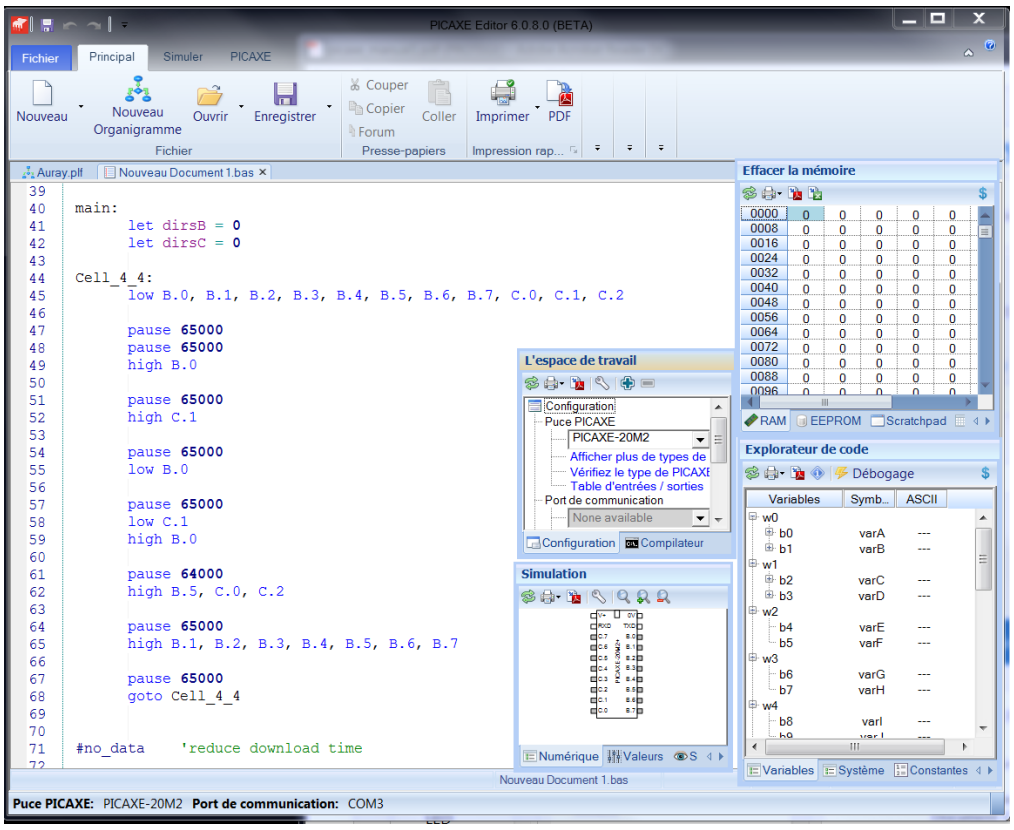
Manuel



www.picaxe.com

Table des Matières

A propos de ce manuel

Le manuel PICAXE est divisé en trois sections séparées :

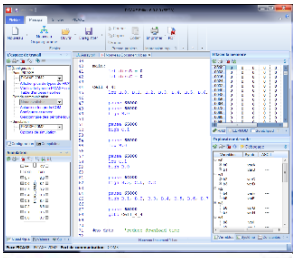
Section 1 – Démarrer	(picaxe_manuel1_fr.pdf)
Section 2 – Commandes BASICS	(picaxe_manuel2_fr.pdf)
Section 3 – Microcontrôleur circuits interfaces	(picaxe_manuel3_fr.pdf)*
Section 4-- Utilisation des Organigrammes (Flowchats)	(picaxe_manuel3_fr.pdf)*

* non traduit

La première section fournit des informations générales pour démarrer avec le système PICAXE.
Il n'est pas essentiel de comprendre les microcontrôleurs. Une série de tutoriaux introduit les fonctionnalités principales du système.
Pour plus d'informations spécifiques, syntaxe et d'exemples de chaque Commandes BASIC voir la section 2 Commandes BASIC.
Pour les circuits d'interfaçage avec les microcontrôleurs et des exemples de programmes, pour la plupart des transducteurs d'entrée/sortie commune, voir la section 3

Présentation du Logiciel

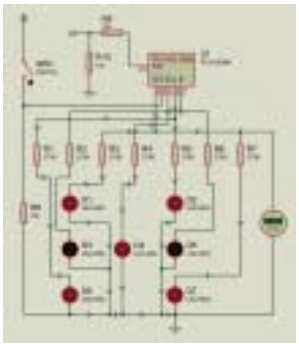
Revolution Education Ltd publie 4titres de logiciel pour utiliser avec les circuits microcontrôleurs PICAXE.
Deux sont libres, les deux autres sont des options peu onéreuses.



PICAXE Editor 6
Le PICAXE Editor 6 est l'application principales Windows utilisée pour la programmation des circuits PICAXE.
Ce logiciel est libre pour les utilisateurs de PICAXE.
Le PICAXE Editor 6 supporte les deux méthodes de développement des programmes, textuel (BASIC) et organigramme (graphique).

AXEpad
AXEpad est une version simplifiée et libre de PICAXE Editor pour une utilisation sur Linux et Mac. Il supporte la méthode de programmation BASIC.

Logicator pour micros PIC
Logicator est une application organigramme faites pour un usage éducatif.
Les programmes sont développés sous forme d'organigrammes graphiques à l'écran.
Ces organigrammes sont convertis automatiquement en fichier BASIC pour téléchargement dans les circuits PICAXE.



PICAXE VSM
PICAXE VSM est un simulateur Berkeley complet de circuit SPICE, qui simule des circuits électroniques complets qui utilisent des circuits PICAXE.
Le programme BASIC peut être déroulé ligne par ligne pour voir les entrées/sorties périphériques réagirent en fonction du programme

Ce manuel se concentre sur le langage de programmation textuel BASIC, utilisé aussi bien par PICAXE Editor, AXEpad et PICAXE VSM.

Voir le manuel 4 Flowchart pour plus de détails sur la méthode de programmation par Organigramme.

Comparaison des Logiciels

	Prog Editor 6	AXEpad	PICAXE VSM
Option programmation BASIC	X	X	X
Option programmation Organigramme	X		
Option code Assembleur	X		
Version Windows	X	(X)	X
Version Linux		X	
Version Mac OSX		X	
Simulation à l'écran	X		X
Simulation Spice Berkeley circuit			X
Accepte tous les types PICAXE	X	X	X
Coût/ Distribution	Libre	Libre	Option Payante (50)

Clé : *X = Pris en charge*
(X)= Pris en charge, mais produit plus adapté également disponibles.

Guide de Choix Rapide de Logiciel

Windows	→ Programmation textuelle BASIC	→ PICAXE Editor 6
	→ Programmation par Organigramme	→ PICAXE Editor 6
	→ Simulation SPICE	→ PICAXE VSM
Mac	→ Programmation textuelle BASIC	→ AXEpad
Linux	→ Programmation textuelle BASIC	→ AXEpad

Logiciels Tiers

Revolution produit des drivers PICAXE libre qui peuvent être utilisés pour ajouter d'ajouter la prise en charge PICAXE pour des produits tiers. Actuellement ces logiciels tiers incluent :

Win/Mac/Linux	→ Programmation par Organigramme	→ Yenka PICs
	→ Simulation de Circuit	→ Yenka Electronics
	→ PCB Artwork	→ Yenka PCB
	→ Programmation par Organigramme	→ Flowol

Forum et Support Technique

Si vous avez une question au sujet de n'importe quel aspect du système PICAXE, postez une question sur les forums très actifs et amicaux à l'adresse : www.picaxeforum.co.uk

À la même adresse vous trouverez un forum francophone très actif.

Démarrage Rapide - Conseil Préparation du PCB du projet

Beaucoup de circuits de projet Révolution Education tel que fourni dans les packs de démarrage, sont fournis avec une feuille de protection pelable sur les pastilles à souder par l'utilisateur à l'arrière du circuit. Cette feuille est rouge ou verte et peut être facilement pelée avec vos doigts avant soudure.

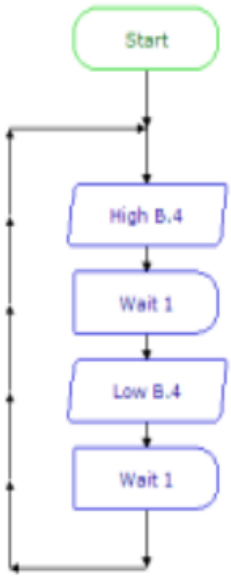
Cette feuille pelable protège les pastilles à souder durant la fabrication et le stockage pour garder ces pastilles propres et sans graisse.

Notez aussi que les pastilles à souder sur votre circuit peuvent maintenant être d'une couleur terne blanche laiteuse Terne, pas argent brillant.
Cela est dû aux produits chimiques sans plomb plus ' écologique' utilisés actuellement pour la métallisation RoHS conforme.
Ce n'est pas un défaut et la pastille peut encore être soudée aussi facilement que les pastilles d'un style" brillant.
Aucun nettoyage n'est généralement requis avant de souder.

Circuit projet CHIO 30, fourni dans le Pack de Démarrage PICAXE – 18

Démarrage Rapide – LED Clignotante

Il est fortement recommandé que vous lisiez les premiers chapitres de ce manuel avant d'utiliser le système PICAXE. Toute fois si vous êtes impatient aller à ce guide de démarrage rapide, il vous fournira un sommaire des informations expliquées plus en détails plus tard dans ce manuel.

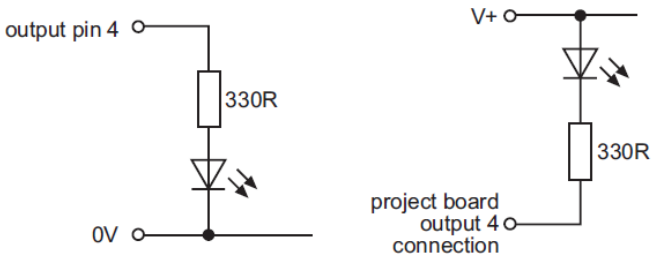


1. Installez le logiciel PICAXE Editor depuis le CDRom (ou le télécharger depuis www.picaxe.co.uk).
2. Insérer le câble USB AXE027 dans un port USB disponible (et installez le pilote USB quand c'est demandé – voir le datasheet AXE027pour plus de détails).
3. Démarrer le logiciel PICAXE Editor (cliquez Start → Programs → Revolution Education → PICAXE Editor).
   Puis cliquer Espace de travail → menu Configuraton pour afficher le panneau des Options (celui-ci peut apparaître aussi automatiquement au démarrage).
   Dans l'onglet Puce PICAXE sélectionnez le bon type de circuit PICAXE.
   Sur l'onglet Port de communication sélectionnez aussi le port COM série approprié (le port où vous reliez la câble USB série.
4. Connectez une LED et une résistance de 330 Ω sur la borne de sortie 4 du circuit PICAXE.
   Sur les circuits prototypes ou sur les 'circuits personnels', connectez la LED/Résistance entre la borne de sortie et le 0V.
   Sur les plaques projets (qui ont un buffer en transistor Darlington) connectez la LED/résistance entre V+ et la borne de sortie.
   Assurez-vous de la polarité correcte de la LED.
5. Connectez les câbles du PICAXE au matériel.
6. Connectez le 4,5V (3xPiles AA) ou une alimentation régulée 5Vpour la plaque projet. NE PAS utiliser une Pile 9V PP3.
7. En utilisant le logiciel, tapez les lignes de programmes suivantes :

```
main :  high 4
        pause 1000
        low  4
        pause 1000
        goto main
```

8. Cliquez PICAXE → Exécuter pour télécharger le programme dans le matériel.
   Après le téléchargement la LED doit flashée au rythme d'une seconde

Bravo vous avez maintenant programmé un microcontrôleur utilisant le système PICAXE.



En bref : Spécifications

Alimentation :
4,5v ou 5v DC est recommandé. Surtout n'utilisez pas des packs de piles de 6V, 7,2V ou 9V, ceux-ci pourraient endommager définitivement votre circuit. Utiliser les packs 3xAA en dépannage seulement.
Des éléments 28X2/ 40X2 étaient aussi disponibles avec des tensions faibles (1,8Và 3,3V) variantes appelées 28X2-3V et 40X2-3V. Notez que le 4,5V ou le 5V endommage définitivement ces éléments à tension d'alimentation faible.

Sorties :
Chaque sortie peut être réceptrice ou source de 20mA. Ceci est suffisant pour allumer une LED mais pas, par exemple un moteur. Le courant total maximal ne peut excéder 90mA.

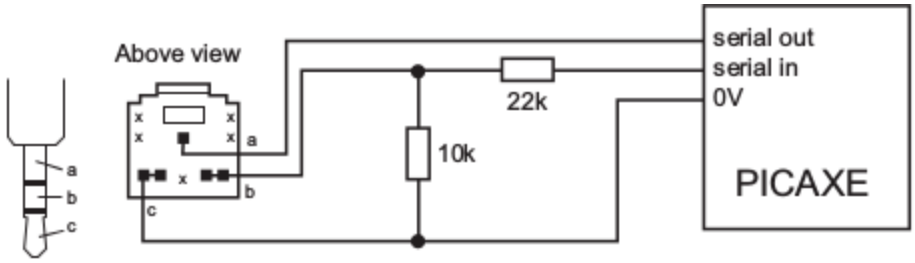
Entrées :
Une entrée peut être au-dessus de 0,8 x tension d'alimentation pour être à un niveau **H**aut, et en dessous de 0,2 x tension d'alimentation pour être à un niveau **B**as. Il est recommandé mais non essentiel de mettre les entrées inutilisées à un niveau Bas via une résistance de 10 k□.

ADC :
La gamme ADC est la gamme de tension d'alimentation. L'impédance maximum de l'entrée recommandée est de 20k. Une pin ADC non connectée sera 'flottante', donnant de fausses lectures. Cependant les pins de capteurs tactiles doivent être flottantes (pas de forçage à 1 ou 0).

Broche Chargement Série :
La pin chargement série ne doit jamais être laissée flottante. Cela donnera des opérations non fiables. Toujours utiliser les résistances 10kΩ/22kΩ□ comme montré ci-dessus, même si le circuit a été programmé sur d'autre plaquette.

