de la mémoire "EEPROM" et "RAM" de votre programme, etc...

Les menus du « PICBASIC-STUDIO » sont à peu de chose prêt similaire les mêmes que ceux du logiciel « PICBASIC-LAB » (voir description en page 29).

La fenêtre « DEBUG » du « PICBASIC-LAB »

Une fois votre programme écrit, il vous suffira (alors que le module PICBASIC est correctement alimenté par son application et relié au PC via le câble spécifique) de "télécharger" ce dernier en sollicitant simplement une icône adéquate (**Run**) du programme. A ce stade, le module "PICBASIC" devient opérationnel et réalise les opérations de votre programme de façon autonome. Vous conserverez toutefois "la main" au niveau de l'éditeur du "PICBASIC-STUDIO" et vous pouvez modifier en même temps votre programme puis le télécharger à nouveau dans le module "PICBASIC" en sollicitant encore l'icône (**Run**).





Néanmoins à l'inverse de la plupart des produits concurrents, Il vous sera possible de disposer d'un très puissant système de "débuggage" en sollicitant simplement l'icône "**Break**". A ce moment, le "PICBASIC" va interrompre son programme et s'arrêter sur la dernière instruction qu'il était en train d'exécuter. De même une nouvelle fenêtre va s'afficher sur l'écran de votre compatible "PC". Cette fenêtre appelée "**Debug Window**" est scindée en 2 parties.

La partie de gauche affiche votre programme où la prochaine instruction à réaliser apparaît en vidéo inverse (bleue).

La partie de droite affiche l'ensemble des variables utilisées dans votre programme avec les valeurs de chacune d'entres elles (en décimal et hexadécimal).

A ce stade, Vous allez pouvoir exécuter **une par une** les instructions de votre programme en visualisant les effets sur votre module "PICBASIC" et surtout l'évolution des valeurs de chaque variable de votre programme sur l'écran du "PC".

Suivant le modèle de "PICBASIC" connecté au PC (et déclaré à la première ligne de votre programme), vous disposez pour les PICBASIC-1B / 1S / 2B / 2S / 2H / 3B / 3H, des mêmes fonctions de debuggage du logiciel "PICBASIC-LAB".

Pour les PICBASIC2000 de la gamme « PBM » (PBM-R1 / PBM-R5) vous disposerez de fonctions de debuggage supplémentaires.

Une icône spéciale "**Modify Variable**" vous permet de modifier depuis le "PC" la valeur de toutes les variables du programme afin d'accélérer les phases de tests. Une fois la ou les variables modifiés, il vous sera possible de "relancer" à nouveau l'exécution temps réelle du programme du module PICBASIC2000 ou de tester à nouveau le programme en mode "pas-à-pas".
L'icône "**Step Over**" exécute également le programme en mode "**pas-à-pas**" à l'exception que lorsque le programme "arrive" sur une instruction "GOSUB", ce dernier réalise complètement la sous-routine en temps réel. Une fois la sous-routine terminée, le programme vous "redonne la main" après le retour de la sous-routine. Cet icône permet également d'accélérer les phases de développement en "sautant" les sous-routines dont vous êtes sûr du "bon" fonctionnement.

L'icône "**Step Out**" permet l'exécution du programme en temps réel jusqu'à ce que le "PICBASIC2000" rencontre la fin d'une sous-routine. A ce stade, ce dernier stoppe l'exécution de son programme et vous "redonne" la main. Ceci se reproduira pour toutes les sous-routines de votre programme.

Les icônes "**Exit**" permettent de sortie de la fenêtre "**Debug Window**" pour revenir à la fenêtre de l'éditeur et vous permettre ainsi de modifier votre programme.

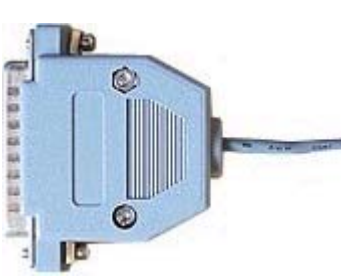
Vous pouvez également en insérant une ou plusieurs instructions "**Break**" au sein de votre listing, demander au PICBASIC de stopper automatiquement l'exécution de son programme dès qu'il rencontre cette instruction et de vous afficher la fenêtre "**Debug Window**" sur le "PC".

Configuration de base nécessaire au fonctionnement du "PICBASIC-STUDIO"
- Compatible PC (Pentium minimum).
- WindowsXP™.
- RAM 16 Mo.
- 8 Mo de libre sur votre disque dur.
- Lecteur de CD-rom.
- Câble de programmation "imprimante" ou "USB".

Chapitre 4.

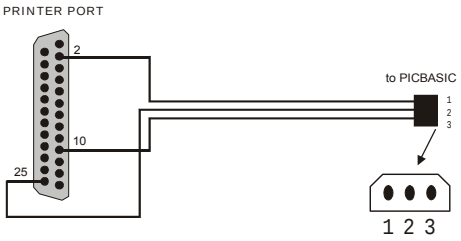
Les câbles de téléchargement

Si vous travaillez sous « Windows98™ » et que vous utilisez un PICBASIC-1B / 1S / 2S / 2H / 3B ou 3H



Votre câble de programmation doit **IMPERATIVEMENT** être identique (forme et couleur) à un des 2 modèles ci-dessus. Ces 2 modèles sont identiques (le "gris" correspond aux toutes premières versions livrées il y a plusieurs années de cela - le câble noir entièrement « moulé » correspond aux modèles actuellement livrés) – Ces câbles ne fonctionnent **QU'AVEC** utiliser le logiciel "PICBASIC-LAB V 3.9A". Vérifiez également les paramètres de configuration concernant la vitesse de votre PC et l'adresse du port imprimante dans le logiciel "PICBASIC-LAB" (voir le chapitre 3 : Les logiciels de programmation).

Ces versions de câbles ne renferment aucune électronique. Vous pourrez dés lors réaliser vous-même votre propre câble selon le schéma donné ci-dessous (nous commercialisons également le câble prêt à l'emploi).



Si vous travaillez sous « WindowsXP™ » et que vous voulez utiliser le port parallèle du PC pour programmer les PICBASIC-1B / 1S / 2S / 2H / 3B / 3H / PBM-R1 / PBM-R5

Votre câble de programmation doit être identique à celui ci-dessus - De plus, vous devez **IMPERATIVEMENT** utiliser le logiciel "PICBASIC-STUDIO" (version 1.6 minimum). Vérifiez également que le câble "REV.B" est sélectionné dans le menu "SETUP" (PC Interface SETUP) et que l'adresse du port imprimante est correcte. Ce câble contient une petite platine électronique. Il ne sera donc pas possible de le réaliser soit-même et il vous faudra l'acheter.

Pour utiliser ce dernier, vous devez **IMPERATIVEMENT** créer une imprimante de type "**HP LaserJet 4**" (même si vous ne disposez pas d'imprimante), sans quoi le PICBASIC risquera de n'être pas reconnu ou le mode "débug" ne fonctionne pas correctement.

Pour ce faire:

- Cliquez sur "Démarrer", puis "Paramètres", puis "Imprimantes et télécopieurs".
- Faite "Ajouter une imprimante", puis "Suivant".
- Cochez la case "Imprimante locale connectée à cet ordinateur".
- Décochez la case "Détection et installation automatique de l'imprimante Plug-and-Play".
- Puis "Suivant", sélectionnez "Utiliser le port suivant" [LPT1: port imprimante recommandée]
- Puis encore "Suivant", dans la fenêtre "Fabricant", sélectionnez "HP" puis dans la fenêtre "Imprimantes", choisissez "HP LaserJet 4" (à peu près au milieu de la liste), puis suivant.
- Cochez "non" à "Voulez-vous utiliser cette imprimante par défaut ?"
- Inutile d'imprimer la page de test puis cliquez sur "Terminer".

Note sur les câbles parallèles

Si vous utilisez **un PC portable** pour programmer les "PICBASIC", il se peut dans certains cas que le câble parallèle des PICBASIC ne fonctionne pas correctement (le programme vous indique que le PICBASIC n'est pas détecté ou que le câble a un problème). Ce phénomène est généralement du à la conception interne de l'interface parallèle des portables. Dans d'autres cas, il se peut que la masse ne soit pas présente sur la connexion 25 de la Sub-D (recherchez alors la présence de la masse sur votre ordinateur et reconnectez-la sur la prise 25 de la Sub-D - Certains modèles disposent d'une masse sur la prise 24 - à vérifiez selon votre modèle).

Dans tous les cas, si vous utilisez un PC portable pour programmer vos PICBASIC, il est conseillé d'avoir recours au câble de programmation USB décrit ci-après.

Si vous désirez programmer les PICBASIC à partir de PC connectés en réseaux, il vous faut choisir le câble USB ci-dessous et ne pas utiliser le câble parallèle qui n'est pas prévu pour ce type de configuration.

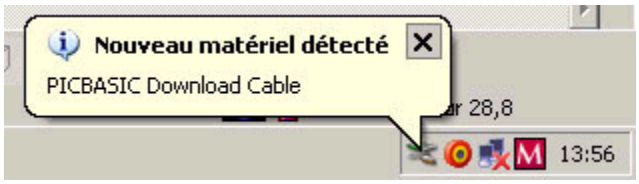
Si vous travaillez sous « WindowsXP™ » et que vous voulez utiliser le port USB du PC pour programmer les PICBASIC-2H / 3B / 3H / PBM-R1 / PBM-R5

Votre câble de programmation doit être identique à celui ci-dessus - De plus, vous devez **IMPERATIVEMENT** utiliser le logiciel "PICBASIC-STUDIO" (version 1.6 minimum). Vérifiez également que le câble USB est sélectionné dans le menu "SETUP" (PC Interface SETUP) et que vous avez installé le drivers USB comme indiqué dans la procédure ci-après.

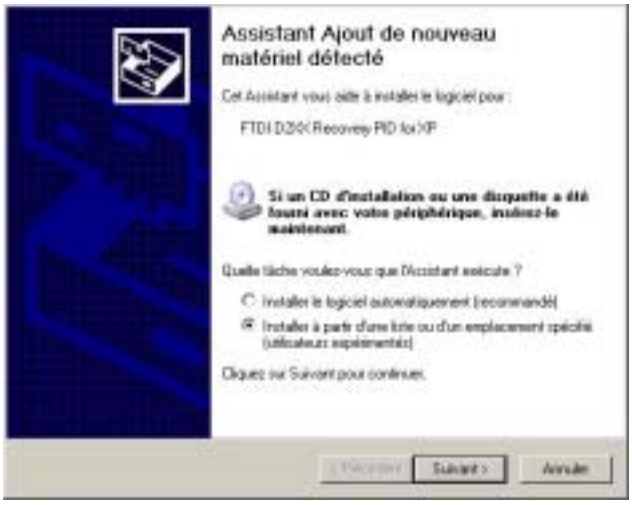
Installation du driver du câble USB

Une fois le driver téléchargé, décompactez **uniquement** les fichiers dans le répertoire de votre choix (par exemple sous C:\Drp10504). **ATTENTION, NE PAS OUVRIR LES FICHIERS FTD2XXUN.EXE OU FTXPRCVR.exe**

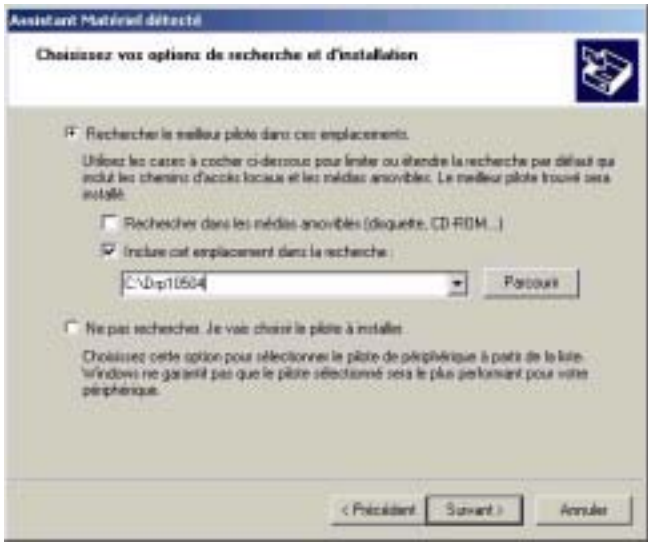
Connectez votre cordon, Windows XP™ va le détecter (une notification va apparaître en bas à droite comme indiqué ci-dessous).



Suivez alors les étapes suivantes:



Cocher "Installation à partir d'une liste ou d'un emplacement spécifié" puis cliquer sur "**Suivant >**".



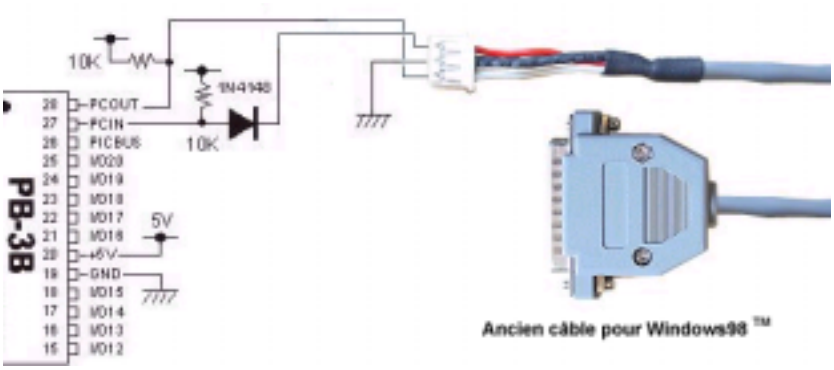
Cochez la case "inclure cet emplacement dans la recherche:", puis cliquez sur "**Parcourir**" et indiquer l'emplacement où vous avez décompacté le fichier Drp10504). Cliquez sur "**Suivant >**".

Si l'emplacement indiqué est correct, Windows va installer les fichiers nécessaires. Cliquez sur "**Terminer**" pour finaliser l'installation.