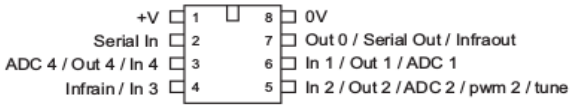
Broche Reset :
La pin Reset (si présente ne doit jamais être flottante. Ceci donnera des résultats non fiables. Toujours tirer vers le haut (le + d'alimentation) par une résistance de 4,7k□ ou 10kΩ□.

En bref – circuit de Téléchargement :

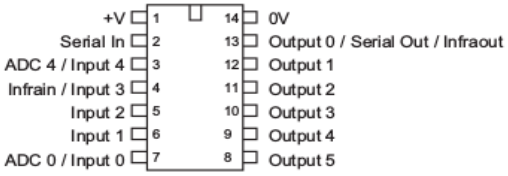


En bref – diagrammes des pins (modèles anciens)

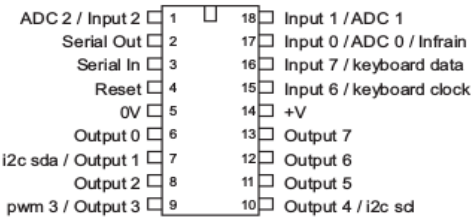
PICAXE-08M



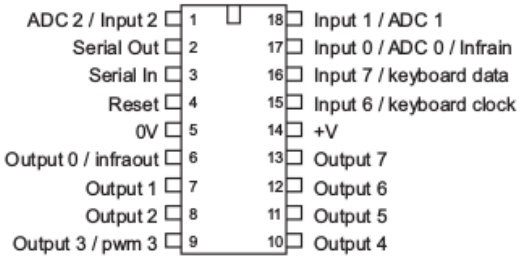
PICAXE-14M



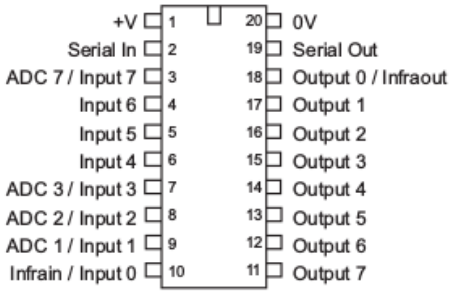
PICAXE-18X



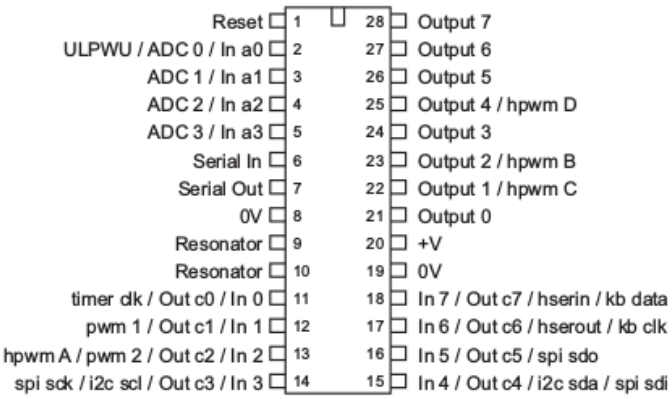
PICAXE-18M



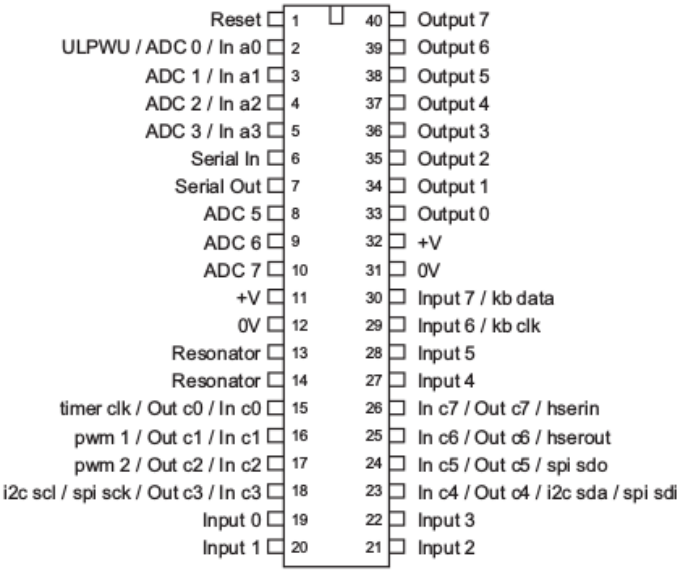
PICAXE-20M



PICAXE-28X1



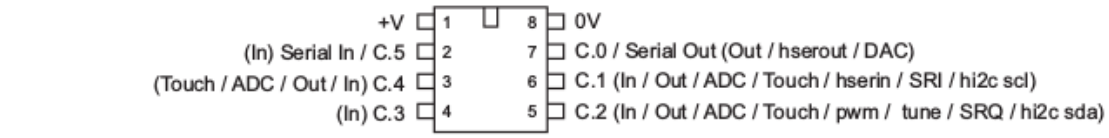
PICAXE-40X1



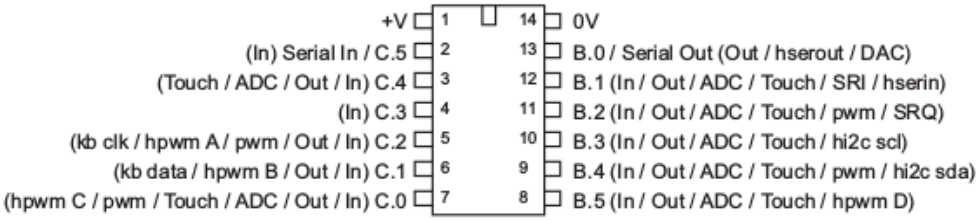
(c) Revolution Education Ltd
www.picaxe.co.uk

En bref – diagramme pins de sortie (éléments M2)

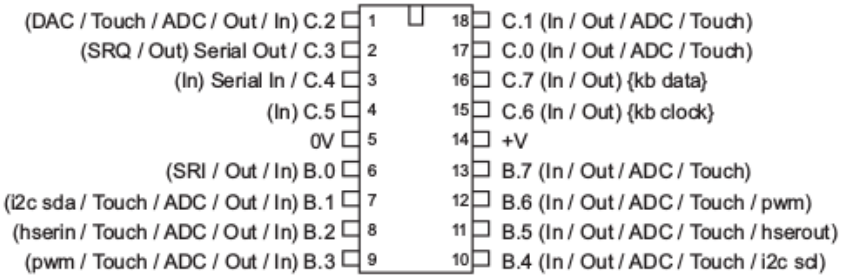
PICAXE-08M2



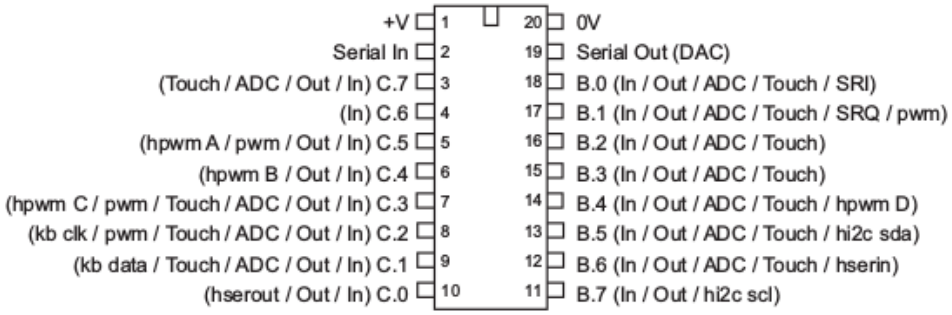
PICAXE-14M2



PICAXE-18M2

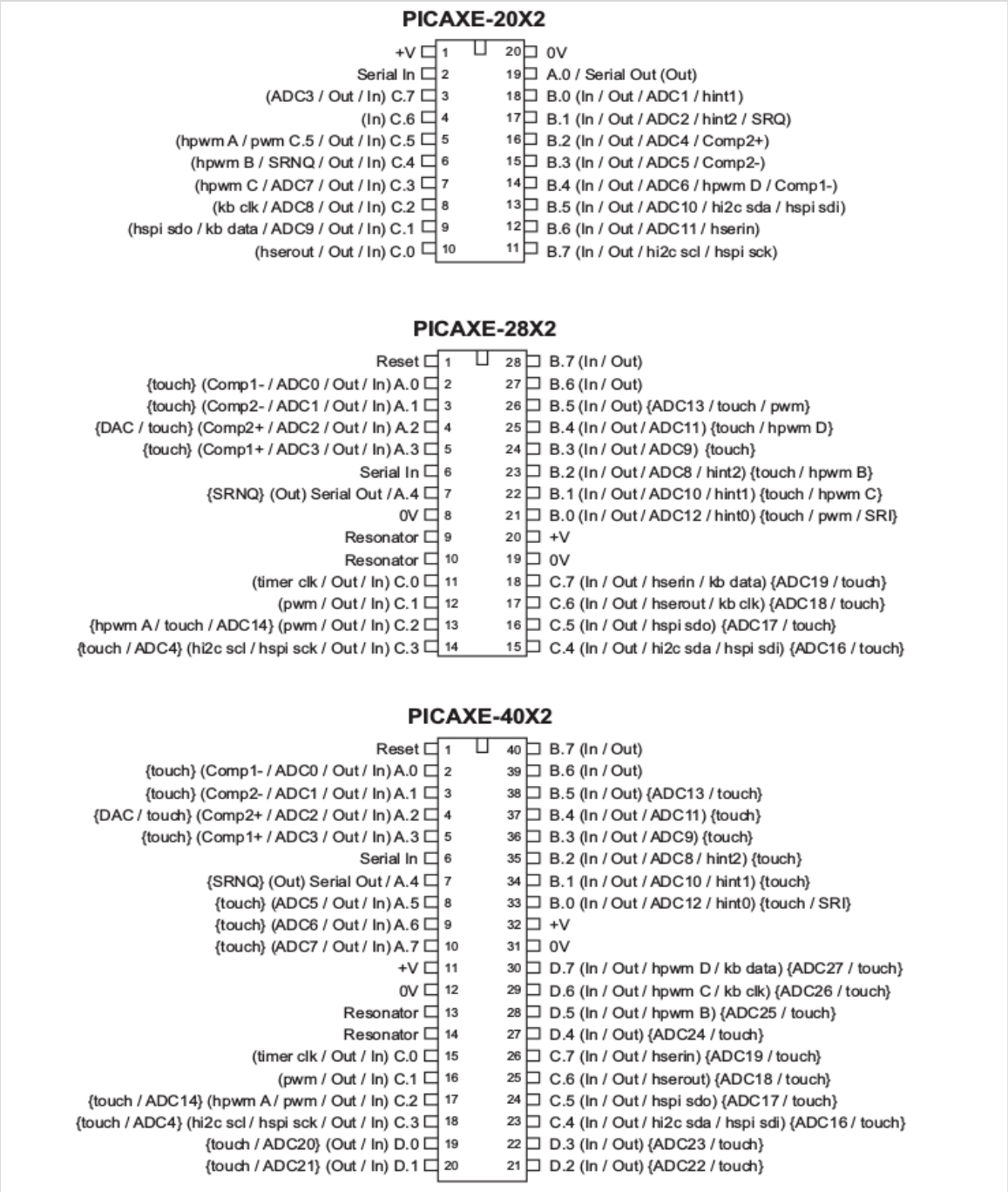


PICAXE-20M2



(c) Revolution Education Ltd
www.picaxe.co.uk

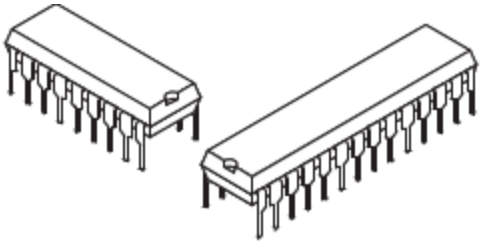
En bref – diagramme pins de sortie (éléments X2)



Les caractéristiques indiquées entre crochets {} ne sont pas disponible dans les versions plus âgées -5V et -3V.

Qu'est-ce qu'un microcontrôleur

Un microcontrôleur est souvent décrit comme un calculateur mono circuit.



C'est un circuit intégré à faible coût qui contient, la mémoire, l'unité processeur, et les circuits d'entrée et sortie dans un seul composant. Les microcontrôleurs sont vendus ' vides' et ensuite programmés avec un programme spécifique de commande.

Une fois programmé le microcontrôleur est construit dans un produit pour rendre celui-ci plus intelligent et plus facile à utiliser.
Comme exemple, un micro- onde peut souvent utiliser un simple microcontrôleur pour effectuer les informations depuis le clavier, afficher les informations utilisateur sur un afficheur 7 segments et commander les éléments (moteur du plateau, lumière, sonnerie, magnétron)
Un microcontrôleur peut souvent remplacer un nombre d'éléments séparés et même un circuit électronique complet.

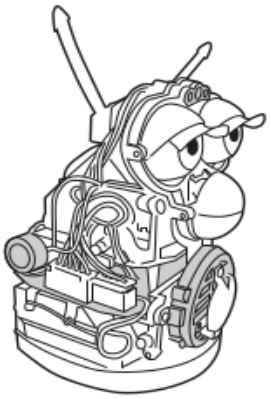
Certains des avantages de l'utilisation des microcontrôleurs dans une conception de produit sont :

- fiabilité accrue par un plus petit nombre de pièces
- les niveaux de stocks réduits, comme un microcontrôleur remplace plusieurs pièces
- l'assemblage du produit simplifié et petits produits finaux
- une plus grande flexibilité et adaptabilité du produit puisque les caractéristiques sont programmées dans le microcontrôleur et non intégrées dans le matériel électronique
- changements de produits ou développement rapides en modifiant le programme et non le matériel électronique

Les applications qui utilisent des microcontrôleurs comprennent les appareils ménagers, les systèmes d'alarme, les équipements médicaux, les sous-systèmes de véhicules, et l'instrumentation électronique. Certaines voitures modernes contiennent plus de trente microcontrôleurs - utilisés dans des systèmes de gestion du moteur pour le verrouillage à distance!

Dans l'industrie, les microcontrôleurs sont habituellement programmés en utilisant l'assembleur ou le langage de programmation 'C'. Cependant, la complexité de ces langages signifie qu'il est souvent difficile pour les jeunes étudiants dans l'enseignement, ou beaucoup d'amateurs sans formation formelle, d'utiliser ces méthodes de programmation.
Le système PICAXE surmonte ce problème par l'utilisation d'un langage beaucoup plus simple, facile à apprendre : le langage de programmation BASIC. Les programmes peuvent également être créés graphiquement par l'utilisation de l'éditeur d'organigramme.

Microcontrôleurs, Entrées et Sorties



Le jouet électronique populaire pour enfants est montré dans le diagramme. Ceci est un bon exemple d'un système *mécatronique*, car il utilise un circuit électronique pour commander un certain nombre de mécanismes. Il contient également un certain nombre de capteurs, de sorte qu'il peut réagir à des changements lorsqu'il est déplacé (par exemple être mis dans un endroit sombre ou être renversé).

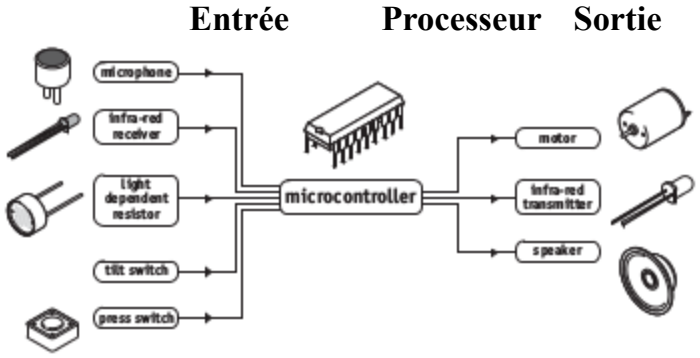
Les transducteurs d'entré pour les jeux électroniques qui détectent les changements dans un 'monde réel' et envoient des signaux dans le bloc de traitement du système électronique.

Ces transducteurs d'entrée peuvent être :

- Des boutons poussoirs à l'avant ou à l'arrière pour détecter quand le jouet est touché, et quand un contact dans le montage détecte quand le jouet est alimenté
- une LDR entre les yeux pour détecter s'il y a de la lumière ou passionnés
- un micro pour détecter le bruit ou la parole
- interrupteur à bascule pour détecter le renversement du jouet
- un détecteur infra-rouge pour détecter les signaux infra-rouge des autres jouets

Les transducteurs de sortie sont des appareils électroniques qui peuvent être mis 'en' ou 'hors' service par le bloc de traitement du système électronique. Quelques-uns de ces transducteurs électroniques peuvent être :

- un moteur pour faire bouger la bouche et les yeux
- un haut-parleur pour produire des sons
- une LED infra-rouge pour envoyer des signaux aux autres jeux.



Le microcontrôleur utilise les informations des transducteurs d'entrée pour prendre les décisions sur la façon de commander les équipements de sortie. Ces décisions sont prises par le programme de commande qui est chargé dans le microcontrôleur. Pour changer le 'comportement' du jouet, il est plus simple de changer le programme et de le télécharger dans le microcontrôleur.

Qu'est-ce que le système PICAXE

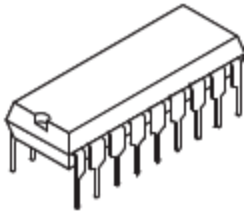
Le système PICAXE exploite les caractéristiques uniques de la nouvelle génération des microcontrôleurs basés sur les mémoires ' FLASH' économiques.

Ces microcontrôleurs peuvent être programmés et reprogrammés (100000 fois) sans nécessité un programmateur onéreux.
Le système PICAXE utilise un langage BASIC simple (ou un organigramme graphique) ainsi de jeunes étudiants peuvent démarrer la réalisation de programme après moins d'une heure d'utilisation.
Il est plus facile à apprendre et à déboguer qu'un langage de programmation industriel (C ou code assembleur)
Contrairement à d'autres systèmes basés sur des modules BASIC, toute la programmation PICAXE est sur la puce. Par conséquent au lieu d'acheter une carte pré-assemblée (coûteuse et difficile à réparer), avec le système PICAXE vous achetez simplement une puce standard et utiliser directement avec votre carte projet.
La puissance du système PICAXE est sa simplicité. Pas de programmateur, pas d'effaceur ou de système électronique compliqués ne sont requis. Le microcontrôleur est programmé via une connexion 3 fils au port série de l'ordinateur. Le circuit opérationnel PICAXE utilise 3 composants et peut être facilement construit sur une plaque d'essai, une plaque à bande cuivrée ou un circuit imprimé.
Le logiciel 'PICAXE Editor ' est libre de sorte que la seule dépense par ordinateur est le câble de téléchargement à faible coût. Dans l'environnement éducatif ceci permet aux étudiants d'acheter leur propre câble et pour les écoles d'équiper chaque ordinateur avec un câble de téléchargement.
Finalement comme la puce PICAXE ne quitte jamais le circuit projet, tous les dommages (comme ceux qui peuvent arriver quand un circuit est enlevé et remis sur un programmateur) sont éliminés.

Construire votre propre circuit / circuit imprimé

Le système PICAXE a été conçu pour permettre aux étudiants / amateurs de construire leur propres circuits imprimés pour le système PICAXE. Toutefois, si vous ne souhaitez pas faire votre propre circuit un certain nombre de kits de carte projet et de circuits imprimés sont disponibles - s'il vous plaît voir le catalogue PICAXE actuel pour plus de détails.
Si vous souhaitez faire votre propre circuit imprimé certaines références de modèles sont disponibles à la section Circuit Creator du site PICAXE à www.picaxe.co.uk des exemples de Circuit Imprimé sont disponible pour un usage éducatif et en formats assistants
Si vous souhaitez faire un "montage expérimental" le kit de développement AXE091 est très recommandé.

Qu'est-ce qu'un microcontrôleur PICAXE



Un microcontrôleur PICAXE est microcontrôleur standard Microchip microPIC® qui a été préprogrammé avec le code d'amorçage PICAXE. Le code d'amorçage active le microcontrôleur PICAXE pour être préprogrammé directement via une simple liaison série. Ceci élimine le besoin d'un programmateur conventionnel onéreux, faisant du système de téléchargement un simple câble série économique.

Le code d'amorçage préprogrammé contient aussi des routines ordinaires (telle que comment générer une pause ou une sortie son), de sorte que chaque téléchargement ne perdra pas de temps à télécharger ces données couramment requises. Cela rend le temps de téléchargement beaucoup plus rapide.

Comme les microcontrôleurs vierges achetés pour 'faire' les microcontrôleurs PICAXE sont achetés en grandes quantités, il est possible pour le constructeur de programmer le code d'amorçage et encore de vendre le microcontrôleur PICAXE à des prix proches du catalogue standard pour les simples microcontrôleurs PIC non programmés. Ceci signifie que le coût du microcontrôleur PICAXE pour l'utilisateur final est très économique.

Le code d'amorçage PICAXE n'est pas disponible pour programmer des microcontrôleurs vierges. Vous devez vous procurer des microcontrôleurs PICAXE (plutôt que des microcontrôleurs non programmés vierges) pour une utilisation dans le système PICAXE.

Libellé des puces PICAXE

Les puces PICAXE sont des microcontrôleurs Microchip PICmicro ® préprogrammés et testés.

Les composants M2 sont des composants personnalisés à l'usine et gravés 'avec le nom complet PICAXE. D'autres composants sont simplement 'gravés» avec le nom de la référence Microchip.

Type PICAXE *Gravage sur le dessus de la puce*

- PICAXE-08M2 PICAXE-08M2+
- PICAXE-14M2 PICAXE-14M2
- PICAXE-18M2 PICAXE-18M2
- PICAXE-20M2 PICAXE-20M2
- PICAXE-20X2 PIC18F14K22
- PICAXE-28X1 PIC16F886
- PICAXE-28X2 PIC18F25K22
- PICAXE-40X1 PIC16F887
- PICAXE-40X2 PIC18F45K22

Remplacement des puces PICAXE âgées

Type PICAXE	*Gravage*	*Type de Remplacement*
• PICAXE-08	PIC12F629	*Superseded by 08M2*
• PICAXE-08M	PIC12F683	*Superseded by 08M2*
• PICAXE-14M	PIC16F684	*Superseded by 14M2*
• PICAXE-18	PIC16F627(A)	*Superseded by 18M2*
• PICAXE-18A	PIC16F819	*Superseded by 18M2*
• PICAXE-18M	PIC16F819	*Superseded by 18M2*
• PICAXE-18X	PIC16F88	*Superseded by 18M2*
• PICAXE-20M	PIC16F677	*Superseded by 20M2*
• PICAXE-28A	PIC16F872	*Superseded by 28X1*
• PICAXE-28X	PIC16F873A	*Superseded by 28X1*
• PICAXE-28X2-5V	PIC18F2520	*Superseded by 28X2*
• PICAXE-28X2-3V	PIC18F25K20	*Superseded by 28X2*
• PICAXE-40X	PIC16F874A	*Superseded by 40X1*
• PICAXE-40X2-5V	PIC18F4520	*Superseded by 40X2*
• PICAXE-40X2-3V	PIC18F45K20	*Superseded by 40X2*

Quelle puce PICAXE ?

Le système PICAXE peut être utilisé avec différentes tailles physiques de puces PICAXES (8, 14, 18, 20, 28, 40 pins). La première différence entre les tailles est le nombre d'entrée/sortie disponibles- la puce la plus grande est un peu plus chère, mais elle a le plus d'entrée/sortie disponibles. Le même langage BASIC est commun à toutes les tailles de puces.

Dans une taille de puce, il y a aussi différentes variantes (par exemple pour le PICAXE à 20 pins les variantes 20M2 et 20X2 sont disponibles). La principale différence entre les variantes est la quantité de mémoire (par exemple la longueur d'un programme qui peut être téléchargé dans la puce). Les variantes avec des spécifications supérieures ont aussi plus de fonctionnalités (par exemple, entrées analogiques à haute résolution et la compatibilité I2c, comme décrit dans la section suivante).
Tout projet peut être mis à niveau vers la variante supérieure à tout moment (par exemple, si votre programme est trop long pour la variante de puce utilisée) en remplaçant simplement le microcontrôleur dans votre circuit par la variante améliorée.

Toutes les variantes améliorées sont compatibles pin à pin et par programme avec l'appareil ayant les plus faibles spécifications.

Les composants recommandés pour de nouvelles conceptions sont :

Standard :

08	PICAXE-08M2
14	PICAXE-14M2
18	PICAXE-18M2
20	PICAXE-20M2

Avancé :

20	PICAXE-20X2
28	PICAXE-28X2
40	PICAXE-40X2

La table suivante montre les différences fonctionnelles primaires entre les microcontrôleurs PICAXE disponibles :

Pour 'les amateurs' en général les séries M2 et X2 sont recommandées.

Standard :(800-1800 ligne de mémoire, pour chacun des emplacements distincts (jusqu'à 2)

08M2	5 E/S configurables	0-3 ADC	32MHZ
14M2	11 E/S configurables	0-7 ADC	32 MHZ
18M2	16 E/S configurables	0-10 ADC	32 MHZ
20M2	16 E/S configurables	0-11 ADC	32 MHZ
28X1	0-12 Entrées, 9-17 Sorties	0-4 ADC	20MHZ
40X1	8-20 Entrées, 9-17 Sorties	3-7 ADC	20MHZ

Avancé : (2000-3200 lignes de mémoires, pour chaque emplacement séparé (jusqu'à 4)

20X2	18 E/S configurables	0-8 ADC	64 MHZ
28X2	22 E/S configurables	0-16 ADC	64 MHZ
40X2	33 E/S configurables	0-27 ADC	64 MHZ

Tous les composants fonctionnent par défaut à 4 MHz (8 MHz pour les références X2). Pour une utilisation à une plus grande vitesse s'il vous plaît voir la commande *'setfreq'* dans la partie 2 du manuel.

Les composants plus âgés 08, 14, 18 et 20 pins 'A' et 'M' et 'X' ne sont plus fabriqués car ils sont maintenant remplacés par les composants M2.

Les composants plus âgés 28 et 40 pins 'A' et 'X' ne sont plus fabriqués car ils sont maintenant remplacés par les composants X1 et X2.

Utilisation du système PICAXE

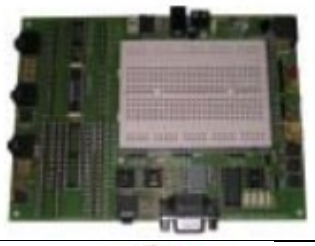
Pour utiliser le système PICAXE vous devez avoir :

- Un microcontrôleur PICAXE
- Un circuit d'essai PICAXE
- Une alimentation (4 batteries AA soit 4,8V ou 3 piles alcalines 1,5V soit 4,5V
- Un câble de téléchargement (USB ou série)
- Le logiciel libre 'PICAXE Editor' ou le logiciel 'AXEpad'

Tous ces éléments sont inclus dans tous les packs 'de démarrage' PICAXE.
Pour exécuter le logiciel 'PICAXE Editor' vous devez avoir un ordinateur fonctionnant sous Windows XP, Vista, 7, 8 ou plus récent.
Pour exécuter le logiciel AXEpad vous devez avoir un PC avec une distribution x386 Linux ou MAC avec OSX (10.2 ou plus récent).
L'ordinateur requière aussi un port USB (pour le câble USB AXE027). Voir la section Installer le Port USB/Série pour plus de renseignements

Ensembles de Démarrage PICAXE

Pour démarrer avec un système PICAXE un ensemble de Démarrage est recommandé. Les 5 ensembles de Démarrage contiennent le câble de téléchargement USB et le boîtier de pile. Toutefois la carte projet et la puce PICAXE varient dans chaque ensemble de démarrage comme indiqué ci-dessous. 3 piles AA sont aussi requises (non incluses).