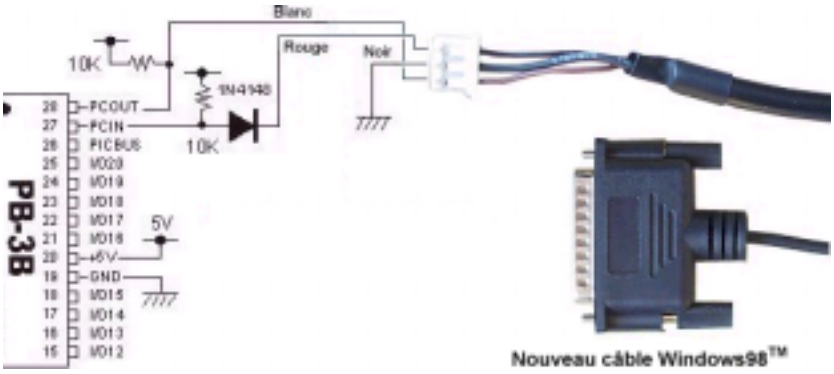
Une notification va alors apparaître en bas à droite durant quelques instant vous indiquant que votre cordon est prêt à l'emploi.
Il ne vous reste plus qu'à installer le logiciel PICBASIC Studio:
Ensuite allez dans le menu "SETUP" puis "PC Interface SETUP" et vérifiez que "USB" soit bien coché

Raccordement des câbles sur les PICBASIC-3B/3H

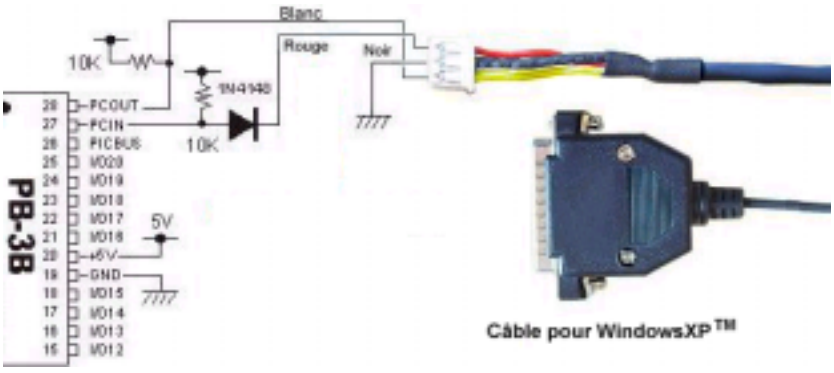
Les schémas ci-dessus donnent les raccordements des différents câbles avec les PICBASIC-3B et PICBASIC-3H.



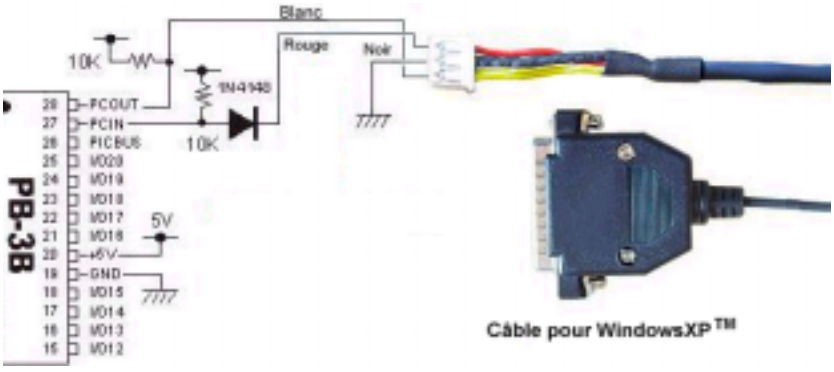
Ancien câble pour Windows98 ™



Nouveau câble Windows98™



Câble pour WindowsXP ™



Câble pour WindowsXP ™

Chapitre 5.

Les instructions des PICBASIC

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)
- ***Long*** : Nombre sur 32 bits signé (-2,147,483,648~2,147,483,647)
- ***Single*** : Nombre décimal 32 bits à virgule avec signe
- ***String*** : Chaîne de caractères 8 bits (92 octets max.)

Ainsi avec les PICBASIC de la série « PB » (PICBASIC-1B / 1S/ 2S / 2H/ 3 B/ 3H) il ne sera possible que d'utiliser 2 types de variables (BYTE et INTEGER).

Avec les PICBASIC de la série « PBM » (PBM-R1 et PBM-R5) vous pourrez utiliser tous les types de variables.

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.

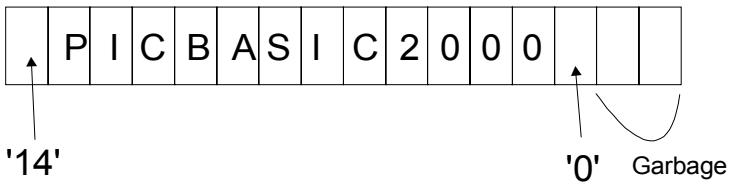
- Noms de variables pouvant être utilisés: A, B0, I, J, TH, BF1
- Noms de variables NE pouvant PAS être utilisés: 23, 3A, INPUT, GOTO

>>> 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

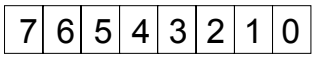
```
CONST DEVICE = 3H        ' Ceci signifiera que votre porgramme sera utilisé sur un PICBASIC-3H
```

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

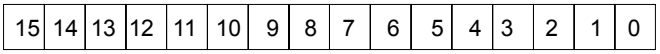
Acces direct aux « Bits »

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.

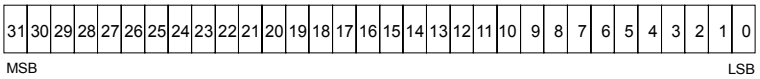
BYTE



INTEGER



LONG



PICBASIC de la série « PBM »

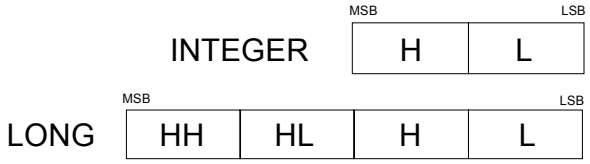
```
DIM I AS BYTE, A AS BYTE
I:7 = 0        ' force 7th bit de la variable Byte I à 0
I:7 = A:2      ' Transmet le 2nd bit de la variable A vers le 7th bit de la variable I
```

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™.

ABS()

PBM

Variable Single= ABS (*Valeur*)

Valeur absolue

Valeur est une variable de type Single.

EXPLICATION

Cette instruction permet de connaitre la valeur absolue d'une variable de type "SINGLE".

EXEMPLE:

```
10        DIM F1 AS SINGLE
20        DIM F2 AS SINGLE
30         F1=ABS(F2)              ' SI F2 contient la valeur -12345,6789 alors F1 contiendra la valeur 1234.5678"
```

Si vous désirez obtenir la valeur absolue d'une variable de type "LONG", il faudra au préalable la convertir dans une variable de type "SINGLE" à l'aide de l'instruction "CSNG".

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.

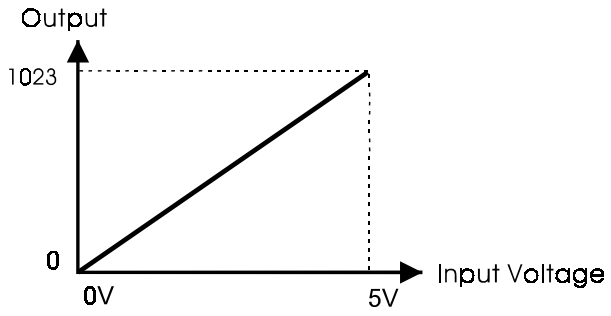
Avec les PICBASIC-1S / 2S / 2H les convertisseurs bénéficient d'une résolution sur 8 bits:
Pour 0 V en entrée -> on obtient le nombre 0
Pour 2.5 V en entrée -> on obtient le nombre 128
Pour 5.0 V en entrée -> on obtient le nombre 255.

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 »**



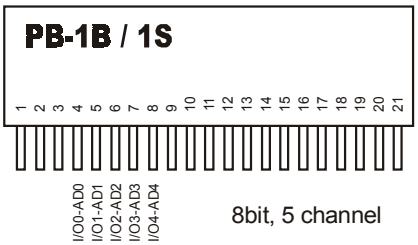
INFORMATIONS COMPLEMENTAIRES

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.

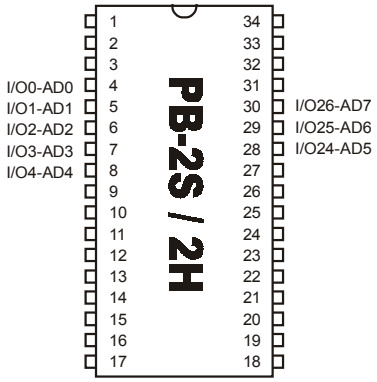
```
DIM K AS INTEGER         ' Déclaration de la variable K en Integer (16 bits)
K = ADIN(32)             '  Récupère la valeur de la conversion sur 12 bits dans la variable K
```

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).

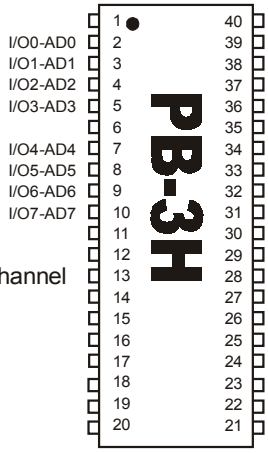
Rappel sur l'emplacement des ports de conversion « A/N » des PICBASIC.



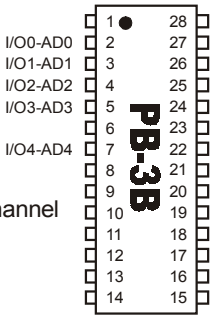
8bit, 5 channel



8bit, 8 channel

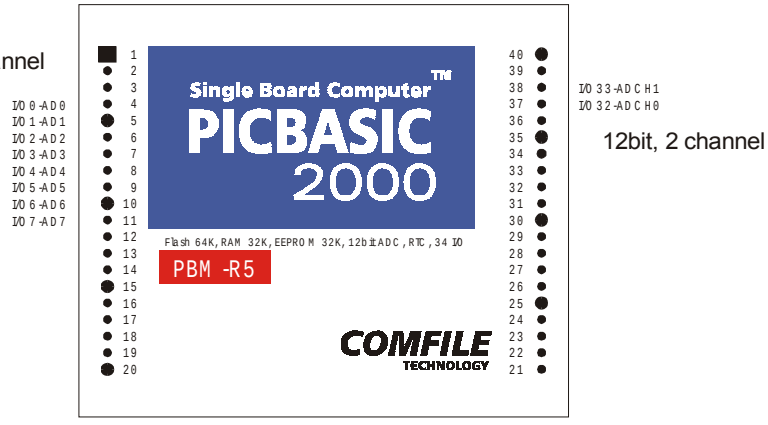


10bit, 8 channel



10bit, 5 channel

10bit, 8 channel



12bit, 2 channel

ADKEYIN ()

Variable Byte = ADKEYIN (*port*)

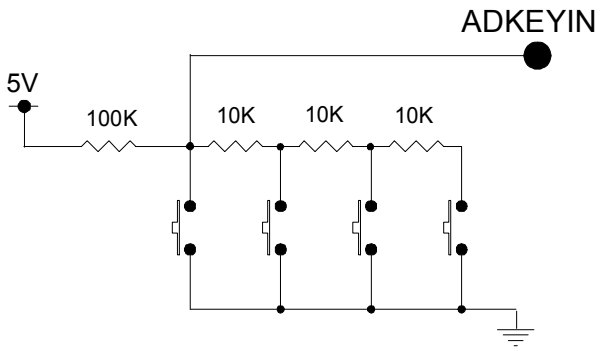
Gestion de touches via entrées de conversion « A/N »

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.

ASC ()

PBM

Variable Byte = ASC (*caractere*)

Code ASCII d'un caractère

Caractere est un caractère.

EXPLICATION

Cette instruction permet de connaître la valeur "ASCII" d'une variable (**caractere**) représentant un caractère.

EXEMPLE

```
DIM ST AS STRING * 16      ' Défini une chaîne de caractères de 16 octets de long
DIM I AS BYTE
ST = "A"
I = ASC(ST)                ' &H41 est stocké dans la variable I (ce qui correspond au code ASCII de la lettre "A").
```

Le tableau ci-dessous donne la correspondance entre les caractères et leur code "ASCII".

4 bits poids fort	4 bits de poids faible															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	B	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	R	s	t	u	v	w	x	y	z	{	|	}		

Lisez en premier les 4 bits de poids fort, puis les 4 bits de poids faible. Par exemple la lettre "A" donne &H41 et "W" donne &H57.

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 « **B**inaire-**C**odé-**D**écimal ».

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

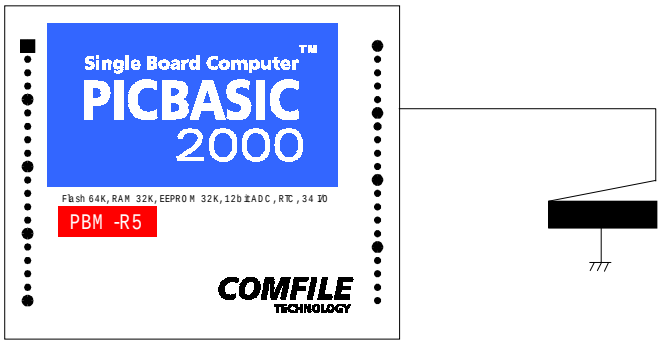
BEEP *port*

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
```

BCLR

PBM

BCLR

Efface buffer RS232

EXPLICATION

Cette instruction permet d'effacer totalement le contenu du buffer "**RS-232 matériel**" des « PBM-R1 / PBM-R5 ».

A noter que le buffer est également automatiquement effacé à la mise sous tension des « PICBASIC ».

Après l'exécution de l'instruction SET RS232C, les données séries réceptionnées sur la broche RX (I/O15) du « PICBASIC » sont présentent dans le buffer matériel RS232C. L'instruction BCLR efface totalement le contenu de ce buffer.

BLEN (0)

PBM

Variable Integer = BLEN (*0*)

Number of data at RS232C

EXPLICATION

Permet de connaître le nombre de données stockées dans le buffer matériel “RS-232"

EXEMPLE

```
      DIM I AS BYTE, J AS BYTE, K AS BYTE
      SET RS232 9600                    ' Configure le buffer matériel RS232C à 9600 bauds
      ON RECV GOSUB 100                 ' Configure interruption du port série matériel
      OUT 19,0
10    PULSE 19,100
      DELAY 100
      IF KEYIN(9) = 0 THEN              ' Efface buffer RS232C lorsque l’entrée 9 est sollicitée
        BCLR
      END IF
      GOTO 10