	**Ensemble de Démarrage PICAXE-08 (AXE003U)** Carte Prototype PICAXE-08, Puce PICAXE, CDROM, câble de téléchargement et boîtier de piles. Kit à assembler.
	**Ensemble de Démarrage PICAXE-14 (AXE004U)**
	Ensemble de Démarrage PICAXE-20 (AXE005U Carte Prototype PICAXE-14, Puce PICAXE, CDROM, câble de téléchargement et boîtier de piles. Kit à assembler.
	Ensemble de Démarrage PICAXE-18 (AXE002U) Carte Prototype PICAXE-18, Puce 18M2 PICAXE, CDROM, câble de téléchargement et boîtier de piles. Pré-assemblé (puce 18M2 fournie)
	**Ensemble de Démarrage PICAXE-28X1 (AXE001U)** Carte Prototype PICAXE-28, Connecteur câble, Puce 28X1 PICAXE, CDROM, câble de téléchargement et boîtier de piles. Pré-assemblé (puce 28X1 fournie)
	**Ensemble de Démarrage Développement (AXE091U)** Spécialement conçu pour les amateurs avec une grande zone de prototypage et des entrées / sorties pour l'expérimentation. Le circuit de développement peut soutenir toutes les tailles de puces PICAXE et est fourni avec une puce PICAXE-18M2. Pré-assemblé.
	**Ensemble de Démarrage Tutoriel (AXE050U)** L'ensemble de didacticiel est conçu pour l'école et utiliser pour permettre aux étudiants d'apprendre rapidement le langage PICAXE par une série de tutoriels structurés (fournis sur le CD ROM). Carte pré-assemblée avec LDR, commutateurs et affichage de sortie.

Cartes Projets PICAXE

Des cartes projets/kit sont aussi disponibles pour les utilisateurs qui ne souhaitent pas réaliser eux-mêmes leur circuit imprimé. Toutes les cartes ont le connecteur série de téléchargement pour la programmation de la puce PICAXE via le câble série/USB de téléchargement.

Carte Proto PICAXE-08 (AXE021)
Petite carte pour permettre le prototypage rapide de circuits PICAXE-08. La carte fournit le circuit de base et le connecteur de téléchargement, avec une petite zone de prototypage pour permettre la connexion de circuits d'entrée et de sortie.

Pilote Moteur PICAXE-08 (AXE023)
La carte pilote moteur peut être utilisée pour piloter 4 sorties on/off (exemple : buzzers) ou les sorties peuvent être utilisées par paire pour permettre la commande avant-inverse-stop de deux moteurs.
Pré-assemblée avec la puce PICAXE-08 incluse.

Carte Projet PICAXE-14 (AXE117)
Carte Projet PICAXE-20 (AXE118)
Le circuit imprimé de la carte projet est un circuit imprimé de qualité professionnelle qui permet aux élèves de construire une carte projet qui a 6 sorties et 5 entrées. La carte fournit la place pour la puce PICAXE-14M2/20M2, la prise de téléchargement et le driver Darlington.
Kit à assembler (incluant les circuits imprimés)

Carte Projet PICAXE-18 (CHI030A)
La carte d'interface standard PICAXE-18 est un circuit pré-assemblé équipée d'une puce driver Darlington de sorte que les équipements en sortie tel que moteur ou buzzer peuvent être connectés directement à la carte. Accepte 5 entrées et 8 sorties.

Carte projet de puissance PICAXE-18 (CHI035A)
La Carte interface de puissance pré-assemblée fournit 4 drivers FET pour commander des équipements de sortie demandant de forts courants. Par l'ajout optionnel de la puce pilote moteur L293D, 2 sorties de commande de moteur peuvent être ajoutées.

Carte Projet PICAXE-28 (AXE020)
Une carte pré-assemblée équipée avec une puce darlington pour 8 sorties.
Par l'ajout optionnel d'une puce pilote moteur, 2 sorties de commande de moteur peuvent être ajoutées à la carte. Livré avec un câble en nappe équipé d'un connecteur.

Carte Proto PICAXE-28/40 (AXE022P)
Le kit carte proto PICAXE-28/40 permet un développement rapide des projets PICAXE -28X1/X2 et 40X1/X2. La carte fournit le circuit basique et le connecteur de téléchargement, avec les connexions pour les E/S. Le support pour l'EEPROM est inclus

Installation du Logiciel

Configuration requise :
Pour installer le logiciel vous devez avoir un ordinateur équipé de Windows 95 ou plus récent avec approximativement 50 MB d'espace libre.

Installation

1. Démarrer et connectez-vous à votre ordinateur (certains systèmes d'exploitation que demande vous vous connectiez comme 'administrateur' pour installer le logiciel).
2. Insérez le CDROM, ou téléchargez et lancez le fichier d'installation depuis la page logiciel depuis www.picaxe.co.uk
3. Suivre les instructions à l'écran pour installer le logiciel. Sur de vieux ordinateurs on peut vous demander de redémarrer l'ordinateur après l'installation.
4. Insérez le câble AXE027 dans un port USB. L'AXE027 aura besoin d'un logiciel pilote pour la première utilisation. Un assistant 'Nouveau matériel détecté' démarre automatiquement (voir la fiche AXE027 pour plus de détails).
5. Cliquez → Démarrer → Programme ->Revolution Education → PICAXE Editor pour démarrer le logiciel.
6. Dans l'Espace de travail, sur l'onglet 'Configuration' sélectionnez la taille et le type du microcontrôleur PICAXE que vous utilisez. Sur l'onglet 'Port' sélectionnez le port série COM approprié.

Vous êtes maintenant prêt à utiliser le système.

Installation sur un réseau RM CC3

Le logiciel s'exécutera sur tous les réseaux d'écoles, incluant le RM CC3.

1. Il est recommandé d'utiliser l'install non-compressée MSI fournie avec le CDROM, plutôt que le téléchargement sur le net.
2. Connectez-vous comme Administrateur Système et utilisez votre logiciel préféré (RM application Wizard) pour construire un paquet distribution utilisant l'install MSI trouvée dans le dossier progedit sur le CDROM. Si vous préférez vous pouvez aussi copier manuellement les fichiers MSI dans la zone RM Packages/Applications.
3. Mettez à jour la liste des paquets du poste de travail approprié utilisant la Gestion Console RM et générer des raccourcis comme désiré.
4. Utilisateurs XP- notez que vous pouvez avoir à créer deux règles 'hash' Software Restriction- une pour l'exécutable progedit.exe et une autre pour le raccourci. Pour faire ce login comme Administrateur Système sur un poste XP, cliquer Démarrer → Programmes → Gestion Système →Réglages Restrictions Logiciel. Ouvrez Configuration Ordinateur → Réglages Windows → Stratégies de Restriction Logicielle → Règles additionnelles et explorer jusqu'à l'exécutable progedit et cliquez OK.
5. Le chemin par défaut pour le dossier sauvegarder/ouvrir peut être édité comme demandé dans le fichier appelé network.ini trouvé dans le dossier principal d'installation.

Installation des pilotes de câble AXE027 USB

Les ordinateurs les plus anciens ont un connecteur série 9 pins pour raccorder un câble de téléchargement PICAXE. Toutefois les ordinateurs les plus récents n'ont pas ce connecteur 9 pins pour gagner de la place, dans ce cas un port USB peut être utilisé à la place.

Le système d'interface USB est un système intelligent qui exige que l'appareil connecté se configure automatiquement lorsqu'il est connecté au port USB de l'ordinateur. Bien qu'il soit théoriquement possible de construire une version USB de PICAXE, la mémoire supplémentaire nécessaire ferait augmenter le coût de chaque puce PICAXE de près de 3 £.

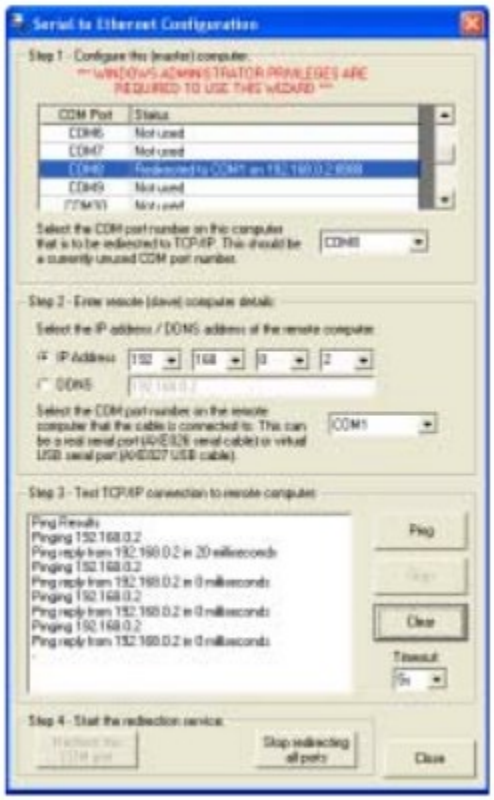
Par conséquent, un autre système est utilisé. L'utilisateur achète un câble USB 'série' (référenceAXE027) peu cher, qui est un câble PICAXE spécial intelligent qui permet aux puces d'être programmées via le port USB.

Procédure d'installation du câble USB

(Voir le fichier d'aide (AXE027.pdf) pour plus d'instructions détaillées. Ceci est disponible sur la page software de www.picaxe.co.uk ou le fichier USB du CDROM).

1. Acheter le câble USB AXE027.
2. Le Connecter sur le port USB de l'ordinateur
3. Insérer le CDROM fournit avec l'adaptateur USB pour installer le dernier pilote.
4. Noter le numéro du port série USB alloué à l'adaptateur USB.
5. Connecter le câble standard PICAXE à l'adaptateur USB.
6. Démarrer le logiciel PICAXE Editor et sélectionner le port COM approprié depuis le L'Espace de Travail → Port de Communication.
7. Cliquer Actualiser les ports COM pour rafraichir la liste des ports disponibles.
8. Utiliser le logiciel et le matériel normalement.

Téléchargement sur un réseau utilisant TCP/IP



Le logiciel PICAXE Editor 6 accepte la redirection des ports Com vers une connexion TCP/IP 'ethernet'. Cette connexion peut être un réseau local ou même l'internet.

Pour utiliser cette configuration un port COM 'virtuel' est créé sur l'ordinateur local (l'ordinateur qui exécute le logiciel PICAXE Editor) et crée une connexion TCP/IP. Sur l'ordinateur distant (où le câble de téléchargement est connecté au port USB série) une petite application de service de redirection est installée et redirige le port COM réel vers la connexion TCP/IP.

Ce système permet au logiciel PICAXE Editor 6 d'utiliser le port série sur l'ordinateur distant exactement comme s'il était sur l'ordinateur local - de nouveaux téléchargements de programmes et même des données série peuvent être transmis de façon transparente en arrière et en avant sur la connexion TCP/IP.

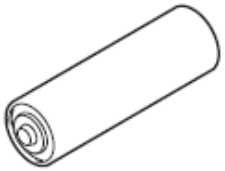
Pour démarrer cette connexion deux étapes sont nécessaires :
1) Lancez l'assistant (PICAXE → Assistants → COM au menu TCP/IP) sur l'ordinateur local pour configurer la connexion locale
2) Installez le logiciel SEC sur l'ordinateur distant et 'exécuter l'Assistant pour sélectionner le port série à utiliser.
Ce logiciel fonctionne comme un service et peut donc être configuré pour toujours démarrer lorsque l'ordinateur est sous tension. Cela lui permet d'être installé sur les machines sans surveillance (par exemple dans un musée)

Pour plus de détail voir la feuille du logiciel Connexion Série Ethernet.

Alimentation PICAXE

Toutes les puces PICAXE exécuteront les programmes à une tension comprises entre 3 et 5,5V DC
Les composants dernière génération (M2 et X2) acceptent une tension de 1,9V

NOTE IMPORTANTE- ce manuel décrit l'usage des composants standards dans la gamme 3-5,5V, les derniers composants 28X2 et 40X2 sont aussi disponible en option pour une variante de tension basse 1,8 à 3,3V. L'utilisation d'une alimentation 5v sur un composant 3,3v, l'endommagera définitivement,



Il est recommandé d'utiliser l'une des trois sources suivantes pour fournir l'alimentation :

- 3 piles alcalines AA (4,5V)
- 4 batteries rechargeables AA (4,8V)
- Une alimentation régulée 5V

Ne pas utiliser une pile 9V PP3, cela est au-dessus de la tension maximale de la puce PICAXE et causerait des dommages permanents. Notez que la plupart des boîtiers de batterie 4xAA et 3xAA utilisent le même connecteur de style 'bouton-pression comme une batterie PP3 9V. Notez que la fourniture de ce type connecteur ne signifie pas que la carte projet doit utiliser une batterie PP3 9V, il est juste dommage que tous les boîtiers de batteries utilisent le même style de connecteur.

Les batteries PP3 9V sont conçues pour de très faible intensité, des applications à long terme (par exemple, un détecteur de fumée ou multimètre). Bien qu'une batterie PP3 9V régulée à 5V va travailler pour de courtes périodes avec un microcontrôleur, elle se videra très rapidement quand un périphérique de sortie (p.ex. LED, moteur ou buzzer) est connecté. Par conséquent, utilisez toujours des piles AAA ou batterie AA plutôt que des piles 9V PP3 dans les projets de microcontrôleur (tel qu'il est utilisé avec de nombreux biens de consommation portables par exemple CD portables, torches LED, etc.) Prenez soin lors de l'insertion des puces PICAXE dans votre circuit de vous assurer qu'ils sont dans le bon sens. Prenez un soin supplémentaire avec les puces 18 pins, si elle est insérée 'à l'envers' les connexions d'alimentation seront inversées, causant des dommages permanents à la puce.