100   K = BLEN(0)                       ' Renvoi un « ECHO »  du nombre de données du buffer
      FOR J = 1 TO K
            GET I
            PUT I
      NEXT
      RETURN
```

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*)

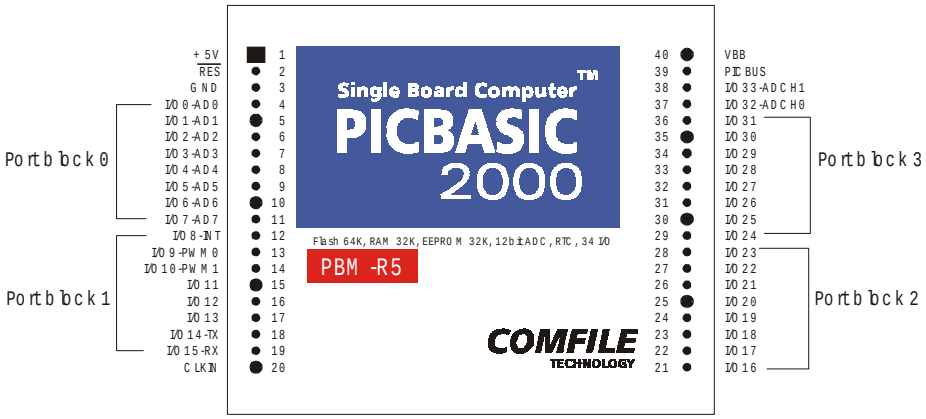
Gestion d'entrées sur 8 bits

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*

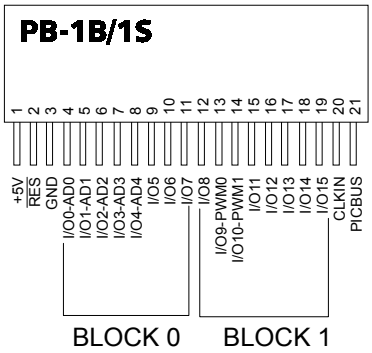
Sortie d'une valeur sur 8 bits

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

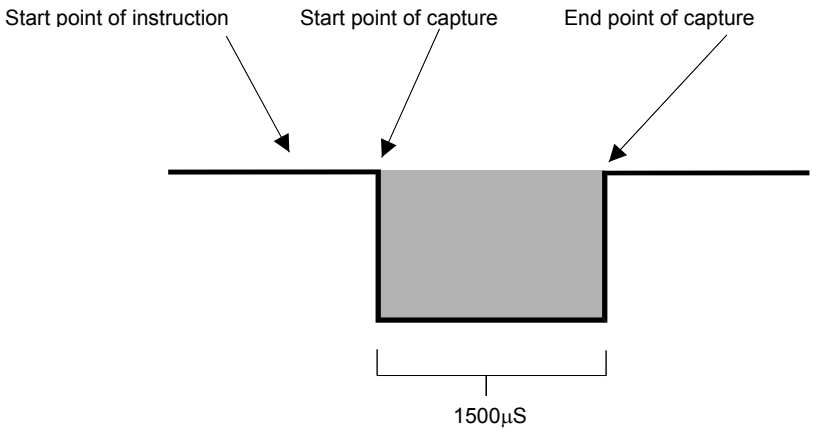
Port est le port de la mesure pouvant être défini par une constante (0~3) ou une variable de type Byte.
Target est une contante (0 ou 1 seulement) – Ce paramètre ne peut pas être utilisé avec une variable.

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.

CHR ()

PBM

Variable chaine = CHR (*Valeur*)

Convertion code ASCII en un caractère

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

EXPLICATION

Cette instruction converti un code ASCII en un caractère (image de sa représentation).

Par exemple CHR(&H41) retournera le caractère “A”.

EXEMPLE

```
DIM ST AS STRING * 16          ‘ Défini une chaine de caractères de 16 octets
ST = CHR(&H41)                 ‘ ”A” sera stocké dans la variable ST.
```

Le tableau ci-dessous donne la correspondance entre les caractères et leur code “ASCII”.

4 bits poids fort \ 4 bits de poids faible	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2		!	“	#	$	%	&	‘	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	B	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	R	s	t	u	v	w	x	y	z	{	|	}		

Lisez en premier les 4 bits de poids fort, puis les 4 bits de poids faible. Par exemple la lettre “A” donne &H41 et “W” donne &H57.

CINT ()

PBM

Variable Integer = CINT (*Valeur*)

Convertion Single en Integer

Valeur est une variable de type Single.

EXPLICATION

Cette instruction permet de convertir la valeur d'une variable définie en "SINGLE" afin de pouvoir la transférer dans une variable de type "INTEGER" (au passage, les chiffres après la virgule disparaissent).

EXEMPLE

```
10        DIM  ST  AS  STRING * 16
20        DIM  I     AS  SINGLE
30        DIM  L1  AS  INTEGER
40        ST  =  "1234.5678"
50        I    =  VALSNG(ST)              ' Charge la valeur dans la variable de type "SINGLE".
60        L1  =  CINT(I)                  ' la variable “L1” est égale à "1234".
```

CLNG ()

PBM

Variable Long = CLNG (Valeur)

Converttion Single en Long

Valeur est une variable de type Single.

EXPLICATION

Cette instruction permet de convertir la valeur d'une variable définie en "SINGLE" afin de pouvoir la transférer dans une variable de type "LONG" (au passage, les chiffres après la virgule disparaissent).

EXEMPLE

```
10                                                          DIM  ST  AS  STRING * 16
20                                                            DIM  I    AS  SINGLE
30                                                              DIM  L1  AS  LONG
40                                                               ST  =  "1234.5678"
50                        I    =  VALSNG(ST)              ' Charge la valeur dans la variable de type "SINGLE".
60          L1  =  CLNG(I)                    ' la variable “L1” est égale à "1234".
```

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".

COS ()

PBM

Variable Single= COS (*Valeur*)

Calcul valeur COSINUS

Valeur est une variable de type Single.

EXPLICATION

Cette instruction permet de calculer le cosinus d'une variable (préalablement définie en tant que "SINGLE").

EXEMPLE

```
10        DIM   F1   AS   SINGLE
10        DIM   F2   AS   SINGLE
30        F1   =   COS(F2)                    ' F1 contient la valeur du cosinus de F2.
```

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.

CSNG ()

PBM

Variable Single= CSNG (*Valeur*)

Conversion Long en Single

Valeur est une variable de type Long.

EXPLICATION

Cette instruction permet de convertir la valeur d'une variable définie en "LONG" afin de pouvoir la transférer dans une variable de type "SINGLE". Elle doit aussi être utilisée en cas de comparaison entre des variables de type "LONG" et "SINGLE".

EXEMPLE

```
10        DIM   L1   AS   LONG
20        DIM   S1   AS   SINGLE
30        DIM   S2   AS   SINGLE
40        S2    =    S1 * CSNG(L1)          ' Ne pas écrire simplement   S2    =   S1  * L1
50        IF   S2   <   CSNG(L1)             ' Ne pas écrire simplement    IF   S2   <   L1.
```

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.

DACOUNT

PBM

DACOUT *port*, *Val 1*, *Val 2*

Génération tension analogique

Port est une constante comprise (0~31) ou une variable de type Byte
Val 1 est une constante (0~255) ou une variable de type Byte représentant le rapport cyclique du signal généré.
Val 2 est une constante (0~65525) ou une variable de type Byte/Integer représentant le nombre de cycle généré.

EXPLICATION

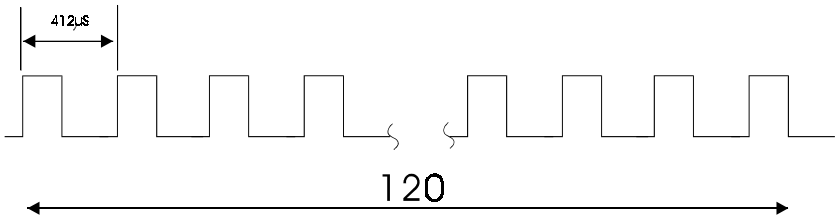
Cette instruction est principalement destinée à créer une tension analogique "temporaire" (en association avec un petit montage externe) grâce à la génération d'un signal de type "PWM" d'une fréquence de l'ordre de 2,4 KHz dont le rapport cyclique est fonction de (Val1) et la durée est fonction de (Val2) sur la broche (Port) du "PICBASIC2000". Val1 peut prendre une valeur comprise entre 0 et 255, Val2 entre 1 et 65535.

Le tableau ci-contre donne une comparaison technique entre les possibilités de l'instruction DACOUT et PWM. Comme on le voit, la génération des signaux PWM est attribuée à des ports particuliers et figés alors qu'il est possible de reconfigurer les broches pouvant être utilisées avec l'instruction DACOUT.

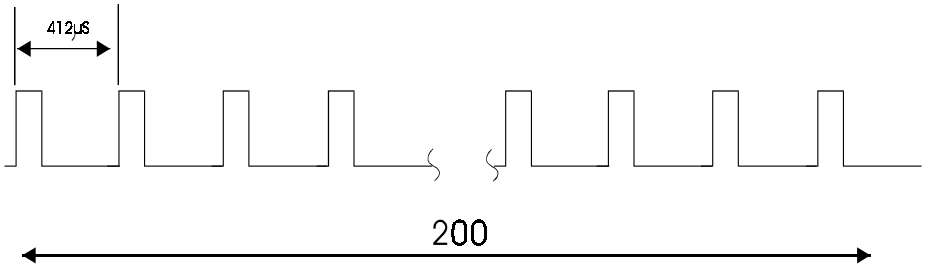
	DACOUT	PWM
PORTs utilisables	0 ~ 15	9, 10
Méthode de génération	Par logiciel	Matérielle
Fréquence	Fixe 2.4 KHz env.	variable
Continuité du signal	Non	Arrêt possible via PWMOFF

EXEMPLE

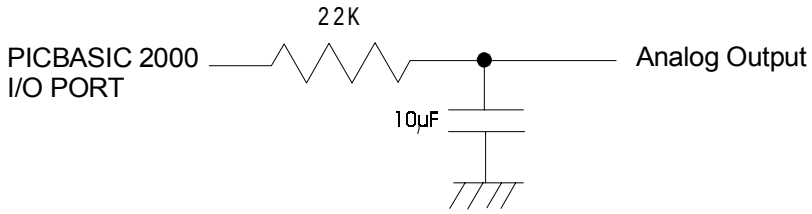
DACOUT 5,100,120



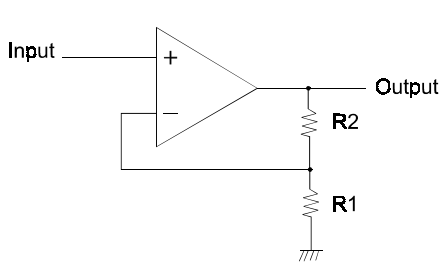
DACOUT 5,10,200



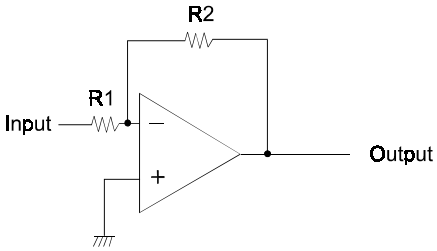
Pour créer une tension analogique (non chargée en sortie), on pourra avoir recour au schéma ci-dessous.



Le schéma ci-dessous devra être utilisé en cas de "charge" sur la sortie analogique ainsi générée.



Noninverting amplifier circuit



Inverting amplifier circuit

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.
```

DEC ()

PBM

DEC (*Var*, *param1, param2*)

Retourne la valeur décimale dans une chaîne

Var est une variable de type Integer
Param1 est la taille de la chaîne (jusqu'à 10 max. – valeur par défaut : 5)
Param2 est permet suivant sa valeur (0 ou 1) d'afficher des "0" dans la caîne devant le nombre converti – valeur par défaut : 1.)

EXPLICATION

Cette instruction retourne la valeur décimale d'un nombre dans une chaîne selon un certain format (déterminé par "Param1" et "Param2". Son utilisation est identique à celle expliquée pour l'instruction "PRINT DEC" des "PICBASIC 1B-1S-2S-2H" (voir ci-après dans le document) mise à part qu'elle peut désormais s'utiliser en étant ou non associée à "PRINT". D'autre part, il est également possible d'avoir recours à cette instruction pour des variables de type "BYTE", "INTEGER" ou "LONG" et avec un nombre de caractères réservés qui peut aller jusqu'à 10 (contre 5 seulement sur les "PICBASIC 1B-1S-2S-2H".

EXEMPLE

```
10    DIM  F1  AS  STRING * 16
20    DIM  I  AS  INTEGER
30    I  =  123
40    F1 = DEC(I,10,0) + "456"                ' F1 contient la chaîne "0000000123456"
```

EEREAD ()

Variable Integer = EEREAD (*adr*, *Param*) PBM

Variable Integer = EEREAD (*adr*)

Lecture de données depuis la mémoire EEPROM

Adr est une constante/variable de type Byte/Integer (0~&HFFFF)
Param est une constante/variable (0~255) de type Byte et permet avec les PICBASIC PBM de définir combien d'octets on veut lire.

EXPLICATION
Cette instruction permet de récupérer une donnée à l'adresse (Adr) depuis la mémoire EEPROM du "PICBASIC".

EXEMPLE 1:

```
10        DIM  DONNEE  AS  BYTE
20        DONNEE  =  EEREAD(&HFFF)          ' Lit la valeur mémorisée à la dernière adresse d'un "PICBASIC-1S".
```

EXEMPLE 2:

```
10        DIM  ADRESSE  AS  INTEGER
20        DIM  DONNEE  AS  BYTE
30        ADR  =  &FFE
40        DONNEE  =  EEREAD(ADR)
```

A noter que pour des besoins particuliers, il vous est également possible de relire les codes correspondant à votre programme en partie basse de l'EEPROM.

Pour les PICBASIC de type « PBM », il est possible avec le paramètre **Param** de définir le nombre d'octets à lire.

EXEMPLE 3:

```
10          DIM  I  AS  BYTE
20          DIM  J  AS  INTEGER
30          DIM  K  AS  STRING * 16

40          I  =  EEREAD (&FFF)           ' Lit l'octet à l'adresse &FFF (Si il n'y a qu'un seul octet, Param n'est pas utilisé).

50          J  =  EEREAD (&F00,2)         ' Lit 2 octets à partir de l'adresse &F00

60          K  =  EEREAD (&F00,18)        ' Lit 18 octets à partir de l'adresse &F00 et les ranges dans la variable 'K'
                                            N'oubliez pas, qu'il est nécessaire d'allouer
                                            2 octets supplémentaires aux données de type "STRING".
```

NOTE
Pour les PICBASIC de type « PICBASIC-3B / 3H », la plage mémoire de l'instruction **eeread** est comprise entre F00 à FFF.

EEWRITE

EEWRITE *adr*, *Val*, *Param*

EEWRITE *adr*, *data*

PBM

Ecriture de données en EEPROM

Adr est une constante/variable de type Byte/Integer (0~&HFFFF).
Val est une constante (0~255) ou une variable de type Byte.
Param est une constante/variable (0~255) de type Byte, permettant avec les « PBM » de définir combien d'octets on veut écrire.

EXPLICATION

Cette instruction permet d'enregistrer des données dans la mémoire EEPROM du "PICBASIC". Cette mémoire non volatile (même en cas de coupure d'alimentation) peut être modifiée plus de 1000 fois. Il vous sera ainsi possible par exemple de sauvegarder des données de configuration qui pourront être relues avec l'instruction "EEREAD". L'instruction permet de mémoriser une donnée (ou plusieurs données avec les « PBM » si vous utilisez une valeur pour Param) dans la mémoire EEPROM du PICBASIC.