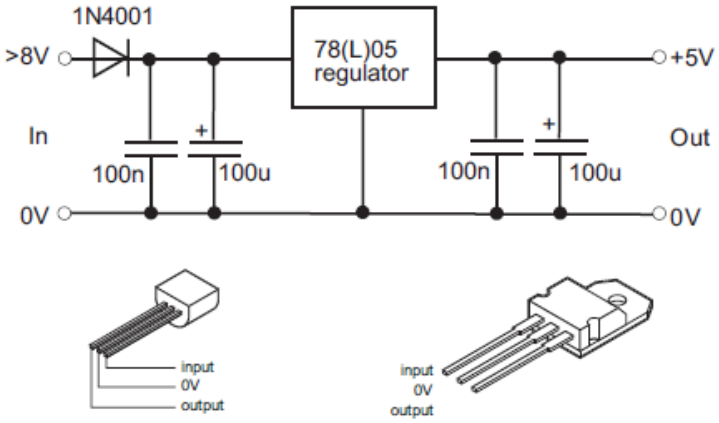
Pack batterie AA
Les piles alcalines AA ont une tension nominale de 1,5V, ainsi trois éléments donnent 4,5V, Si vous souhaitez utiliser 4 piles, alors utilisez une diode 1N001 en série avec le pack de batterie. La diode produit une protection en tension, et comme la diode a une chute de tension de 0,7V, la tension du microcontrôleur devient acceptable 5,3V .
Les batteries rechargeables AA ont une tension nominale de 1,2V, ainsi 4 éléments donneront 4,8V,
Faite attention de ne pas faire un court-circuit sur les batteries, le courant important de court-circuit pourrait causer des dommages ou démarrer un feu

Utilisation des connexions de la batterie.
Les packs de batteries sont souvent connectés aux cartes de circuit imprimé par des fils. Assurez-vous toujours de bien connecter les fils rouges et noirs sur les bonnes pastilles. Il est également utile d'enfiler la connexion à travers les trous sur la carte avant de souder en place - cela fournit une articulation beaucoup plus forte qui est moins susceptible de casser.

Alimentation Régulée.
Certains utilisateurs peuvent souhaiter utiliser une alimentation de style 'chargeur de portable' (par exemple une PWR009). Il est essentiel qu'un dispositif régulé 9V DC de bonne qualité soit utilisé avec un régulateur de 5V. Des dispositifs non réglementés peuvent donner des tensions excessives (dans des conditions de faible charge) qui endommageront le microcontrôleur. Les anciennes alimentations d'ordinateurs 12V/5V ne sont pas adaptées pour les travaux avec le microcontrôleur PICAXE

L'alimentation 9V DC doit être régulée à 5V en utilisant un régulateur de tension (par exemple 7805 (capacité de 1A) ou 78L05 (capacité de 100mA)). Le circuit de régulation complet est illustré ci-dessous. La diode 1N4001 offre une protection d'inversion de connexion, et les condensateurs aident à stabiliser l'alimentation 5V. Notez que les régulateurs de tension ne fonctionnent pas correctement à moins que l'alimentation d'entrée dans ce circuit soit d'environ 8V ou plus généralement. Les condensateurs indiqués sont également essentiels.

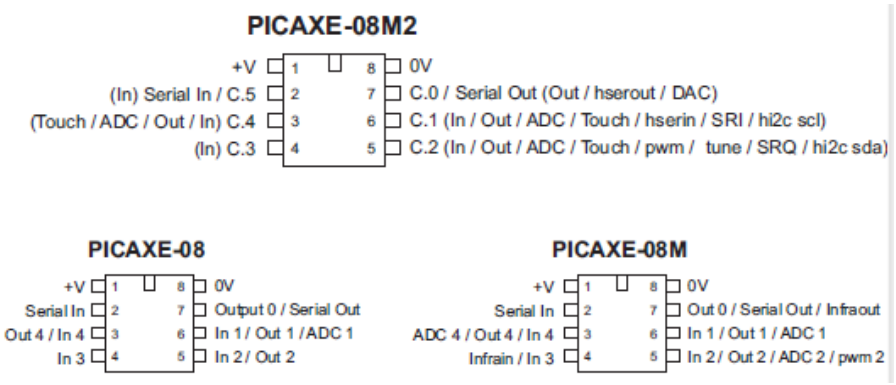
Ne jamais essayer d'utiliser une batterie 9V PP3 avec ce circuit, La batterie PP3 a une capacité en courant insuffisante et n'est recommandée pour aucun projet avec des PICAXES.



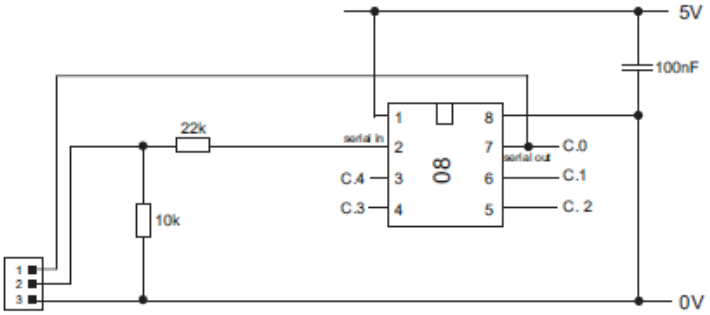
NOTE IMPORTANTE : Ce manuel décrit l'utilisation d'éléments standards (3-5V), les anciens composants X2 sont aussi disponibles avec une variante faible tension (1,8 à 3,3V). L'utilisation d'une alimentation 5V sur un composant 3,3V l'endommagera de façon permanente !

PICAXE-08M2/08M/08 Brochage et Circuit

Le diagramme des pins pour les composants 8 pins est le suivant (0.3" DIL ou 150mil SOIC)



Le circuit minimum pour les équipements 8 pins est :



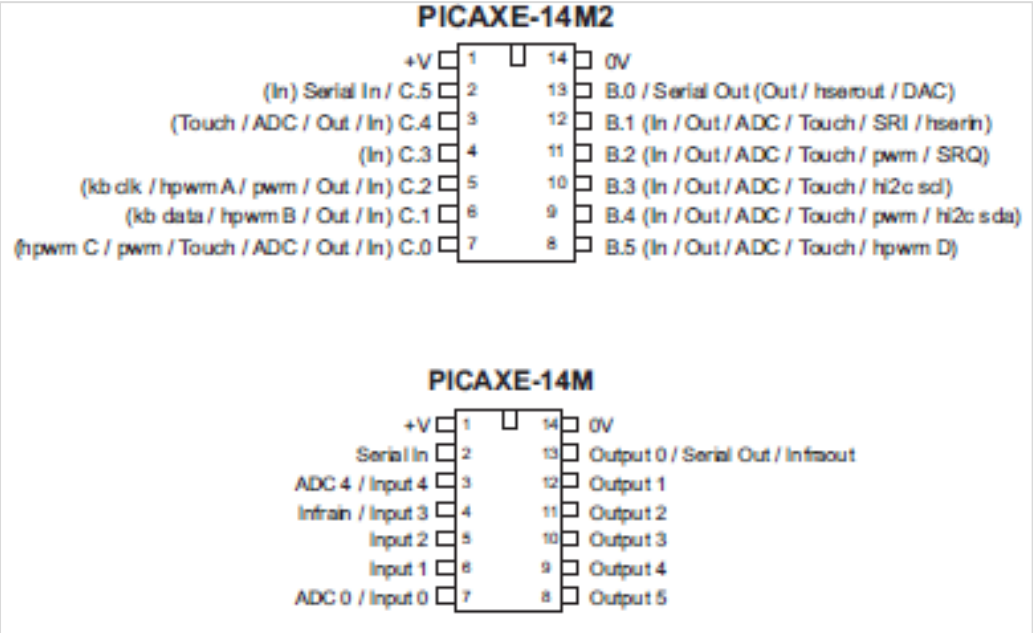
Voir la section Circuit Série de Téléchargement de ce manuel page 44 pour plus de détails sur le circuit de téléchargement.

Notes :
1) Les résistances 10kΩΩ/22kΩΩ doivent être incluses pour un fonctionnement fiable.
NE PAS laisser la pin 'sérial in' flottante sinon le programme ne fonctionnera pas!
2) La pin de sortie 0 (patte 7) est utilisée pendant le téléchargement du programme, mais peut également être utilisée en tant que sortie à usage général, une fois le téléchargement terminé. Sur les cartes projets un cavalier permet de connecter cette pin du microcontrôleur soit à la prise de téléchargement (position PROG) ou à la sortie (position OUT). Rappelez-vous de placer le cavalier dans la position correcte lorsque vous testez votre programme!

Si vous faites votre propre circuit imprimé, vous pouvez inclure un cavalier similaire ou petit inverseur, ou vous préférerez peut-être connecter la pin du microcontrôleur à la fois au dispositif de sortie et à la prise de programmation. Dans ce cas, vous devez vous rappeler que votre périphérique de sortie passera rapidement de On à Off lorsque le téléchargement a lieu (ce n'est pas un problème avec des sorties simples comme les LED, mais pourrait causer des problèmes avec d'autres périphériques tels que des moteurs).

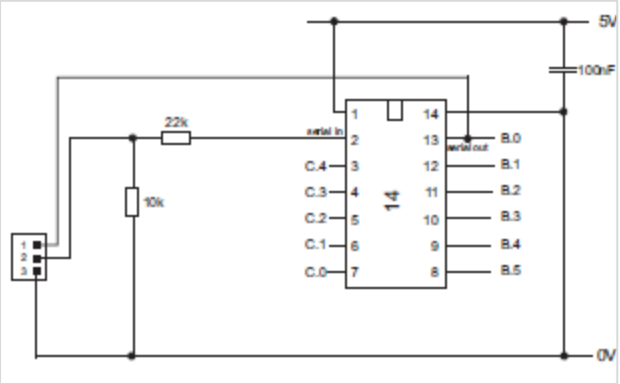
PICAXE-14M2/14M Brochage et Circuit

Le diagramme des pins pour les composants 14 pins est le suivant (0.3" DIL ou 150mil SOIC)



Voir l'Annexe C pour les informations de configuration des E/S des 14M pour des utilisateurs avertis

Le circuit minimum pour les équipements 14 pins est le suivant :



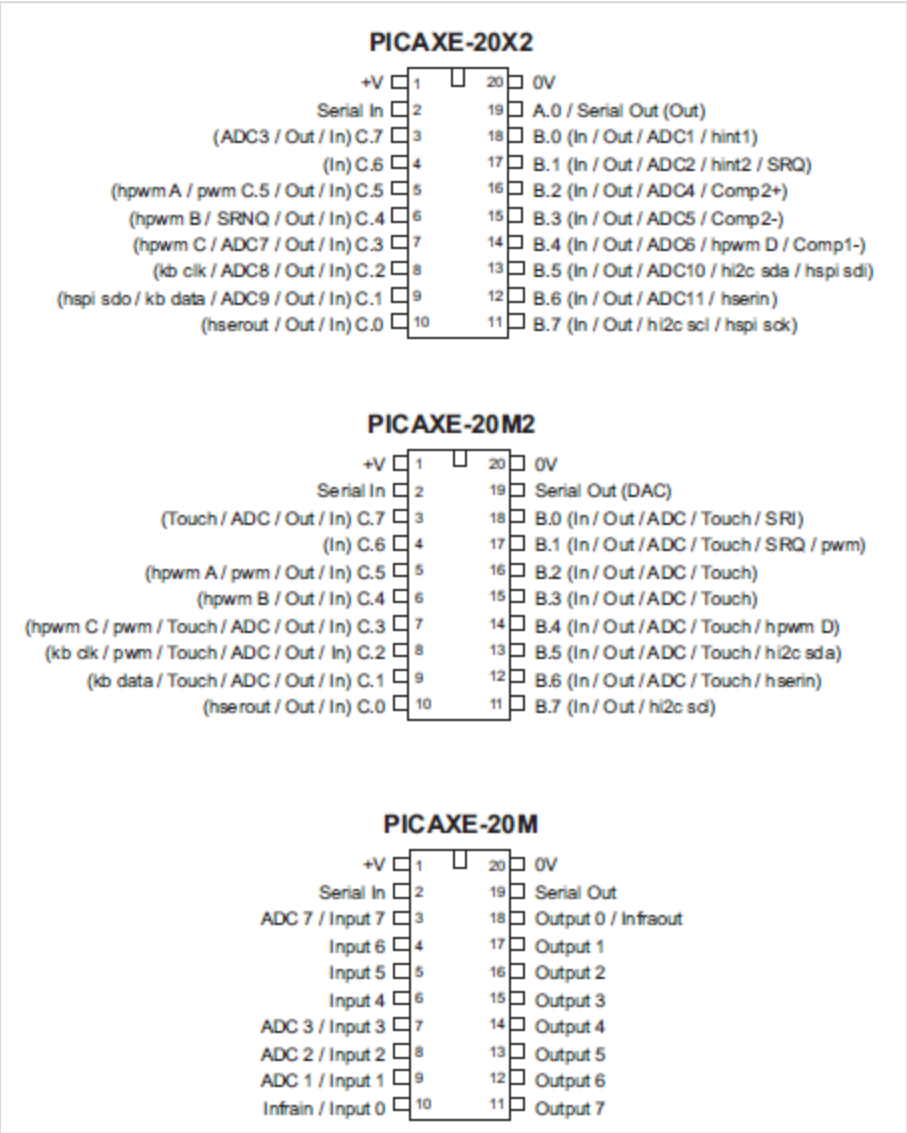
Voir la section Circuit Série de Téléchargement de ce manuel page 44 pour plus de détails sur le circuit de téléchargement.

Notes :
1) Les résistances 10kΩΩ/22kΩΩ doivent être incluses pour un fonctionnement fiable.
NE PAS laisser la pin 'sérial in' flottante sinon le programme ne fonctionnera pas!
2) La pin de sortie 0 (patte 7) est utilisée pendant le téléchargement du programme, mais peut également être utilisée en tant que sortie à usage général, une fois le téléchargement terminé. Sur les cartes projets un cavalier permet de connecter cette pin du microcontrôleur soit à la prise de téléchargement (position PROG) ou à la sortie (position OUT). Rappelez-vous de placer le cavalier dans la position correcte lorsque vous testez votre programme!

Si vous faites votre propre circuit imprimé, vous pouvez inclure un cavalier similaire ou petit inverseur, ou vous préférerez peut-être connecter la pin du microcontrôleur à la fois au dispositif de sortie et à la prise de programmation. Dans ce cas, vous devez vous rappeler que votre périphérique de sortie passera rapidement de On à Off lorsque le téléchargement a lieu (ce n'est pas un problème avec des sorties simples comme les LED, mais pourrait causer des problèmes avec d'autres périphériques tels que des moteurs).

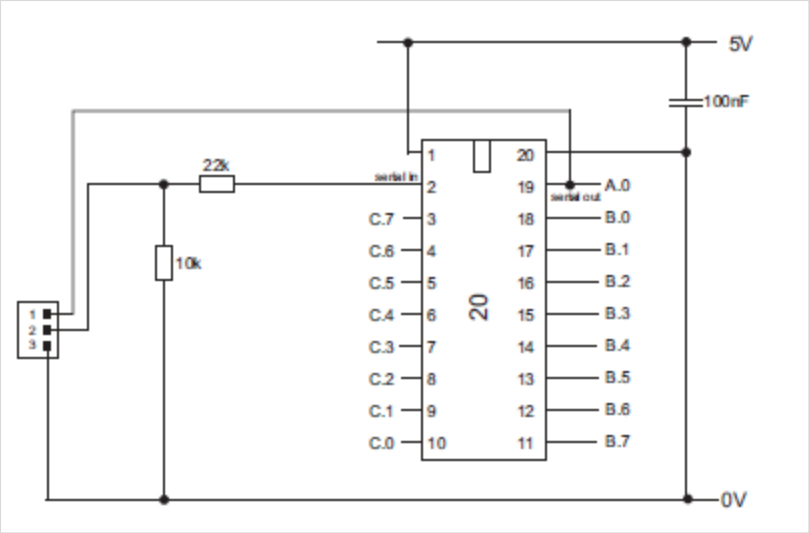
PICAXE-20X2/20M2/20M Brochage et Circuit

Le diagramme des pins pour les composants 20 pins est le suivant (0.3" DIL ou 150mil SOIC)



Notez que la pin C, 6 est seulement une entrée sur les références 20M2 et 20X2. Cela est dû à la conception interne de la puce de silicium et ne peut être modifié.

Le circuit minimum pour les équipements 20 pins est le suivant :

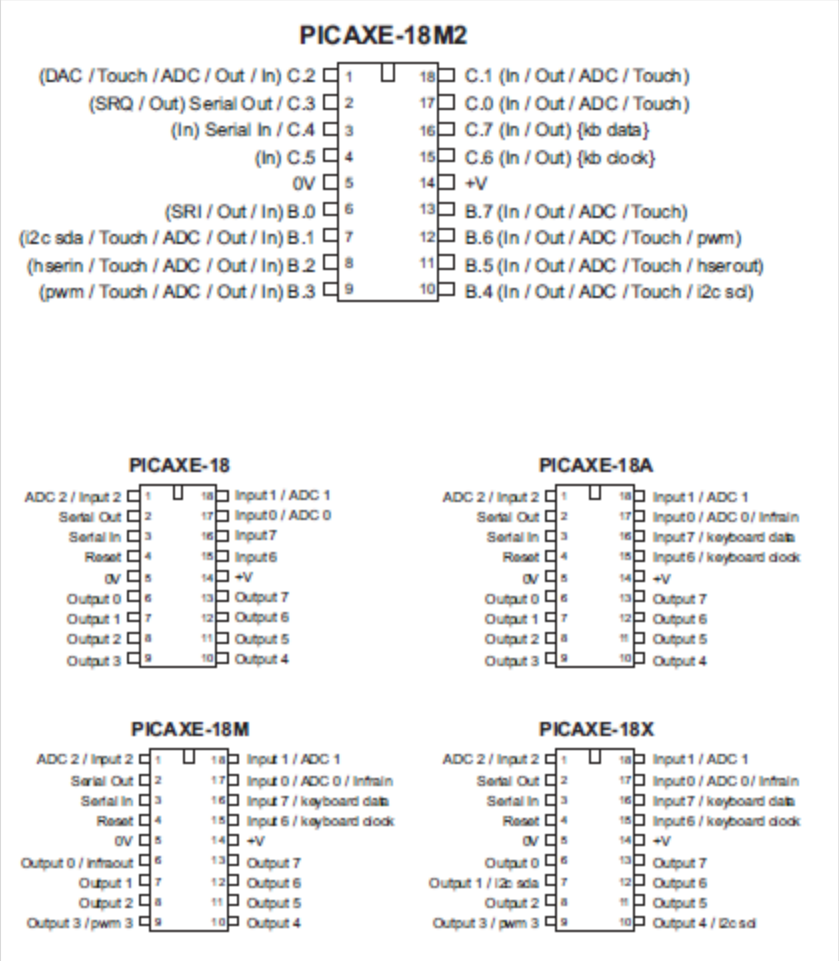


Voir la section Circuit Série de Téléchargement de ce manuel page 44 pour plus de détails sur le circuit de téléchargement.

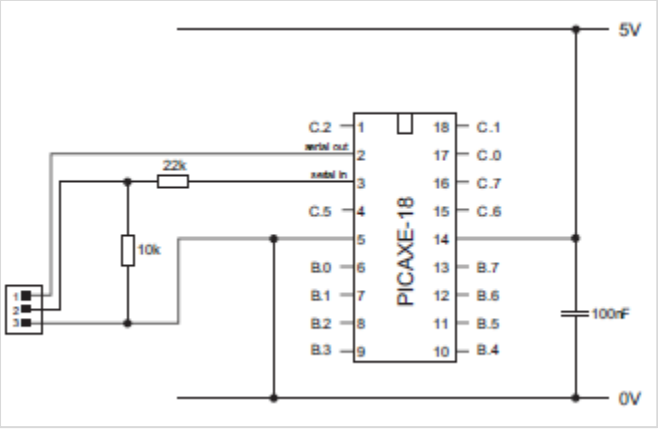
Notes : Les résistances 10kΩΩ/22kΩΩ doivent être incluses pour un fonctionnement fiable.NE PAS laisser la pin 'sérial in' flottante sinon LE PROGRAMME NE FONCTONNERA PAS!

PICAXE-18M2/18X/18M/18A/18 Brochage et Circuit

Le diagramme des pins pour les composants 14 pins est le suivant (0.3" DIL or 300mil SOIC)



Le circuit minimum pour les équipements 18 pins est le suivant :



Voir la section Circuit Série de Téléchargement de ce manuel page 44 pour plus de détails sur le circuit de téléchargement.

Notes :

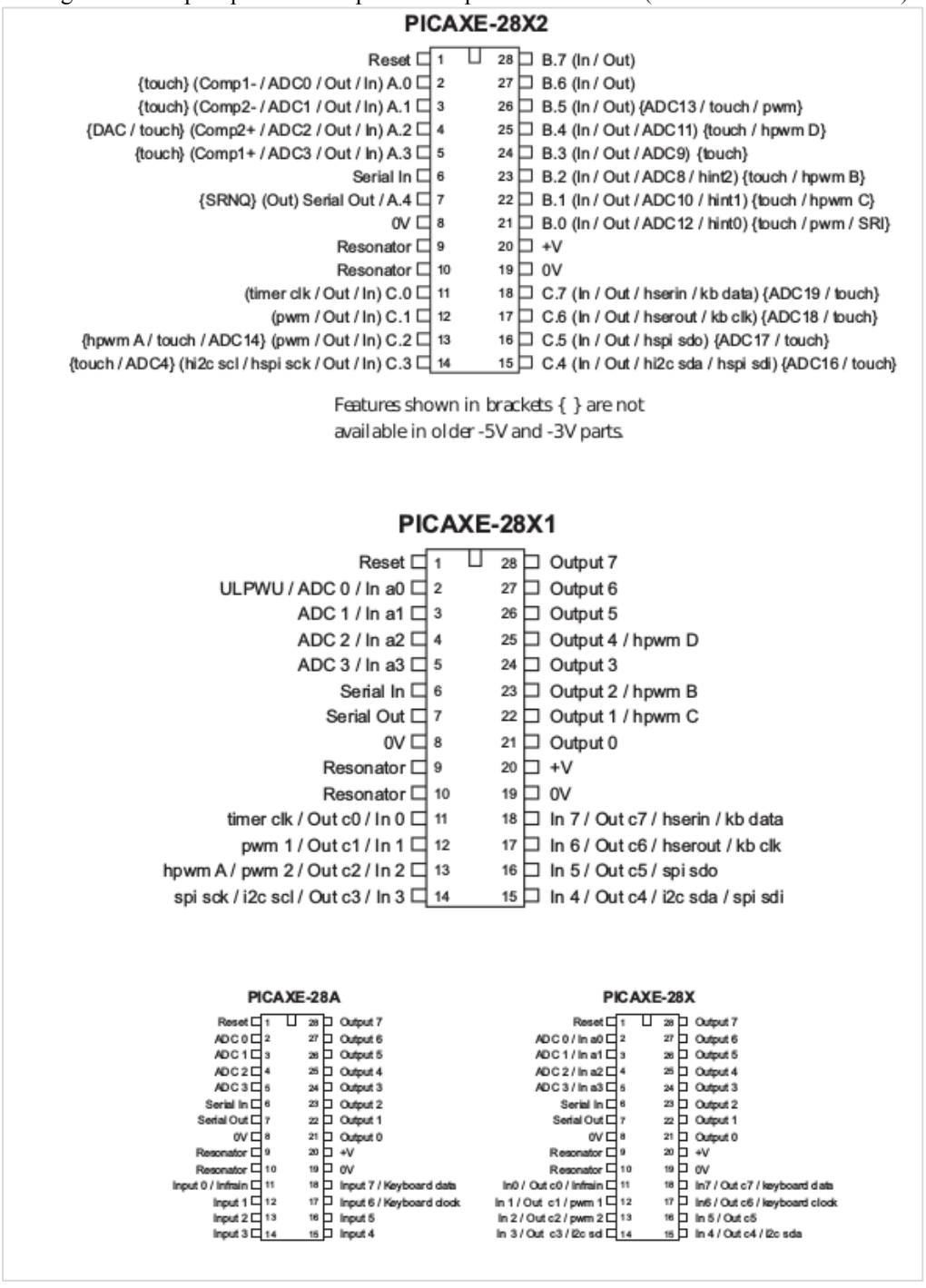
1) Les résistances 10kΩΩ/22kΩΩ doivent être incluses pour un fonctionnement fiable.NE PAS laisser la pin 'sérial in' flottante sinon LE PROGRAMME NE FONCTONNERA PAS!

2) La pin 'reset' doit être reliée à **Haut** par une résistance de 4,7kΩ pour fonctionner sur les références non-M2. Sur les références 18M2 il n'y a pas de pin 'reset'. C'est une entrée à usage général.

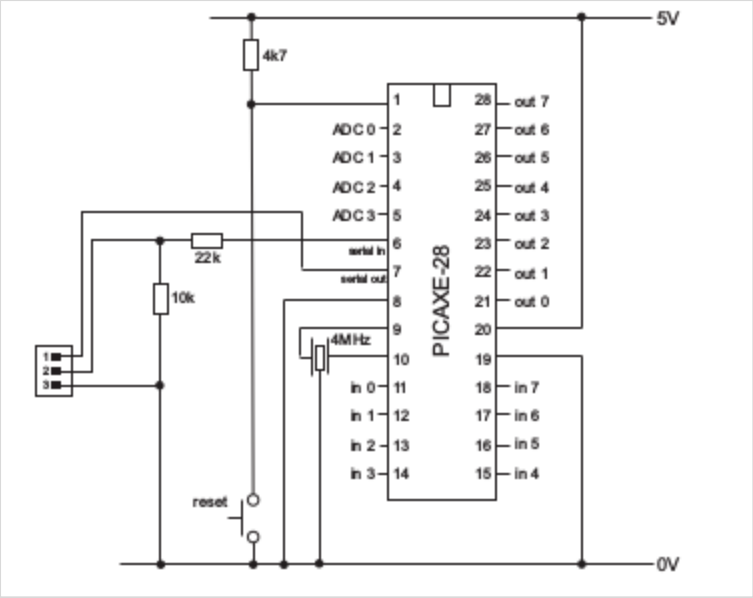
3) Aucun résonateur n'est requis pour ces composants, ils contiennent un résonateur interne.

PICAXE-28X2/28X1/28X/28A Brochage et Circuit

Le diagramme des pins pour les composants 28 pins est le suivant (0.3” DIL or 300mil SOIC)



Le circuit minimum pour les équipements 28 pins est le suivant :



Voir la section Circuit Série de Téléchargement de ce manuel page 44 pour plus de détails sur le circuit de téléchargement.