EXEMPLE

```
DIM I AS BYTE
DIM ST AS STRING * 16
EEWRITE &HFFF, I          ' Enregistre la valeur de la variable I à l'adresse "&HFFF" de la mémoire EEPROM
EEWRITE &HF00, ST, 18     ' Enregistre le contenu de la variable ST à l'adresse éHF00 de la mémoire EEPROM
```

INFORMATIONS COMPLEMENTAIRES CONCERNANT EEREAD & EEWRITE

La mémoire EEPROM (ou Flash) des PICBASIC est utilisée à la base pour stocker le programme du PICBASIC. Si vous n'utilisez pas tout l'espace mémoire pour votre programme, vous pourrez exploiter cet espace libre pour venir y mémoriser/lire des données à l'aide des instructionS eewrite/eeread. Toutefois, dans la majorité des cas, la mémoire EEPROM sera entièrement utilisée pour le stockage du programme du PICBASIC. La mémorisation du programme du PICBASIC s'effectue à partir de la partie basse de la mémoire EEPROM. Il conviendra donc d'utiliser la partie haute de la mémoire pour venir y stocker les données en prenant soin de partir de la dernière adresse et de « redescendre » vers les adresses basses.

```
EEWRITE &HFFF, R1
EEWRITE &HFFE, R2
EEWRITE &HFFD, R3

R1 = EEREAD(&HFFF)
R2 = EEREAD(&HFFE)
R3 = EEREAD(&HFFD)
```

Par exemple, pour le PICBASIC 1-S on utilisera l'espace mémoire EEPROM comme suit :

Il est donc impératif avant d'utiliser l'instruction eewrite de vérifier que les adresses liées au stockage de vos données ne viennent pas « entrer » dans l'espace mémoire occupé par la mémorisation de votre programme sans quoi des situations de dysfonctionnements imprédictibles pourront survenir sur le PICBASIC. Pour éviter ceci, vérifiez la taille de votre programme dans la fenêtre du logiciel du PICBASIC sur votre PC "Code Size : XXX byte" après l'opération de téléchargement.

Le tableau ci-dessous donne un rappel de l'espace mémoire des différents PICBASIC

PB-1B	PB-1S	PB-2S	PB-2H
0~&H7FF (2K)	0~&HFFF (4K)	0~&H1FFF (8K)	0~&H3FFF (16K)

Pour les « PBM », le problème ne se pose pas car ces derniers disposent d'une mémoire EEPROM externe dédiée aux instructions EEWRITE et EEREAD (vous pouvez donc commencer par écrire à partir de l'adresse 0 pour ces derniers).

Taille EEPROM externe sur série « PBM »

PBM-R1	PBM-R5
0 ~ &H1FFF (8K)	0 ~ &H7FFF (32K)

```
EEWRITE &H0, R1
EEWRITE &H1, R2
EEWRITE &H2, R3
```

NOTES

Pour les PICBASIC de type « PICBASIC-3B / 3H », la plage mémoire de l'instruction **eewrite** est comprise entre F00 à FFF.

Il est également impératif d'éviter de mettre une instruction **eewrite** au sein d'une boucle sans fin afin d'éviter de dépasser la valeur limite de ré-écriturede la mémoire EEprom (qui est de l'ordre de 1000 fois – sans quoi celle-ci serait altérée).

EPADIN ()

PBM

Variable Integer = EPADIN (*param1*, *param2*)

Entrée Clavier

Param1 représente le nombre de lignes (de 4 à 8)
Param2 représente le nombre de colonnes (de 4 à 8)

EXPLICATION

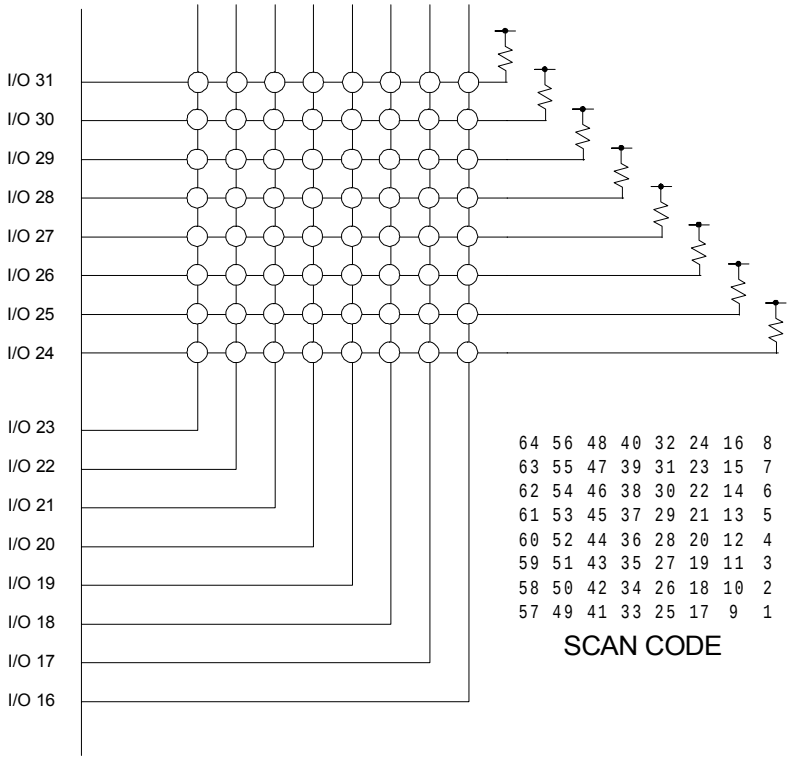
Cette instruction permet de gérer automatiquement un clavier de 64 touches max. de type matriciel. Les "colonnes" de ce dernier seront suivant le nombre (indiqué par Param2), reliées de "I/O 16 à I/O 23" tandis que les "lignes" de celui-ci seront suivant le nombre (indiqué par Param1), reliées de "I/O 24 à I/O 31". Le schéma ci-dessous donne un exemple de câblage pour un clavier 64 touches.

Dès lors, en effectuant cette instruction, le module effectuera automatiquement un "scanning" du nombre de touches déclarées 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 1:

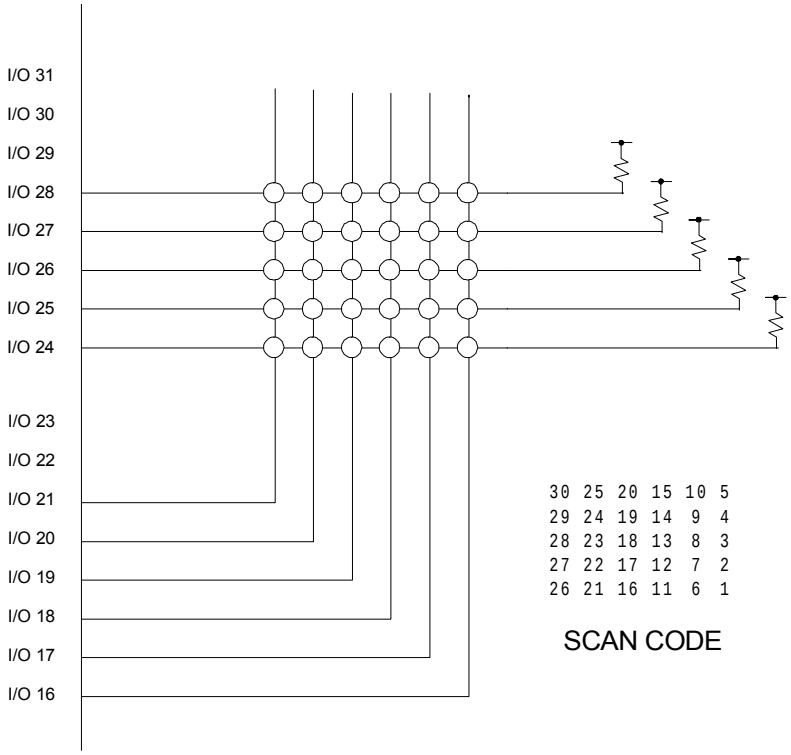
```
10        DIM  I  AS  BYTE
20        I=EPADIN(8,8)
```

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



EXEMPLE 2:
Si vous utilisez un clavier inférieu à 8 x 8, vous devrez ajuster les lignes et les colonnes comme ci-dessous :

Cas d'un clavier matricé 6 X 5 touches




```
I = EPADIN(6,5)              ' Lecture du clavier 6 X 5
LOCATE 0,0
```

NOTE
L'allocation des ports devra être effectuée par ordre numérique. L'allocation des ports de sortie devra débuter à partir du port 16 et l'allocation des ports d'entrée devra démarrer à partir du port 24. Si vous n'utilisez pas tous les ports, il vous sera possible d'utiliser ces derniers pour une autre fonction (mais uniquement en les exploitant en tant qu'entrées – Ne les utilisez jamais en sorties, sans quoi des changements d'états aléatoires pourront intervenir dessus).

EXP ()

PBM

Variable Single= EXP (***Valeur***)

Valeur naturelle logarithm e^x

Valeur est une variable de type Single.

EXPLICATION
Cette instruction permet de connaître la valeur exponentielle d'une variable de type "SINGLE".

EXEMPLE

```
10        DIM  F1  AS  SINGLE,  F2  AS  SINGLE
20        F2  =  EXP(F1).
```

FREQOUT

PB

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.

FLOAT ()

PBM

String variable = FLOAT (*Valeur*, *param1*, *param2*)

Instruction de conversion

Valeur est une variable de type Single.
Param1 est le nombre max. de digit à afficher avant la virgule (valeur par défaut 9)
Param2 est le nombre max. de digit à afficher après la virgule (valeur par défaut 6)

EXPLICATION

Cette instruction permet d'afficher sur un écran LCD à commande série la valeur d'une variable (Var) de type "SINGLE" selon plusieurs possibilités possibles. Param1 permet de sélectionner le nombre de "digits" affichables pour les chiffres avant la virgule, tandis que Param2 sélectionne le nombre de chiffres après la virgule.

EXEMPLE

```
10      PRINT   FLOAT(SN,4,3)          ' Si la variable SN = 1234.567 alors l'affichage sera: 1234.567
20      PRINT   FLOAT(SN,2,2)          ' Si la variable SN = 34.567 alors l'affichage sera: 34.56
```

NOTE

L'utilisation de l'instruction FLOAT peut provoquer un problème d'affichage avec certaines valeurs. Par exemple pour 4.0, FLOAT pourra retourner 3.999999. Ceci est du au fait que le microcontrôleur utilisé ne dispose pas d'un système de conversion basé sur une base de type IEEE 745.

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
```

GET

PBM

GET *var1*, *var2*, *Adress*

Réception RS232C (mode matériel)

Var1 est une variable de type Byte permettant de recevoir les données du Buffer RS232C.
Var2 est une constante de type Integer permettant de définir le temps d'attente de la donnée en provenance du Buffer RS232C.
Adress est l'endroit où le programme doit continuer si aucune donnée n'est reçue après le temps défini par var2.

EXPLICATION

Cette instruction permet la gestion d'un port de communication "RS-232" en tâche de fond (uniquement sur les PICBASIC de la série « PBM »). Avant de pouvoir l'utiliser, il faudra impérativement définir la vitesse de communication du port "RS-232" en début de programme avec l'instruction "SET RS232". Ainsi, toute donnée série arrivant sur le port "I/O 15" sera automatiquement stockée dans un buffer spécifique, même si votre programme est en train d'accomplir d'autres tâches. A l'opposé, en utilisant l'instruction SERIN, le programme sera uniquement occupé à recevoir les données séries. En utilisant l'instruction GET, la première donnée présente dans le buffer est transférée dans (Var1) selon un mode FIFO(First in, First OUT), l'attente de disponibilité d'une donnée à lire dans le buffer se fera pendant (Var2) ms, après quoi le programme passera à la ligne suivante (si Adress n'est pas indiqué) ou à l'adresse (Adress) si indiquée dans l'instruction. Une fois que l'instruction "GET" a pu lire une donnée reçue dans le buffer, celle-ci est effacée du buffer et la donnée suivante est automatiquement "pointée" pour être lue à nouveau avec l'instruction "GET".

EXEMPLE:

```
10      SET  RS232  9600        ' Configuration du port RS232 en 9600 bds
20      DIM  I  AS  BYTE
30      GET  I,  100            ' Attente de donnée pendant 100 ms (en cas contraire, passe à la ligne suivante)

40      GET I                   ' Lecture d'une donnée (si aucune disponible, passe à la ligne suivante)

50      GET I, 100, ERR         ' Attente de donnée pendant 100 ms (en cas d'absence, passe à la ligne
                                ' du programme "pointée" par l'adresse "ERR".
```

INFORMATIONS COMPLEMENTAIRES

Les instructions "GET" et "PUT" utilisent les fonctionnalités "pseudo-multi-tâche" du port série RS-232 dédié (I/O 14 et I/O 15). Le buffer de réception dédié à cette fonction dispose de 96 octets. Vous devez utiliser l'instruction "GET" pour récupérer les données avant que le buffer ne soit "plein" (utilisez l'instruction "ON RECV ... GOSUB") pour ce faire. ATTENTION, l'instruction "GET" ne lit pas en temps réel les données arrivant sur le port série, mais se contente de récupérer les données stockées dans le buffer de réception du port RS-232. Pensez également à définir en premier lieu les paramètres du port RS232 à l'aide de l'instruction SET RS232 (avant de pouvoir utiliser "GET" et "PUT").

Pour finir, rappelez-vous que les niveaux logiques présents aux ports (I/O 14 et I/O 15) sont de 0 / 5 Vcc. Si vous devez raccorder le PICBASIC à un PC ou à tout autre dispositif doté d'une liaison RS232 « standard », il vous faudra utiliser un composant MAX-232 additionnel (lequel devra être câblé au plus près du PICBASIC).

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: DELAY_RTN: ACCIN_PROC:	123AB: Non car commence par un nombre IN: Non car il s'agit du même nom qu'une instruction

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
```

HEX ()

PBM

HEX (*Var*, *Param1*, *Param2*)

Convert an Integer constant/variable into hexadecimal string

Var est une constante/variable de type Integer.
Param1 est le nombre de digit max. à afficher (jusqu'à 8) – Valeur par défaut 4.
Param2 permet d'afficher ou non des « 0 » à la place des espaces (0 = affichage, 1 = pas d'affichage) – Valeur par defaut 1)

EXPLICATION

Cette instruction retourne la valeur hexadécimale d'un nombre dans une chaîne selon un certain format (déterminé par "**Param1**" et "**Param2**". Son utilisation est identique à celle expliquée pour l'instruction "PRINT DEC" des "PICBASIC 1B-1S-2S-2H" mise à part qu'elle peut s'utiliser en étant ou non associée à "PRINT". D'autre part, il est également possible d'avoir recours à cette instruction pour des variables de type "BYTE", "INTEGER" ou "LONG" et avec un nombre de caractères réservés qui peut aller jusqu'à 8 (contre 5 seulement sur les "PICBASIC" de la série « PB ».