Notes :
1) Les résistances 10kΩΩ/22kΩΩ doivent être incluses pour un fonctionnement fiable.NE PAS laisser la pin 'sérial in' flottante sinon LE PROGRAMME NE FONCTONNERA PAS!
2) La pin 'reset' doit être reliée à **Haut** par une résistance de 4,7kΩ pour fonctionner.
3) Résonateur :

28X2	(option)	4 (16), 8 (32), 10 (40) ou 16 (64) MHz
28 X2-5V	(option)	4 (16), 8 (32), 10 (40) MHz
28X2-3V	(option)	4 (16), 8 (32), 10 (40) ou 16 (64) MHz
28X1	(option)	16 MHz
28X		4, 8 ou 16MHz
28/28A		4 MHz

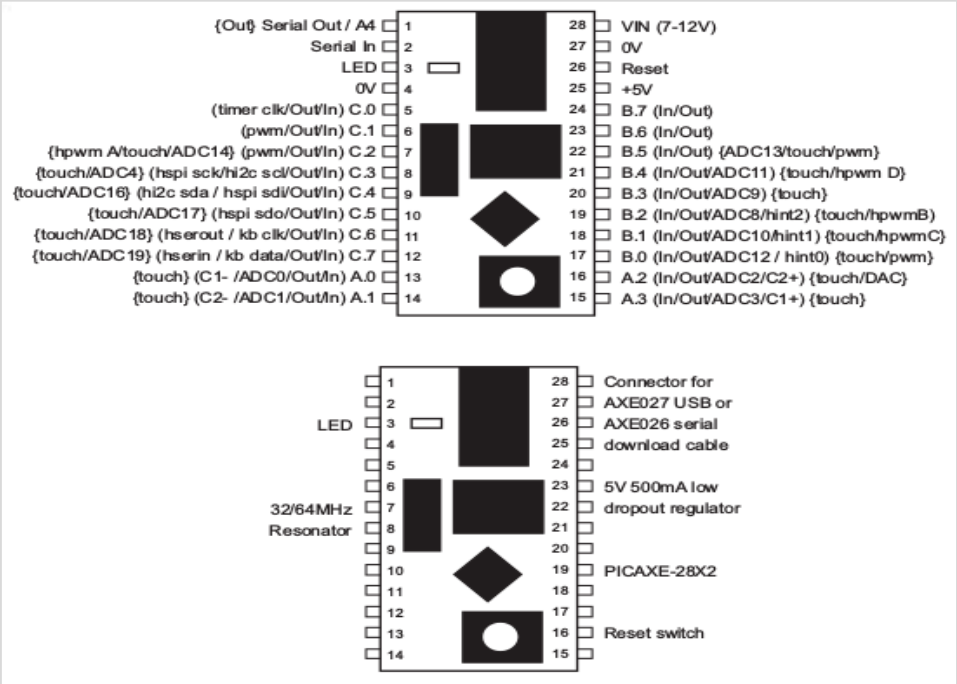
Les 28X1 et X2 ont un résonateur interne (4 ou 8MHz) et ainsi le résonateur externe est optionnel. Pour les références 28A et 28X il est obligatoire.
Le 28X2 a un circuit interne 4xPLL. Celui-ci multiplie la vitesse d'horloge externe par 4. Par conséquent, un oscillateur externe de 8 MHz donne une fréquence réelle d'horloge interne 4x8MHz = 32MHz.

NOTE IMPORTANTE : *Ce manuel décrit l'utilisation des références de la gamme standard (3-5V). Les références X2 sont aussi disponibles avec une variante basse tension (1,8-3,3V). L'utilisation d'une alimentation 5V sur une référence 3,3V l'endommagera.*

PICAXE-28X2 Module (AXE200/AXE201)

Le module 28X2 est un circuit PICAXE complet comportant un package DIN 28 pins. Le module est destiné à être placé dans un support de style circuit intégré sur la carte de projet final de l'utilisateur

Module PICAXE 28X2
28X2
module AXE200



Notes
Le Module est alimenté par un connecteur support 28 pins. Il est fortement recommandé que le module soit laissée sur le support en permanence – c'est à dire utiliser un support distinct sur la carte projet. Ensuite, si une patte est accidentellement cassée net sur le connecteur support, il est possible de la retirer très soigneusement et remplacer la patte du connecteur du support économique.

AXE201/200
Le AXE201 et AXE202 sont physiquement identiques à part la puce et le résonateur

AXE201	**AXE200**
28X2 PICAXE	28X2-5V PICAXE
Résonateur 16MHz	Résonateur 8MHz
(= 64MHZ en exploitation)	(= 32 MHZ en exploitation)

L'AXE201 prend également en charge les nouvelles fonctionnalités supplémentaires PICAXE-28X2 telles que des capteurs tactiles et les canaux analogiques/pwm supplémentaires. Ils sont signalés dans les accolades {} ci-dessus.

Alimentation
L'alimentation peut être de 7- 12v DC via la pin 28. Elle est régulée par régulateur 5V. La tension 5 V est disponible sur la pin 25. Une alimentation de 4,5V ou 5V peut être connectée directement à la pin 25, laissant la 28 non connectée.

Bouton Reset
Il y a un bouton reset sur la carte (avec une résistance de tirage Haut de 4,7kΩ. Le module peut aussi être réinitialisé en connectant la pin 26 à 0V.

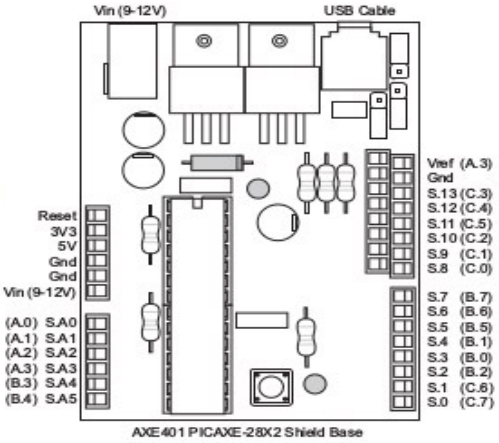
Téléchargements
Le téléchargement peut être fait par le connecteur série (câble de téléchargement série AXE027 USB ou AXE026) ou par les pins Entrée Série/Sortie Série.

LED

La pin LED (pin 3) connecte une LED et résistance de 330Ω quand elle est connectée au 0V. Si laissée non connectée, la LED ne sera pas opérationnelle, et ne consomme donc pas de courant (parfois souhaitable dans les systèmes à base de batteries). Pour utiliser la LED comme témoin d'alimentation connectez simplement la borne LED (pin 3) au 5V (pin 25). La pin LED peut aussi être connectée à une borne de sortie et sera donc contrôlée par des niveaux haut ou bas du programme utilisateur.

PICAXE-28X2 Shield Base (AXE401)

Le module de base 28X2-PICAXE est un circuit PICAXE Open Source dans le format populaire ' Shield' qui permet la connexion des shields tiers (pilote moteur, Ethernet,GSM, etc.). Il est fourni avec un une carte shield prototype libre correspondante.



Alimentation
L'alimentation 9-12V DC peut être fournie par le connecteur d'alimentation de 2,1 mm. Elle est ensuite régulée par un régulateur à faible chute de tension 5V - 500mA. Un autre régulateur fournit aussi du 3,3V. Le système d'alimentation peut être sélectionné soit en 5V soit en 3,3V, le PICAXE-28X2 fonctionnera avec chaque tension. Toutes les connexions électriques sont également disponibles sur la barrette d'alimentation 6 broches.

Bouton Reset
Il y a un bouton 'reset' sur la carte (avec une résistance 4,7kΩ de tirage à Haut). Le module peut être réinitialisé en connectant la pin 'reset' à 0V.

Téléchargements
Le téléchargement peut être fait via le jack (le câble de téléchargement sérieAXE027 USB ou AXE026). Les Broches de téléchargement sont séparées des pins E/S.

LED utilisateur
La LED se connecte à la pin S.13 (et par une résistance quand elle est connectée au 0V). Si le cavalier H4 est laissé déconnecté, la LED est déconnectée, et ne consomme pas de courant (parfois utile lors d'alimentation sur batteries). Pour utiliser la LED ajouter un cavalier H4 pour connecter la LED à la pin de sortie S.13. La LED peut alors être commandée par des niveaux hauts et bas du programme utilisateur.

Entrées/Sorties
Les connexions d'entrée/sortie ont été très soigneusement conçues pour correspondre à presque tous les shields tiers, y compris les shields de conceptions avancés qui utilisent la communication SPI, sorties PWM ou la communication série haute vitesse. La notation des pins shield généralement utilisée est A0-A5 et 0-13, et ainsi le compilateur PICAXE-28X2 comprend également cette notation en acceptant les surnoms de pin du shield (S.A0, S.A1, S.1, S.2, etc.), ainsi que le nom d'origine des pins PICAXE.

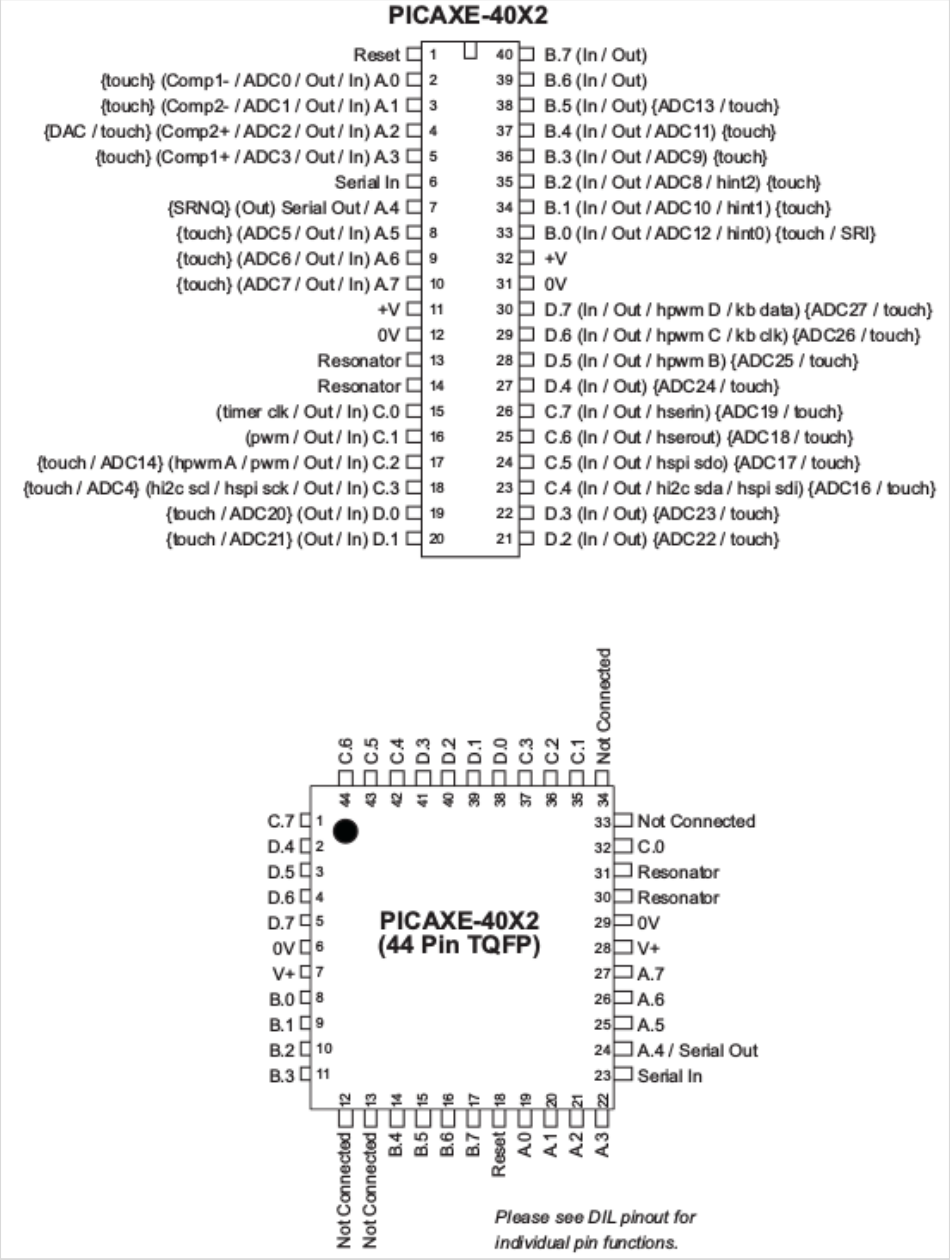
Shield Header	Shield Nickname	Primary Pin Function	Advanced Pin Function	PICAXE Pin Name	PICAXE ADC
RESET		Reset		Reset	
3V3		3.3V Supply Out		V+	
5V		5V Supply Out	5V Supply In	V+	
GND		0V	0V Supply In	0V	
GND		0V		0V	
VIN		Supply In (9-12V DC)			
A0	S.A0	In / Out / ADC / Touch	Comp1-	A.0	0
A1	S.A1	In / Out / ADC / Touch	Comp2-	A.1	1
A2	S.A2	In / Out / ADC / Touch	Comp2+ / DAC	A.2	2
A3	S.A3	In / Out / ADC / Touch	Comp1+ / Vref	A.3	3
A4	S.A4	In / Out / ADC / Touch		B.3	9
A5	S.A5	In / Out / ADC / Touch	hpwm D	B.4	11
0	S.0	In / Out / ADC / Touch	hserin / kb data	C.7	19
1	S.1	In / Out / ADC / Touch	hserout / kb clk	C.6	18
2	S.2	In / Out / ADC / Touch	hpwm B / hint 2	B.2	8
3	S.3	In / Out / ADC / Touch	pwm / hint0	B.0	12
4	S.4	In / Out / ADC / Touch	hpwm C / hint 1	B.1	10
5	S.5	In / Out / ADC / Touch	pwm	B.5	13
6	S.6	In / Out		B.6	-
7	S.7	In / Out		B.7	-
8	S.8	In / Out	timer clk	C.0	-
9	S.9	In / Out	pwm	C.1	-
10	S.10	In / Out / ADC / Touch	hpwm A / pwm	C.2	14
11	S.11	In / Out / ADC / Touch	hspi sdo	C.5	17
12	S.12	In / Out / ADC / Touch	hspi sdi / hi2c sda	C.4	16
13	S.13	In / Out / ADC / Touch (or LED via H4)	hspi sck / hi2c scl	C.3	4
GND		0V		0V	
VREF	S.A3	In / Out / ADC / Touch	Comp1+ / Vref	A.3	3

Pour plus de détail sur le système shield 28X2-PICAXE voir les feuilles de données librement disponibles à :