EXEMPLE

```
10        DIM   F1   AS   STRING * 16
20        DIM   I   AS   LONG
30        I   =   &H123ABC
40        F1 = DEC(I,8,0) + "DEFG"                    ' F1 contient la chaîne "00123ABCDEFG".
```

Les PICBASIC de la série « PBM » permettent de sauvegarder le résultat d'une conversion via l'instruction HEX dans une variable de type String. Si vous utilisez une variable de type Long, Param1 devra être configuré à 8 afin de pouvoir obtenir le résultat complet.

```
DIM ST AS STRING * 16   ' Définition de la taille e la chaîne
DIM I AS LONG
I = &H1234ABCD
ST = HEX(I,8)           ' Si vous ne mettez pas le nombre 8, le résultat obtenu sera ABCD.
```

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

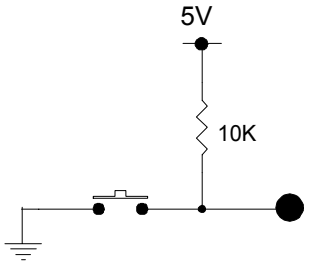
Port est une constante (0~31)
Param est une constante exprimant un delais en mS (0~255). La valeur par defaut est de 20 mS.

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

PB

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

LEFT ()

PBM

String variable = LEFT (*Var, Val*)

Gestion caractères de gauche

Var est une variable de type String
Val est une constante/variable de type Byte.

EXPLICATION

Cette instruction permet de récupérer les (**Val**) caractères de gauche de la chaîne de la variable (**Var**).

EXEMPLE :

```
10        DIM  S1  AS  STRING*16
20        DIM  S2  AS  STRING*16
30        S1  =  "LEXTRONIC"
40        S2  =  LEFT(S1,3)              'Après l'exécution de la ligne 40, la variable S2 contient la chaîne "LEX".
```

LEN ()

PBM

Variable Integer = LEN (*Var*)

Longueur d'une chaîne

Var est une variable de type String.

EXPLICATION

Cette instruction permet de connaître le nombre de caractères qui composent une chaîne contenue dans la variable (Var).

EXEMPLE :

```
10        DIM  S1  AS  STRING*16
20        DIM  I  AS  INTEGER
30        S1  =  "LEXTRONIC"
40        I  =  LEN(S1)                  'Après l'exécution de la ligne 40, la variable I=9
```

LOCATE

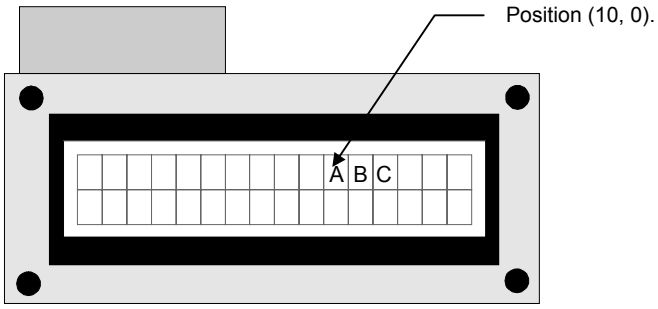
LOCATE *Val1*, *Val2*

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**.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0											A	B	C			
1																

EXEMPLE

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

LOG ()

PBM

Variable Single= LOG (*Valeur*)

Fonction Logarithmique

Valeur est une variable de type Single.

EXPLICATION
Cette instruction permet de calculer la valeur Logarithmique d'une variable.

EXEMPLE:

```
10        DIM  F1  AS  SINGLE
20        DIM  F2  AS  SINGLE
30        F2=LOG(F1)
```

LOG10 ()

PBM

Variable Single= LOG (*Valeur*)

Fonction Logarithmique commune

Valeur est une variable de type Single.

EXPLICATION

Cette instruction permet de calculer la valeur "Log10" d'une variable (même utilisation que pour **LOG**, voir ci-avant).

MID ()

PBM

String variable = MID (*Var*, *Val1*, *Val2*)

Découpage de chaîne

Var est une variable de type Chaîne.
Val1 est une constante (0~31) ou une variable de type Byte indiquant la position à partir de laquelle on veux opérer.
Val2 est une constante (0~31) ou une variable de type Byte représantant le nombre de caractères à sélectionner.

EXPLICATION

Cette instruction permet de récupérer "un extrait de chaîne" à l'intérieur de la chaîne (Var). Cet "extrait" débutera à partir du (**Val1**) caractère en partant de la gauche et comportera (**Val2**) caractères.

EXEMPLE:

```
10        DIM   S1   AS   STRING*16
20        DIM   S2   AS   STRING*16
30        S1  =  "LEXTRONIC"
40        S2  =  MID(S1,4,6)              ' Après l'exécution de la ligne 40, la variable S2 contient la chaîne "TRONIC"
```

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

ON...GOSUB

PBM

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

Appel de sous-routine 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 s'utilise de la même façon que l'instruction ci-dessus, mise à part que l'on ne réalise plus un "saut" à une adresse donnée en fonction de la valeur de la variable (**Var**), mais une "sous-routine" particulière en fonction de celle-ci.

EXEMPLE

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

ON INT () GOSUB

PBM

ON INT (*Param*) GOSUB *Adresse*

Interruption sur un Port

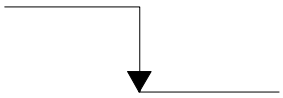
Param est une constante (0 ou 1).
Adresse est une ligne ou étiquette correspondant à la sous-routine devant être appelée.

EXPLICATION

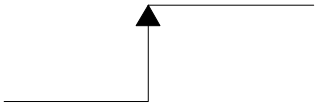
Lorsqu'un front montant/descendant est détecté sur le port 8 d'un PICBASIC de la série « PBM », cette instruction appelle la sous-routine spécifiée par ***Adresse***. Si ***Param*** est à 0, un front descendant (HAUT → BAS) sera détecté. A l'inverse, si ***Param*** est à 1, un front montant (BAS → HAUT) 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) GOSUB 10      ' Pendant l'éxécution du programme, si un front descendant survient le programme ira en ligne 10.
```



```
ON INT(1) GOSUB 10      ' Pendant l'éxécution du programme, si un front montant survient le programme ira en ligne 10.
```



REFERENCE

Sur les PICBASIC de la série « PBM », seul le port 8 peut être utilisé pour détecter des fronts montant/descendant (Il n'est pas possible d'utiliser d'autres ports pour détecter ces transitons). Il n'est pas non plus possible d'effectuer des appels de sous-routine sur une détection de niveau (haut/bas) avec les PICBASIC de la série « PBM »

Il est dans tous les cas impératif que les fils de connexions du signal entrant sur le port 8 du PICBASIC ne dépassent pas quelques cm. Il faudra également impérativement vérifier que la tension ne dépasse pas +5 Vcc sur l'entrée 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 un circuit d'anti-parasitage devra être utilisé pour éviter toute perturbation sur cette entrée.

ON INT ()=x GOSUB

PB

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

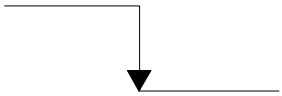
```
ON INT(0) = 0 GOSUB 20    ' Pendant l'éxécution du programme, si port 0 est au niveau bas le programme ira en ligne 10.
10 GOTO 10

20  OUT 1,1
    RETURN
```

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

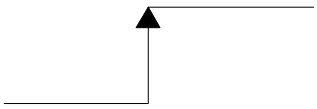
Selon le même principe il est également possible de générer une interruption sur un front (montant ou descendant) en utilisant un port spécial du PICBASIC. Pour les PICBASIC 1B/1S/2S/2H, il s'agira du port 5. Pour les PICBASIC-3B, il faudra utiliser le port 16 et le port 24 pour les PICBASIC-3H. Il suffira 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 RECV GOSUB

PBM

ON RECV GOSUB *ligne*

Interruption sur réception série RS232C

Ligne est un N° ou une étiquette qui sera appelée lors de la détection de l'interruption.

EXPLICATION

Cette instruction permet de réaliser un sous-programme dès que des données sont détectées sur l'entrée RX (I/O 15) des PICBASIC de la série « PBM ». Celles-ci seront automatiquement stockées dans le buffer dédié du PICBASIC" (même si le programme principal est en train de réaliser une autre tâche). A noter qu'il n'est pas possible d'appeler plusieurs fois cette instruction dans votre programme.

EXEMPLE:

```
10        ON   RECV   GOSUB 10

            ' Votre programme principal

100       GET I          ' Récupération de la donnée reçue dans le buffer.
```

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

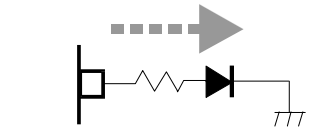
Cette instruction permet de faire passer une broche “I/O” (Port) dès lors configurée en sortie à un niveau logique donné (**Val**).

EXEMPLE:

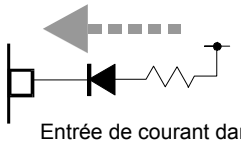
```
10      OUT 0,1     ‘ Place la broche “I/O 0” au niveau logique ‘1’ (+ 5 V)
20      OUT 0,0     ‘ Place la broche “I/O 0” au niveau logique ‘0’ (0 V)
```

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.

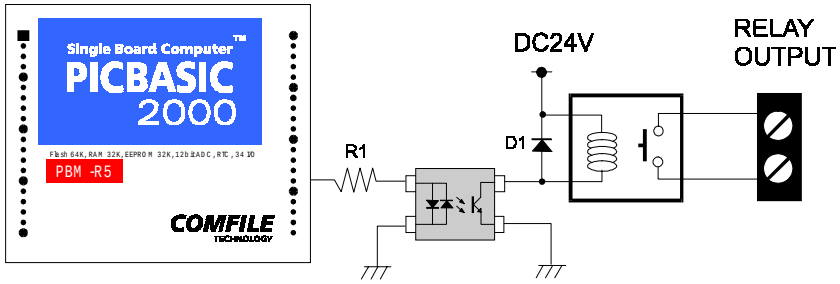


Sortie de courant d’un Port



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

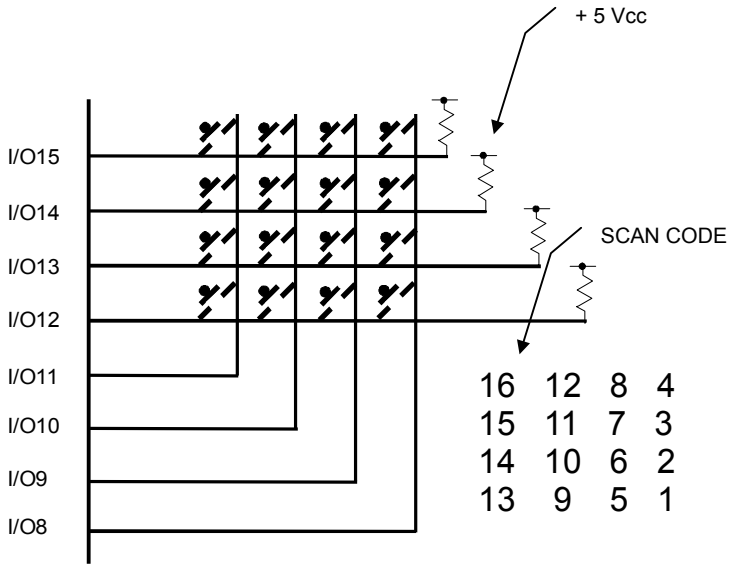
Bloc est une constante (0~3) de type Byte représentant le n° du block à utiliser.

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.

POW ()

PBM

Variable Single = POW (*Var1*, *Var2*)

Valeur de X^Y

Var1 et ***Var2*** sont des variables de type Single.

EXPLICATION

Cette instruction permet de calculer des "puissance" de nombre (X y). Les variables utilisées devront être de type "SINGLE"

EXEMPLE

```
10        DIM  X    AS  SINGLE
20        DIM  Y    AS  SINGLE
30        DIM  R    AS  SINGLE
40        R  =  POW (X,Y)                'R = X y.
```

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.

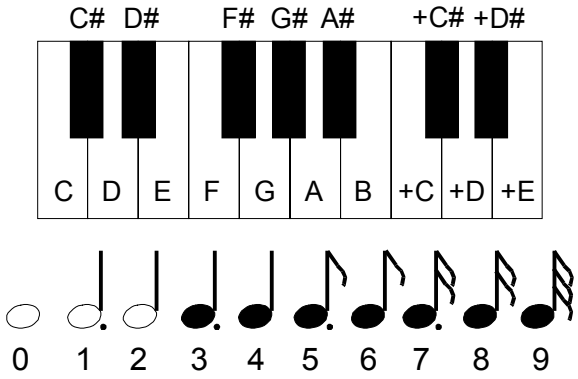
EXPLICATION

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").

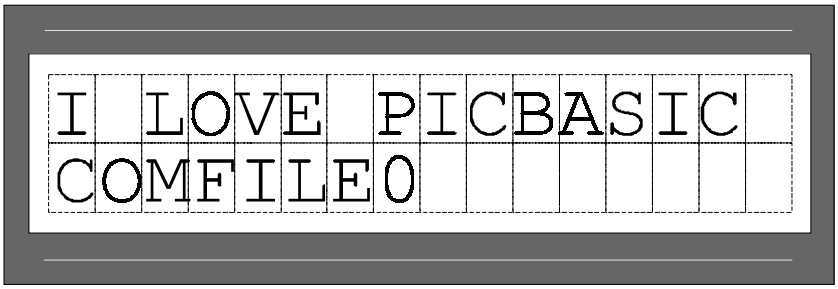
EXEMPLE1

```
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

PB

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.

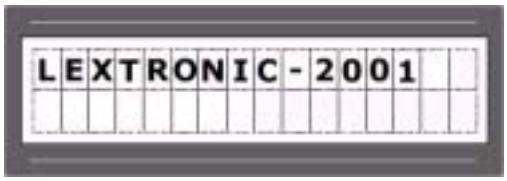
EXEMPLE 1:

```
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

PB

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

PB

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
```

PULSE

PBM

PULSE *port*, *Val*

Génération d'impulsion

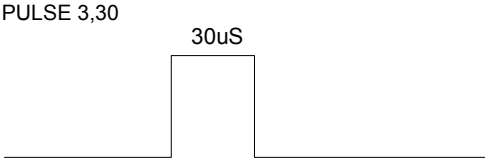
Port est une constante (0~31) ou une variable de type Byte indiquant un N° de port.
Val est une une constante de type Integer designant une durée en microseconde

EXPLICATION

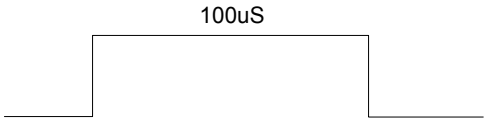
Cette instruction génère une impulsion sur un ***port*** d'une durée définie par ***Val***. Si **Val** est omis l'impulsion générée sera de l'ordre de 18 µs. Il pourra être intéressant d'utiliser l'instruction PULSE dans une boucle afin de pouvoir piloter un servomoteur en générant des impulsions de durées variables. A noter que la durée **Val** n'est pas d'une précision extrême. Le dispositif connecté sur la broche générant des impulsions devra être câblé au plus près de la broche du PICBASIC (1 à 2 cm de fil max.).

EXEMPLES

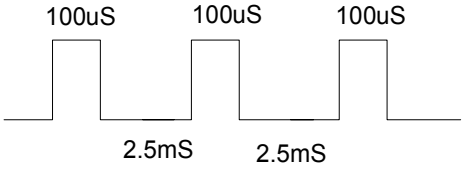
```
OUT 3,0
PULSE 3,30
```




```
OUT 3,0
PULSE 3,100
```




```
OUT 3,0
FOR I = 0 TO 2          ' Génère 3 impulsions
PULSE 3,100
DELAY 2
NEXT
```



Dans l'exemple ci-dessus, on génère 3 impulsions avec une pose de 2 ms générée par l'instruction DELAY (les 0,5 ms supplémentaires étant automatiquement occupés par l'instruction de la boucle FOR…NEXT).

PUT

PBM

PUT *Var*

Transmission de données via liaison série matérielle

Var est une constante/variable de type Byte ou une constante/variable de type chaîne.

EXPLICATION

Cette instruction sert à transmettre des données via le port série matériel des « PBM-R1 » (broche TX "I/O 14"), même si votre programme principal est en train de "réaliser" une autre action. La donnée à transmettre doit être de type “BYTE” ou "STRING". S’il s’agit d’une donnée de type “SINGLE”, aucune transmission n’aura lieu. S’il s’agit d’une donnée de type “INTEGER” ou “LONG”, seuls les 8 bits de poids “faible” seront transmis. Les données transmissent seront au format 8 bits, sans parité avec 1 bit de stop. »). Avant de pouvoir utiliser cette instruction, il faudra impérativement définir la vitesse de communication du port “RS-232” en début de programme avec l’instruction “SET RS232”.

EXEMPLE 1:

```
10        SET   RS232   4800
20        DIM   I  AS   BYTE
30        I  =  &HA0
40        PUT   I                    ' Transmission de la donnée &HA0.
```

EXEMPLE 2:

```
10        SET   RS232   4800
20        DIM   TEXTE   AS   STRING*30
30        TEXTE  =  "LEXTRONIC"
40        PUT   I                    ' Transmission des caractères "LEXTRONIC".
```

EXEMPLE 3:

```
10        DIM   I  AS   BYTE, J   AS   BYTE, K   AS   BYTE
20        SET   RS232   9600
30        ON   RECV   GOSUB 110                ' En cas de réception -> Sous programme en ligne 110

40        OUT 19,0                             ' Ces instructions génèrent des impulsions en
50        PULSE   19,100                       ' permanence afin de montrer que les "PBM"
60        DELAY   100                          ' peut en même temps émettre et recevoir des
                                                 données séries en mode "Ful-duplex".

70        IF KEYIN(9)   =   0   THEN           ' Si la broche 9 du "PICBASIC2000" est sollicitée
80          BCLR                               ' on efface le contenu du buffer de réception.
90          END IF
100         GOTO 40
110       K = BLEN(0)                          ' K contient le nombre de données reçues.
120       FOR J = 1 TO K                       ' Récupère les données et les renvoit aussitôt.
130         GET I
140         PUT I
150       NEXT
160       RETURN
```

Pour finir, rappelez-vous que les niveaux logiques présents aux ports (I/O 14 et I/O 15) sont de 0 / 5 Vcc. Si vous devez raccorder le PICBASIC à un PC ou à tout autre dispositif doté d’une liaison RS232 « standard », il vous faudra utiliser un composant MAX-232 additionnel (lequel devra être câblé au plus près du PICBASIC).

CARACTERE « RETOUR CHARIOT »

La plupart des autres langages BASIC sur compatible PC peuvent générer automatiquement en fin de transmission un « Retour Chariot » (caractères 13 et 10). Ceci permet d’effectuer un retour à la ligne lorsque vous dialoguez avec un logiciel de communication type « émulateur de terminal ». Il n’est pas possible d’utiliser cette fonction sur les « PBM ».

PWM

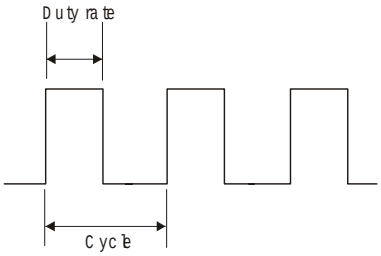
PWM *port*, *Val1*, *Val2*

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
```

RESET

PBM

RESET

Initialise le PICBASIC

EXPLICATION

Cette instruction permet d'initialiser le PICBASIC en lui faisant recommencer son programme depuis le début.

EXEMPLE

```
    RESET          ' Reset le PICBASIC
```

RIGHT ()

PBM

Sting variable = RIGHT (*Var*, *Val*)

Récuperation caractères d'une chaîne

var est une variablede type String.
Val est une constante/variable de type Byte.

EXPLICATION

Cette instruction permet de récupérer les (Val) caractères de droite de la chaîne de la variable (**Var**).

EXEMPLE

```
10        DIM   S1   AS   STRING*16, S2   AS   STRING*16
30        S1  =  "LEXTRONIC"
40        S2  =  LEFT(S1,4)                  'Après l'exécution de la ligne 40, la variable S2 contient la chaîne "ONIC".
```

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

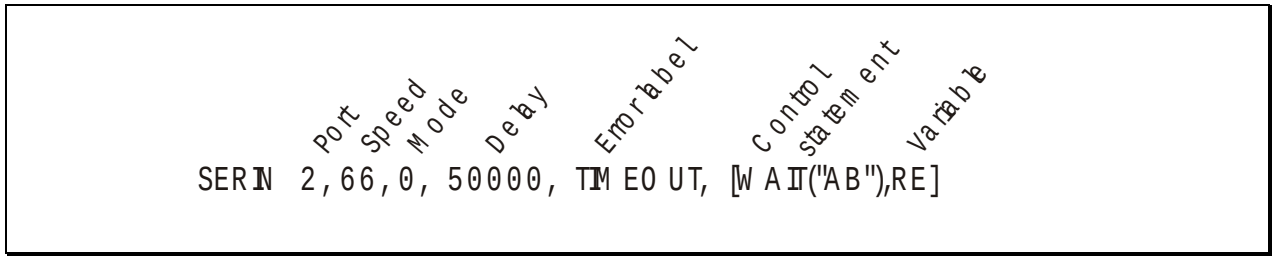
SERIN *port*, *Param1*, *Mode*, *Param2*, *Adress* [*Condition*, *var1*]

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

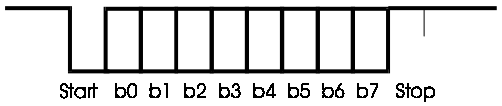
Port est une constante de type Byte représentant un N° de Port (sur la série PBM on ne peut utiliser que les ports 0 ~ 15)
Param1 est une constante qui détermine la vitesse de communication de la liaison série.
Mode est une constante (0 ou 1) permettant d'inverser la polarité des données envoyées.
Interval 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 (**Param1**), 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

PB

SERVO *Port*, *Val*

Gestion d'un servomoteur

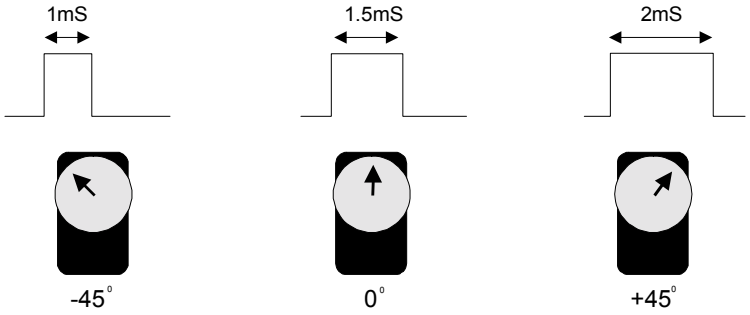
Port est une constante de type Byte représentant un N° de Port
Val est une constante ou une variable de type Byte comprise entre 0~255.

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 générés lors de la rotation du moteur. De plus le fil de liaison reliant le port du PICBASIC vers l'entrée de pilotage du servomoteur devra être le plus court possible.

SET ONINT

PBM

SET ONINT *ON*/*OFF*

Activation / désactivation des interruptions

EXPLICATION

Cette instruction permet d'activer ou de désactiver le mode d'interruption (généré par l'instruction "ON INT... GOSUB") du port 8 sur les PICBASIC de la série « PBM ».

EXEMPLE

```
10      SET ONINT ON            ' Active l'interruption par l'instruction "ON INT ... GOSUB"
20      SET ONINT OFF           ' Désactive l'interruption par l'instruction "ON INT ... GOSUB"
```

SET ONRECV

PBM

SET ONRECV *ON*/*OFF*

Activation / désactivation des interruptions

EXPLICATION

Cette instruction permet d'activer ou de désactiver le mode d'interruption de l'instruction "ON RECEV... GOSUB" sur les PICBASIC de la série « PBM ». L'état initial de ON INT(0) GOSUB est ON. Si vous utilisez l'instruction sans avoir déclaré ON INT, vous obtiendrez une erreur.

EXEMPLE

```
10      SET ONRECEV ON          ' Active l'interruption de l'instruction "ON RECEV ... GOSUB"
20      SET ONRECEV OFF         ' Désactive l'interruption de l'instruction "ON RECEV ... GOSUB"
```

SETONTIMER

PBM

SET ONTIMER *ON*/*OFF*

Activation / désactivation des Timer

EXPLICATION

Cette instruction permet d'activer ou de désactiver le mode d'interruption de l'instruction "ON TIMER... GOSUB". L'état initial de ON TIMER(0) GOSUB est ON. Si vous utilisez l'instruction sans avoir déclaré ON TIMER, vous obtiendrez une erreur.

EXEMPLE

```
10   SET ONTIMER ON              ' Active l'interruption de l'instruction "ON TIMER ... GOSUB"
20   SET ONTIMER OFF             ' Désactive l'interruption de l'instruction "ON TIMER ... GOSUB"
```

UTILISATION DES INTERRUPTIONS AVEC LES PICBASIC DE LA SERIE « PB »

Contrairement aux PICBASIC de la série « PBM », il n'est pas possible avec les PICBASIC de la série « PB » de stopper le fonctionnement des interruptions.

Il est toutefois possible d'utiliser une astuce de programmation qui vous permettra de ne pas exécuter la sous-routine d'interruption en utilisant une variable de test (voir exemple ci-dessous).

```
     DIM  INT_EN AS BYTE          ' Déclaration d'une variable permettant d'inhiber l'exécution de la routine l'interruption
     INT_EN = 0                   ' Ici on autorise la réalisation de la sous routine d'interruption
     ON TIMER(0) GOSUB 100        ' Exécute la routine d'interruption à partir de la ligne 100.
     :
     GOTO 10                      ' Boucle principale

100  IF INT_EN = 1 THEN RETURN    ' Test valeur de la variable INT_EN pour savoir si on exécute la sous-routine
   :
   :
   RETURN                         ' Fin de la sous-routine
```

Dans cet exemple, si vous désirez inhiber l'exécution de la sous-routine d'interruption, il vous suffira de mettre la variable INT_EN à 1 (l'interruption sera toujours détectée, mais la sous-routine associée ne sera pas exécutée et le programme reviendra tout de suite au programme principal).

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
```

SET RS232

PBM

SET RS232 baud rate

Déclare la vitesse de la communication du port série matériel des PICBASIC de la série « PMB »

Baud rate est une valeur fixe.

EXPLICATION

Cette instruction sert à définir le débit de transmission du port série matériel (géré par les instructions: GET, PUT, BCLR, BLEN) et spécifique aux PICBASIC de la série « PBM » (port I/O14 et I/O 15). Le débit sera défini par la valeur indiquée dans le tableau ci-dessous.

SET RS232 2400	2400 baud rate
SET RS232 4800	4800 baud rate
SET RS232 9600	9600 baud rate
SET RS232 19200	19200 baud rate
SET RS232 38400	38400 baud rate
SET RS232 57600	57600 baud rate
SET RS232 76800	76800 baud rate
SET RS232 96000	96000 baud rate
SET RS232 115200	115200 baud rate

EXEMPLE

Le programme ci-dessous affiche des en provenance du PC sur un écran LCD à commande série « Comfile » en renvoyant un echo de la séquence.

```
        SET RS232 4800                  ' Vitessede communication configurée à 4800 bds
        DIM DATA AS BYTE
        LCDINIT
        CSROFF
10      GET DATA,100
        IF DATA=0 THEN GOTO 10
        PRINT CHR(DATA)                 ‘ Affiche les donées sur l’écran LCD
        PUT DATA
        DATA = 0
        GOTO 10
```

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)

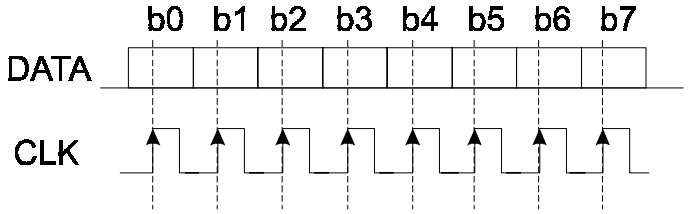
EXPLICATION

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).

```
                                  CLK port
                                    Data port
                                      Mode
                                        Bit
A = SHIFTIN (3, 4, 0, 8)
```

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.
```



SHIFTOUT

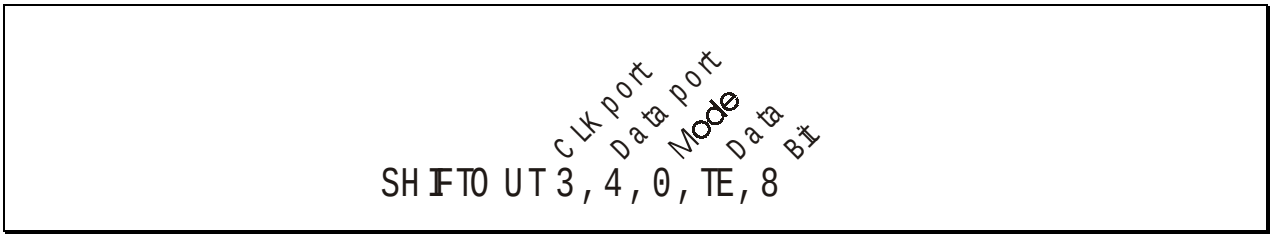
SHIFTOUT *Port1*, *Port2*, *Param*, *data*, *bit*

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

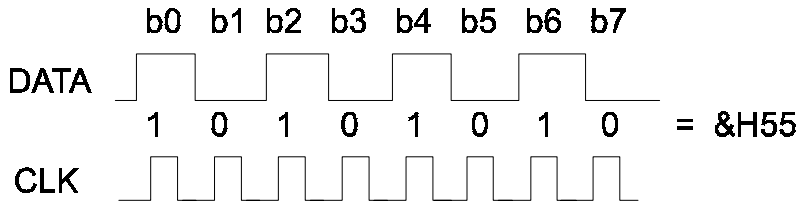
Cette instruction permet de "communiquer" très facilement avec la plupart des composants à adressage série 2 fils (type I2C™, SPI™...). Son exécution génère un signal d'horloge de synchronisation sur la sortie (**Port1**) du "PICBASIC", tout en venant "écrire sériellement" les données présentent sur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



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
```



SIN ()

PBM

Variable Single = SIN (*Valeur*)

Fonction Sinus

Valeur est une constante de type Single.

EXPLICATION

Cette instruction permet de calculer le sinus d'une variable (préalablement définie en tant que "SINGLE").

EXEMPLE

```
10        DIM   F1   AS   SINGLE
10        DIM   F2   AS   SINGLE
30        F1   =   SIN(F2)                    ' F1 contient la valeur du sinus de F2.
```

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
```

SQR ()

PBM

Variable Single= SQR (*Valeur*)

Calcul racine carré

Valeur est une constante de type Single.

EXPLICATION

Cette instruction permet de calculer la racine carrée d'une variable (préalablement définie en tant que "SINGLE").

EXEMPLE

```
10        DIM   F1   AS   SINGLE
10        DIM   F2   AS   SINGLE
30        F1   =   SQR(F2)                    ' F1 contient la valeur de la racine carrée de F2..
```

STEPOUT

PB

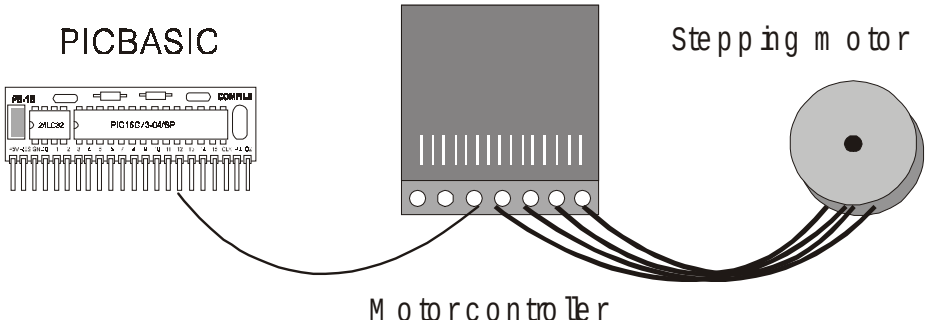
STEPOUT *Port1*, *Var1*, *Var2*, *Port2*, *Var3*

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**).

Plus la valeur de **Val1** est petite, plus les impulsions seront fines et plus le moteur tournera vite.
Inversement, plus la valeur de **Val1** est grande, plus les impulsions seront larges et plus le moteur ira lentement.

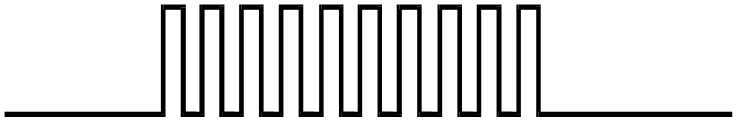
Val1 (1~255)	1B,/1S/ 2S	2H/3B/ 3H
1	11.6 KHz	60.9 KHz
2	10.3 KHz	53.2 KHz
5	7.6 KHz	38.47 KHz
10	5.3 KHz	26.31 KHz
20	3.3 KHz	16.13 KHz
50	1.54 KHz	7.463 KHz
100	0.819 KHz	3.937 KHz
200	0.423 KHz	2.024 KHz
255	0.333 KHz	1.597 KHz

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".

TIME ()

PBM

TIME (*Param*)

Lecture horloge temps réel (uniquement sur PBM-R5)

Param est une constante de type Byte.

EXPLICATION

Cette instruction uniquement utilisable sur le "PBM-R5" permet de récupérer les informations (heure/date…) de son circuit d'horloge temps réel intégré. La valeur (**Param**) permet suivant le tableau ci-dessous de définir le type d'information que l'on désire connaître. Ses informations sont sous le format BCD

Addresse	Contenu	Gamme	Composition des bits							
0	Seconde	0~&H59		10				1		
1	Minute	0~&H59		10				1		
2	Heur	0~&H23				10		1		
3	Date	1~&H31				10		1		
4	Mois	1~&H12				10		1		
5	Jour	1~&H7						1		
6	Année	0~&H99		10				1		

EXEMPLE

```
SS = TIME(0)          ' Lecture des secondes
MM = TIME(1)          ' Lecture des minutes
HH = TIME(2)          ' Lecture des heures
LOCATE 0,0
PRINT HEX(SS)         ' Conversion avant affichage car en valeur BCD
```

INFORMATION COMPLEMENTAIRE

Le PICBASIC « PBM-R5 » dispose d'une capacité de sauvegarde de 0.1F capable l'orsqu'elle est activée de sauvegarder les données de son horloge temps réel.

TIMESET

PBM

TIMESET *Param*, *Val*

Mise à jour des données de l'horloge temps réel (uniquement sur PBM-R5)

Param est une constante de type Byte (0~6).
Val est une constante (0~255) ou une variable de type Byte.

EXPLICATION

Cette instruction uniquement utilisable sur le "PBM-R5" permet de programmer les données relatives à l'horloge temps réel du module (heure/date, etc…). La valeur (**Param**) permet suivant le tableau ci-dessous de définir le type d'information que l'on désire programmer.

Addresse	Contenu	Gamme	Composition des bits							
0	Seconde	0~&H59		10				1		
1	Minute	0~&H59		10				1		
2	Heure	0~&H23				10		1		
3	Date	1~&H31				10		1		
4	Mois	1~&H12				10		1		
5	Jour	1~&H7						1		
6	Année	0~&H99		10				1		

EXEMPLE

```
                    ' Les données doivent être au format BCD
TIMESET 1, &H30     ' Configure les minutes avec la valeur 30
TIMESET 2, &H2      ' Configure les heures avec la valeur 2
TIMESET 3, &H10     ' Configure la date avec la valeur 10
TIMESET 4, &H1      ' Configure le mois de Janvier (1er mois)
```

Le tableau ci-dessous montre la relation entre le code BCD, héxadecimal et décimal.
Ceci vous permettra d'utiliser plus facilement les instructions TIME() et TIMESET() qui nécessitent des codes BCD.

Time (BCD code)	Hexadecimal	Decimal
1	&H01	1
2	&H02	2
3	&H03	3
4	&H04	4
5	&H05	5
6	&H06	6
7	&H07	7
8	&H08	8
9	&H09	9
	&H0A	10
	&H0B	11
	&H0C	12
	&H0D	13
	&H0E	14
	&H0F	15
10	&H10	16
11	&H11	17
12	&H12	18
13	&H13	19
14	&H14	20
15	&H15	21
16	&H16	22
17	&H17	23
18	&H18	24
19	&H19	25
	&H1A	26
	&H1B	27
	&H1C	28
	&H1D	29
	&H1E	30
	&H1F	31
20	&H20	32
21	&H21	33
22	&H22	34
23	&H23	35
24	&H24	36

VAL ()

PBM

Variable Integer = VAL (string)

Instruction de conversion

String est une variable de type Stirng.

EXPLICATION

Cette instruction permet de transformer les chiffres d'une donnée de type "STRING" en une valeur numérique de type "BYTE", "INTEGER" ou "LONG". Si la variable de type STRING ne comprend aucun chiffre (exemple ST="LEXTRONIC"), VAL retournera le chiffre 0.

EXEMPLE

```
10      DIM   ST  AS  STRING*16
20      DIM   I     AS  INTEGER
30      ST   =   "12345"
20      I   =   VAL (ST)                                ' I = 12345
```

VALSNG ()

PBM

Variable Single= VALSNG (*string*)

Instruction de conversion

String est une variable de type String.

EXPLICATION

Cette instruction permet de transformer les chiffres une donnée de type "STRING" en une valeur numérique de type "SINGLE". Si la variable de type STRING ne comprend aucun chiffre (exemple ST="LEXTRONIC"), VAL retournera le chiffre 0.

EXEMPLE:

```
10      DIM   ST  AS  STRING*16
20      DIM   I     AS  SINGLE
30      ST   =   "123.456"

20      I   =   VALSNG (ST)                 ' I = 123.456
```

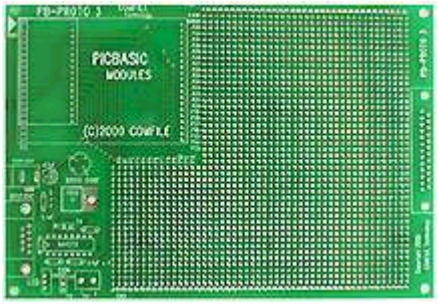
Chapitre 6.

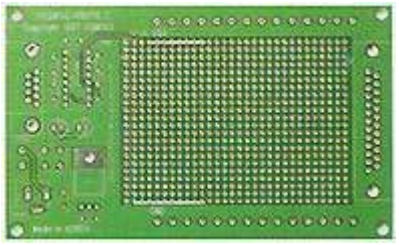
Mon premier programme

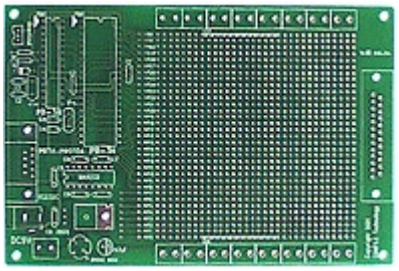
Plaques d'essais pour « PICBASIC »

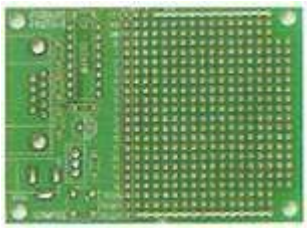
Afin de pouvoir tester rapidement les possibilités de votre PICBASIC, vous devrez en premier lieu réaliser une platine support sur laquelle vous pourrez enficher le PICBASIC. Vous pourrez réaliser cette platine en dessinant votre propre circuit imprimé ou en utilisant une plaque de prototypage à bande ou à pastille. Cette platine devra être munie d'un support permettant d'enficher le PICBASIC et également être capable de fournir une tension d'alimentation de + 5 Vcc au PICBASIC via un étage de régulation.

Nous disposons à ce titre de plusieurs platines de test (en fonction du modèle de PICBASIC que vous voulez utiliser) sous la forme de circuits imprimés (sur lesquels vous devrez souder les composants qui sont livrés à part).

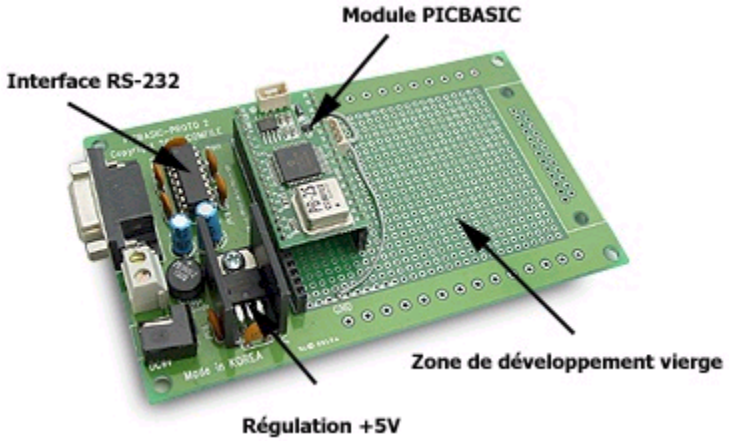






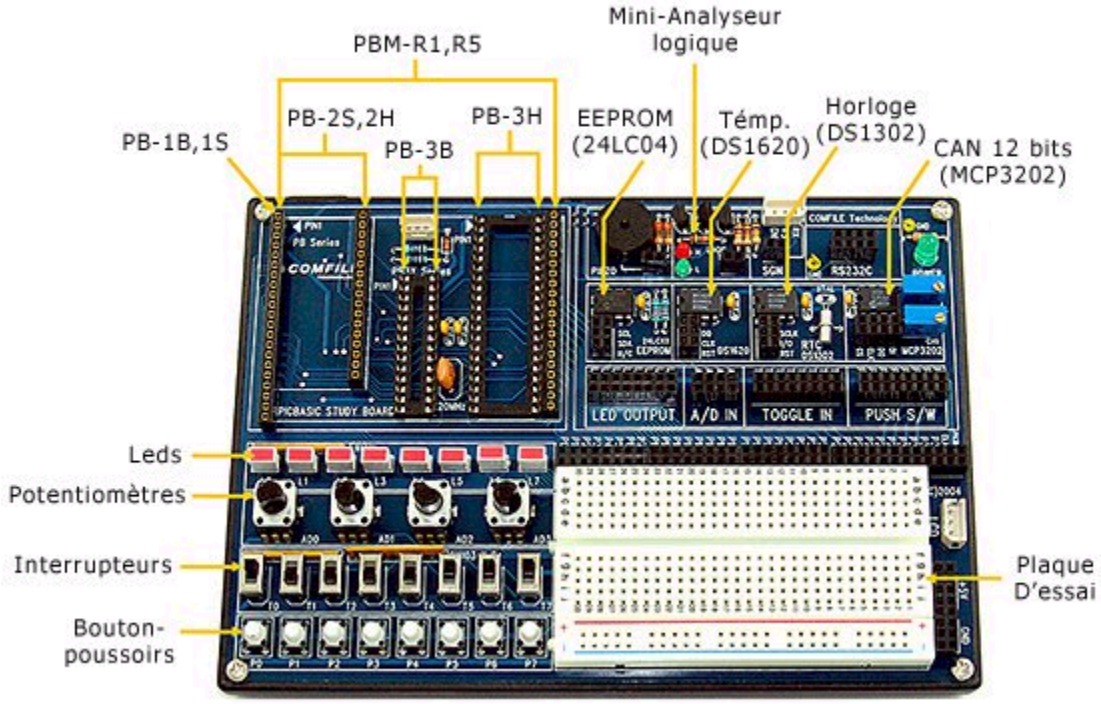


Une fois assemblée la platine ressemble à la photo ci-dessous.



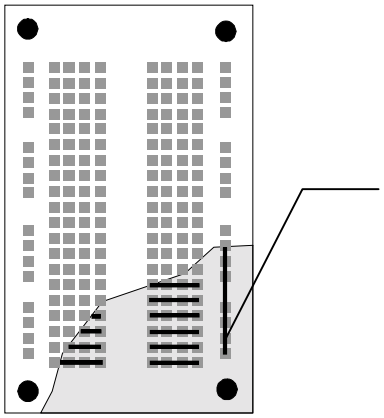
Platine d'essais pour « PICBASIC »

Parmi les autres platines que nous proposons, le modèle « **PICBASIC Study Board** » est dotée de différents supports pouvant recevoir **TOUS** les modèles de PICBASIC, cette platine est idéale pour l'expérimentation et le test. Livrée pré-câblée, elle dispose d'un étage de régulation +5V, d'une interface vous permettant "d'attaquer" directement le port "RS-232" d'un ordinateur si l'application que vous développez le nécessite.

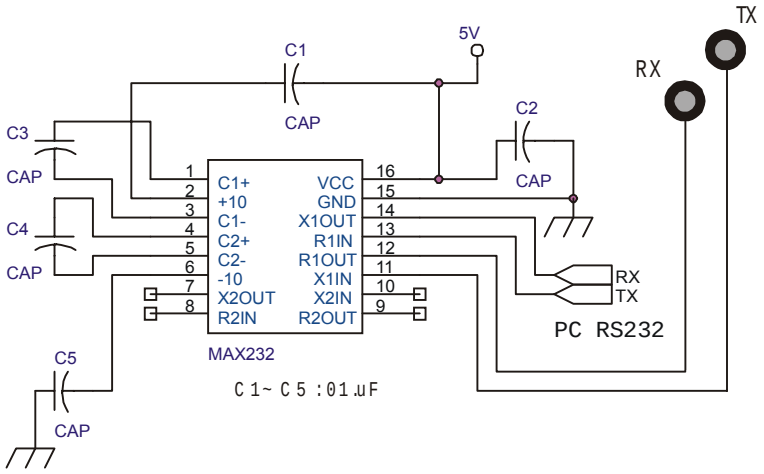


La platine « PICBASIC Study Board » comprend de très nombreux circuits périphériques:

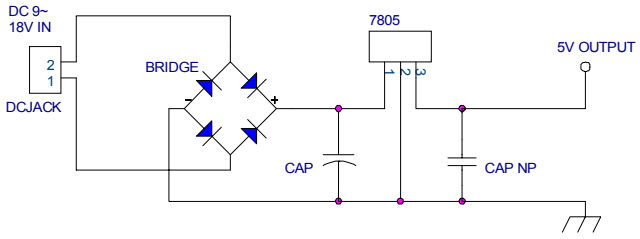
- Une EEPROM 24LC04 (pour étude des communications I2C™)
- Une sonde thermomètre/thermostat -55°C à +125°C "DS1620" (étude communications SPI™).
- Un circuit horloge temps réel "DS1302".
- Un circuit d'interfaçage RS-232 "MAX232" (sortie sur prise Sub-D 9 broches - câble en option).
- Un convertisseur analogique/numérique sur 12 bits "MCP3202" (étude communications SPI™).
- Un buzzer avec oscillateur + 8 Leds de visualisation + 8 boutons-poussoirs + 8 interrupteurs.
- 4 potentiomètres + Un mini-testeur logique avec 2 Leds (rouge / verte).
- Une sortie pour afficheur LCD alphanumériques à commande série optionnel (non livré).
- Une sortie pour afficheurs 7 segments à Leds à commande série optionnel (non livré).
- Une plaque de connexions rapides sans soudure 270 contacts (cette dernière vous permettra d'ajouter de nouveaux composants).



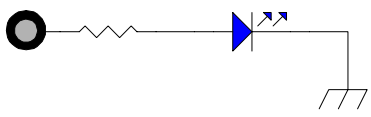
Etage « RS232 »
La platine dispose d'un circuit d'interface RS232C qui vous permettra (si votre application le nécessite) de relier le PICBASIC à un PC ou à un ordinateur via une liaison RS232 (dans le but de réaliser des échanges de données).



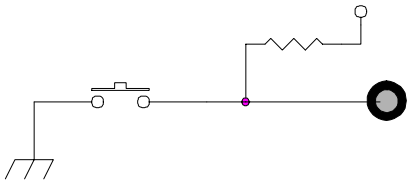
Etage de « régulation »
La platine dispose également d'un étage de régulation (vous devrez alimenter cette dernière sous une tension comprise entre 9 et 15 V – Un pont de diode vous permettra de ne pas avoir à vous préoccuper de la polarité de l'alimentation). Le régulateur de la platine peu délivrer jusqu'à 200 mA env. Si votre application consomme plus, il vous faudra utiliser une source d'alimentation +5 Vcc supplémentaire externe.



« Leds » de visualisation
Le schéma ci-dessous montre comment sont câblées les leds de la platine (la mise au niveau HAUT d'une broche du PICBASIC allumera la led)



« Bontons-poussoirs » de commande
Le schéma ci-dessous montre comment sont câblés les boutons-poussoirs (Lorsqu'un poussoir est sollicité, la broche passe au niveau logique BAS).



Les broches de l'ensemble des périphériques de la platine sont accessibles via des connecteurs femelles au pas de 2,54 mm qu'il vous suffira de relier aux broches des PICBASIC grâce à un jeu de fils livrés.

LED OUTPUT -> Correspond aux 8 broches des Leds

A/D IN -> Correspond aux 4 broches des curseurs des potentiomètres

TOGGLE IN -> Correspond aux 8 interrupteurs

PUSH S/W -> Correspond aux 8 boutons-poussoirs

Chaque composant de la platine dispose également d'un report de ses broches de commande.

A ce titre, vous trouverez des exemples de programmes sur notre site internet : www.lextronic.fr
afin que vous puissiez piloter tous les composants de la platine.

Votre premier programme

La description ci-dessous est basée sur l'utilisation de la platine modèle « **PICBASIC Study Board** » et d'un PICBASIC-3B (vous pourrez facilement transposer cet exemple avec un autre type de PICBASIC).

Préparation matérielle :
Assurez-vous en premier lieu que la platine « **PICBASIC Study Board** » **n'est pas alimentée** et positionnez le PICBASIC sur son support (attention au sens). Reliez le câble de programmation sur le PC et sur le connecteur « Download » de la platine « **PICBASIC Study Board** ». Réalisez enfin une connexion (à l'aide d'un fil livré avec la platine) entre la Led 0 (L0) et le port P8 de la platine.

Mettez la platine « **PICBASIC Study Board** » sous tension, puis "lancez" le logiciel de programmation des PICBASIC sur votre PC. A ce stade, vous devez obtenir l'écran de travail principal (le listing du programme affiché peut être différent - de même, la photo ci-dessous représente PICBASIC-LAB pour Windows™ 98, si vous travaillez sous WindowsXP™, l'écran sera légèrement différent).

Commencez par effacer le programme présent à l'écran en sélectionnant le menu:
"**File**", puis "**New**"

Recopiez alors à l'écran le programme ci-dessous:

```
10      OUT 8,1
        DELAY 1000
        OUT 8,0
        DELAY 1000
        GOTO 10
```

Note : Si vous utilisez le programme PICBASIC-Studio (parce que vous travaillez sous Windows XP™), rappelez-vous qu'il faut au préalable indiquer au programme avec quel PICBASIC vous travaillez en ajoutant en début de programme une ligne supplémentaire. Le programme sera donc :

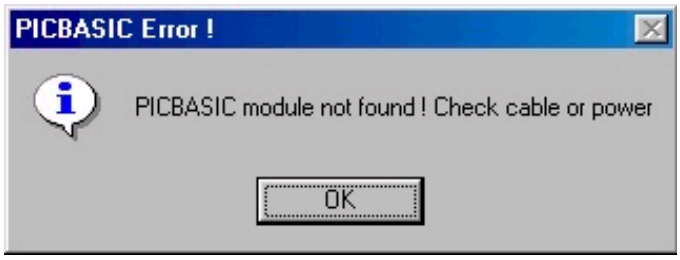
```
        CONST DEVICE = 3B
10      OUT 8,1
        DELAY 1000
        OUT 8,0
        DELAY 1000
        GOTO 10
```

Dans les 2 cas de programmes, comme vous pourrez le remarquer, toutes les lignes doivent être impérativement décalées de plusieurs espace vers la droite (les instructions ne doivent pas être collées complètement vers la gauche de l'écran). Seules les lignes dotées d'un numéro (ou d'une étiquette) doivent être accolées vers la gauche de l'écran (avec un ou plusieurs espace entre le numéro en question et son instruction) -> Dans cet exemple, la ligne 10.

Un fois le programme saisie, cliquez sur le bouton "**RUN**" en haut de l'écran (le programme vous demande alors sous quel nom vous voulez sauvegarder votre listing: tapez LED1). A ce stade, si tout ce passe correctement, une barre de progression bleue en bas de l'écran doit se remplir 2 fois (une fois pour suivre la programmation du PICBASIC et une seconde fois pour suivre sa vérification). Attention, suivant la vitesse de votre PC, la progression de la ligne bleue peu être très rapide.

Dès lors, la Led N° 1 de la platine doit se mettre à clignoter toutes les secondes !

Si rien ne se passe et si le PC vous affiche le message d'erreur ci-dessous:



Vérifiez alors:

- Que vous avez bien positionner le PICBASIC dans le bon sens.
- Si vous avez réalisez votre platine de test vous même, vérifiez que la broche RESET du PICBASIC soit bien reliée au +5 V et vérifiez tout votre montage (les 2 résistances et la diode du PICBASIC-3B doivent être câblés au plus près du PICBASIC).
- Que la platine « **PICBASIC Study Board** » soit correctement alimentée (l'utilisation d'un bloc secteur de mauvaise qualité peut être à l'origine de ce type de dysfonctionnement).
- Que le câble de programmation soit correctement connecté entre le PC et la platine « **PICBASIC Study Board** ».
- Que vous utilisez le "bon" câble de programmation correspondant à votre version de Windows™.
- Si vous travaillez sous WindowsXP™, vous devez indiquer avec quel modèle de PICBASIC vous travaillez en début de programmer et déclarer une imprimante particulière (si vous utilisez le câble parallèle).

Dans tous les cas en cas de non fonctionement, relisez les chapitres 2 – 3 – 4 et 7 pour vous aider à trouver la source du problème.

Consultez également notre site Internet www.lextronic.fr afin de télécharger d'autres exemples d'applications.

Chapitre 7.

F.A.Q

PICBASIC

Vous trouverez ci-dessous les questions qui reviennent le plus souvent au sujet des "PICBASIC" et bien évidemment leurs réponses associées. Si vous rencontrez des difficultés pour mettre en œuvre ces derniers, consulter en priorité cette section.

Je suis sous Windows XP™ et je n'arrive pas à programmer mes PICBASIC en réseaux en mode utilisateur alors qu'en mode administrateur je n'ai aucun souci.

Ce problème intervient avec le câble de programmation parallèle. Le cordon USB corrige le défaut dans la plupart des cas.

J'ai un message d'erreur "NOT PB SERISE - CHECK PB NUMBER" lorsque j'essai de programmer un PICBASIC 3B/3H .

Ce problème ne vient pas du logiciel mais de votre montage. Vérifiez à nouveau ce dernier ainsi que la valeur de la tension d'alimentation. Veuillez également à câbler les 2 résistances et la diode au plus près du PICBASIC 3B/3H.

Je travaille sous Windows XP™ et j'ai installé "PICBASIC Studio", à son lancement j'obtiens un message d'erreur "erreur 13" ?

Ce type d'erreur peut apparaître sur certaines configurations de PC. Pour y remédier, téléchargez la dernière version du "PICBASIC Studio" sur notre site : www.lextronic.fr (rubrique PICBASIC) ou sur le site de Comfile (www.comfile.co.kr) rubrique "Download".

Vous pouvez également effectuer la procédure suivante:

Sous Windows™, faite une recherche du fichier "Pbstudio.ini", puis ouvrez le fichier et recherchez la séquence:

[MenuSetting]

Korean = 0

La valeur après "korean" doit être à 0, si elle est à -1, corrigez la par 0.

Relancez PICBASIC Studio.

Une fois connecté à mon ordinateur et le logiciel chargé, je n'arrive pas à télécharger de programme dans mon PICBASIC (le programme m'indique qu'il y a un problème dans le câble ou que le module n'est pas alimenté).

1) Si vous avez réalisé vous-même le circuit imprimé servant à recevoir le PICBASIC, vérifiez les niveaux d'alimentation, les polarités, le fait que la broche RESET du PICBASIC soit bien reliée au +5 VCC.

2) Si vous utilisez un portable pour programmer les "PICBASIC", ce phénomène peut intervenir sur certains modèles (cas plus rare sur les modèles de bureau) dont la masse n'est pas présente sur la connexion 25 de la Sub-D (recherchez alors la présence de la masse sur votre ordinateur et reconnectez-la sur la prise 25 de la Sub-D - Certains modèles disposent d'une masse sur la prise 24 - à vérifiez selon votre modèle). Il semble également que sur certains modèles de PC portable, si dans le setup vous êtes en Bi-directionnelle ou uni-directionnelle vous n'avez pas les masses nécessaires sur la sortie RS232. Sur certains portables SIEMENS™ pour communiquer avec les PICBASIC vous devez paramétrer le port imprimante en mode EPP.

3) Vérifiez que vous disposez du câble de programmation et du logiciel adapté au système d'exploitation de votre compatible PC et au modèle de PICBASIC que vous utilisez (voir chapitre 4 « Les câbles de téléchargement »).

4) Si vous utilisez « PICBASIC Studio », avez-vous respecté les "règles de déclaration de base" liés au logiciel ?

A la toute première ligne de votre programme vous devez écrire l'instruction suivante pour initialiser le logiciel par rapport au PICBASIC connecté:

CONST DEVICE = R5 (si vous utilisez un "PICBASIC2000" PBM-R5)

ou

CONST DEVICE = 3B (si vous utilisez un "PICBASIC-3B")

ou

CONST DEVICE = 2H (si vous utilisez un "PICBASIC-2H")

etc.....

Je travaille sous Windows XP™ et j'ai installé "PICBASIC Studio", à son lancement j'obtiens des signes « étranges » incompréhensibles dans les menus ?

C'est "normal", le logiciel est pré-configuré pour le Coréen ! Pour obtenir les textes en "Anglais", allez dans le 5ème menu "??(T)" (qui correspond à "Setup"), puis sélectionner le premier paramètre "?? ??? ?? (U)"... C'est mieux maintenant non ? ;-))

Je désire transmettre des variables 16 bits (type single) vers un port série sur les PICBASIC2000 ?

Il vous suffit d'utiliser l'instruction FLOAT pour la convertir: Ex.: serout 1,40,0,0,[FLOAT(I)]

Je n'arrive pas à obtenir les caractères [et] dans l'éditeur du logiciel « PICBASIC-LAB »

Ce phénomène peut intervenir sur certains PC, ouvrez simplement un éditeur de texte annexe (WordPad™ par exemple, saisissez les caractères en question et avec un "Ctrl + C" (Copier) et un "Ctrl + V" (Coller), importez-les dans l'éditeur du PICBASIC.

Vous pouvez également plus simplement :
- Restez appuyé sur la touche « ALT » et taper les chiffres 091 pour obtenir le caractère [

- Restez appuyé sur la touche « ALT » et taper les chiffres 093 pour obtenir le caractère]

Consultez notre site Internet www.lextronic.fr pour être informé des dernières mises à jour de notre FAQ.

Les informations présentes dans ce manuel sont données à titre indicatif. Les caractéristiques techniques des "PICBASIC", la nature, les possibilités et le nombre de leurs instructions, ainsi que les possibilités de leurs logiciels de programmation et les caractéristiques des modules périphériques associés aux PICBASIC peuvent changer à tout moment sans aucun préavis dans le but d'améliorer la qualité et les possibilités de ces derniers.