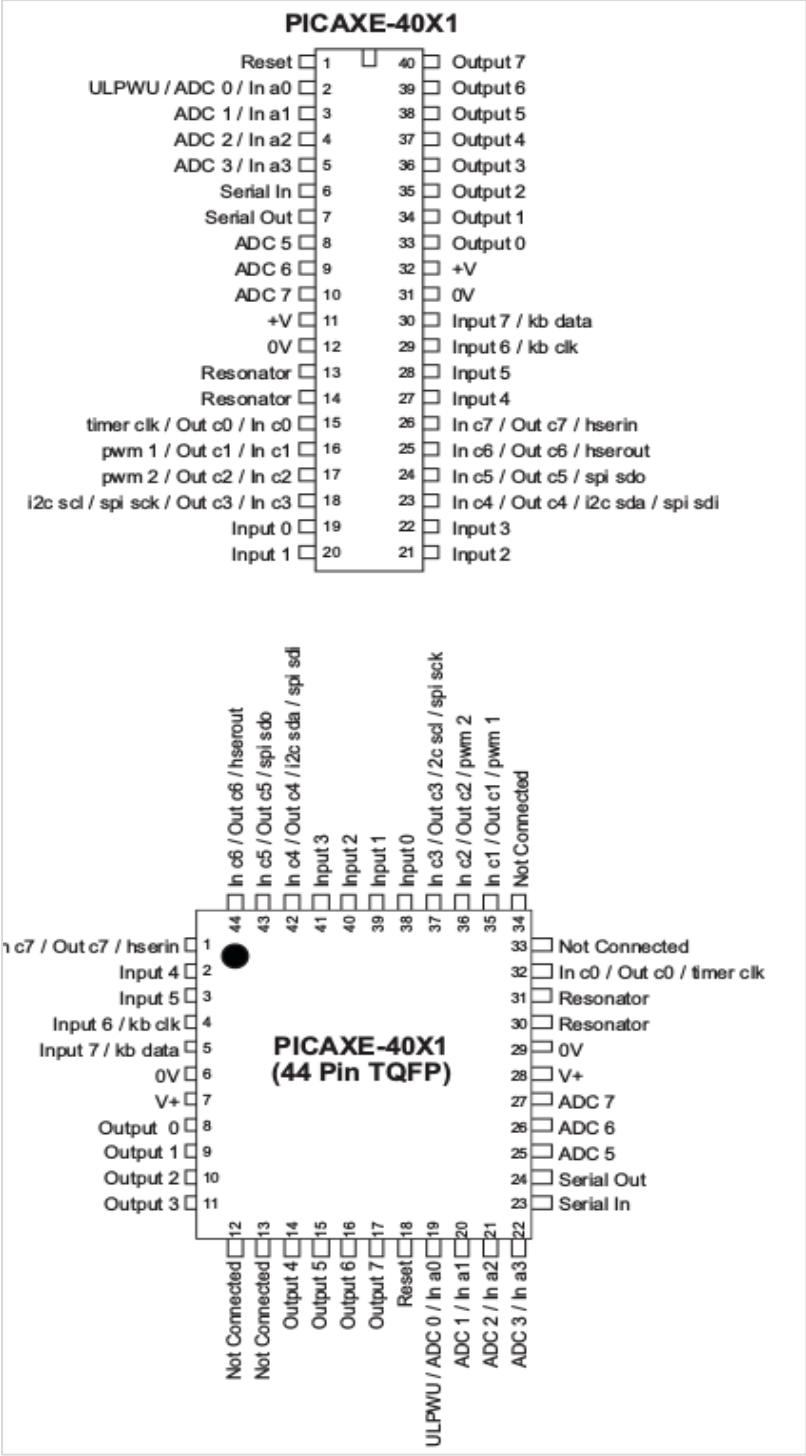
www.rev-ed.co.uk/docs/axe401.pdf

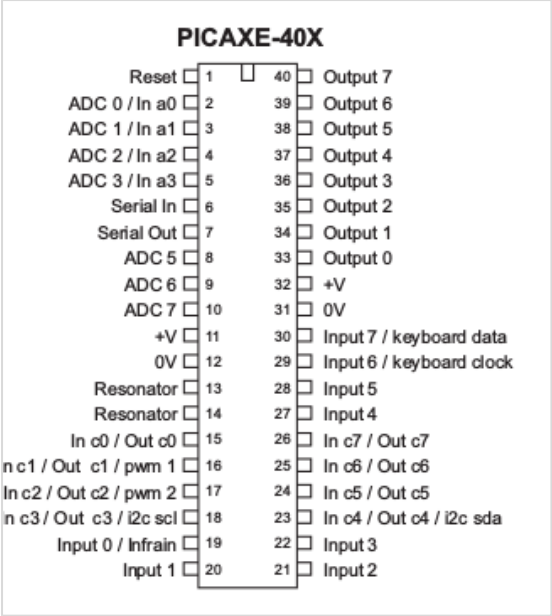
PICAXE-40X2/40X1/40X Brochage et Circuit

Le diagramme des pins pour les composants 48 pins est le suivant (0.3” DIL or 44 pin TQFP)



Les fonctions représentées entre accolades {} ne sont pas disponibles sur les anciennes références en 5V et 3V.





Le circuit minimum pour les équipements 40 pins est le même que le circuit minimum 28 pins (modifier les numéros de pins appropriées au besoin).

Voir la section Circuit de Téléchargement Série p44 de ce manuel pour plus d'informations sur le circuit de téléchargement.

Notes :
1) Les résistances 10kΩ/22kΩ doivent être incluses pour un fonctionnement fiable.
NE PAS laisser l'entrée série flottante LE PROGRAMME NE FONCTIONNERA PAS
2) La pin reset doit être tirée vers le haut par une résistance de 4,7kΩ pour être fonctionnelle.
3) Résonateur :

40X2	(option)	4 (16), 8 (32), 10 (40), 16 (64) MHz
40X2-5V	(option)	4 (16), 8 (32), 10 (40) MHz
40X2-3V	(option)	4 (16), 8 (32), 10 (40), 16 (64) MHz
40X1	(option)	16 MHz
40X		4, 8 ou 16 MHz

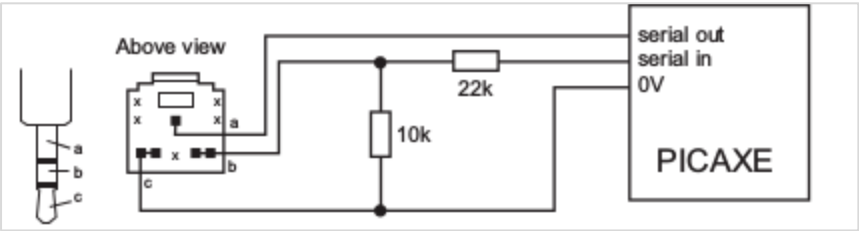
Les 40X1 et X2 ont un résonateur interne (4 ou 8 MHz) et ainsi le résonateur externe est optionnel. Sur la référence 40X il est obligatoire.
Le 40X2 a un circuit PLL x4. Ceci multiplie la vitesse d'horloge externe par 4. Par conséquent, un résonateur à 4MHz externe donne une fréquence d'horloge interne réelle de 16 MHz = 4x4MHz.

***NOTE IMPORTANTE**- ce manuel décrit l'utilisation des références de la gamme standard (3-5V) . Les référence X2 les plus anciennes étaient également disponibles dans une variante à faible tension (1,8V à 3,3V).L'utilisation d'un 5V sur ces références spéciale de 3.3V les endommagera de façon permanente!*

Circuit de Téléchargement USB

Le circuit de téléchargement USB est identique pour toutes les puces PICAXE. Il consiste en 3 fils depuis la puce PICAXE jusqu'au câble USB AXE027. Un fil envoie les données depuis l'ordinateur vers l'entrée série du PICAXE, un fil envoie les données depuis la sortie série du PICAXE vers l'ordinateur et le troisième fil est une masse commune.

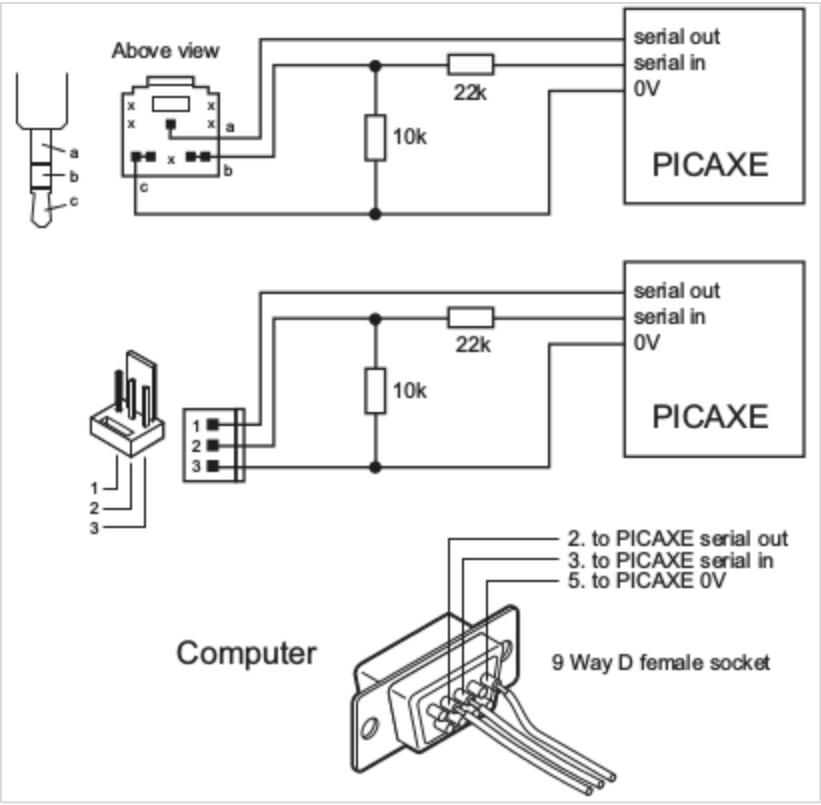
Noter que ce circuit peut aussi être utilisé pour le câble série AXE026. Par conséquent, le même circuit peut être utilisé avec un câble USB ou série.
Le circuit minimum de téléchargement est montré ici :



Noter que les deux résistances ne sont pas un diviseur de tension. La résistance 22kΩ travaille avec les diodes internes du microcontrôleur pour verrouiller la tension série avec la tension d'alimentation du PICAXE et pour limiter le courant de téléchargement à une valeur acceptable. La résistance de 10kΩ empêche l'entrée série d'être 'flottante' quand le câble de téléchargement n'est pas connecté. Ceci est essentiel pour un fonctionnement fiable.

Circuit Série de Téléchargement Série

Le circuit de téléchargement série est identique pour toutes les puces PICAXE. Il consiste en 3 fils depuis la puce PICAXE jusqu'au câble sérieAXE026. Un fil envoie les données depuis l'ordinateur vers l'entrée série du PICAXE, un fil envoie les données depuis la sortie série du PICAXE vers l'ordinateur et le troisième fil est une masse commune. Voir la section adaptateur USB pour plus de détails sur la façon d'utiliser l'adaptateur port USB.

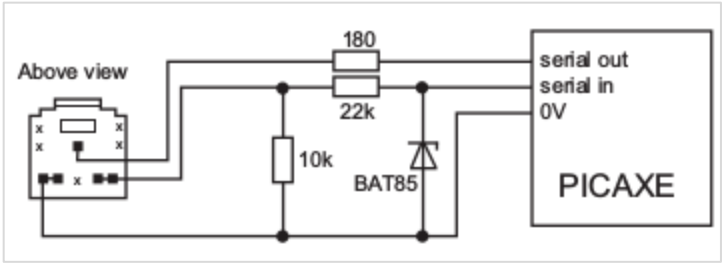


Noter que les deux résistances ne sont pas un diviseur de tension. La résistance 22kΩ travaille avec les diodes internes du microcontrôleur pour verrouiller la tension série avec la tension d'alimentation du PICAXE et pour limiter le courant de

téléchargement à une valeur acceptable. La résistance de 10kΩ empêche l'entrée série d'être 'flottante' quand le câble de téléchargement n'est pas connecté. Ceci est essentiel pour un fonctionnement fiable.

Les deux résistances doivent être incluses sur tous les circuits PICAXE (à savoir non intégrées dans le câble). L'entrée série ne doit jamais être laissée sans connexion. Si elle est laissée sans connexion l'entrée série sera 'flottante' haut ou bas et provoquer un fonctionnement non fiable, la puce PICAXE recevra des signaux parasites flottants qu'elle peut interpréter comme une nouvelle tentative de téléchargement.

Amélioration du Circuit Série de Téléchargement (NB : Obsolète, pour info)



La diode shokkty BAT85 agit à une tension de l'équipement plus faible que les diodes internes du microcontrôleur, fournissant une référence de tension plus précise. La résistance 180R supplémentaire permet une prévention des courts-circuits et une protection statique sur la pin de sortie série.

Non requis lorsque le câble AXE027 USB est utilisé.

Câbles de Téléchargement

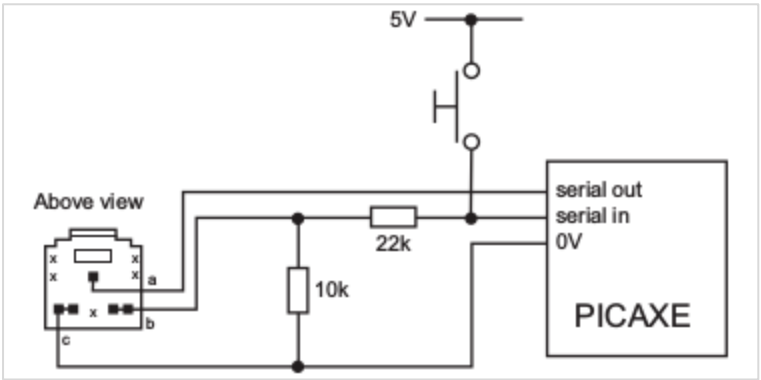
Le câble USB de téléchargement (AXE027) est recommandé pour tous les ordinateurs modernes. Il est compatible avec n'importe quel ordinateur comportant un port USB.

Le plus vieux câble (AXE026) série de téléchargement consiste en un jack stéréo de 3,5mm avec un connecteur stéréo (CON039) sur la carte projet. Ce type de connexion est plus robuste et fiable que les bornes Molex utilisées dans l'environnement éducatif.

Toutes les connexions traditionnelles série des ordinateurs sont via le port série (connecteur sub D 9), si vous avez un ordinateur très vieux avec un port série 25 pts un adaptateur 25/9 (ADA010) est nécessaire, il est disponible dans la plupart des magasins informatiques.

Utilisation de la pin Serial In comme une pin entrée générale

Sur les références M2 et X2 la pin 'serial in' peut être utilisée comme une entrée générale, connectée à un bouton-poussoir comme montré ci-dessous.



1. Toutefois il y a certaines conditions spéciales pour cette utilisation.
2. Le programme doit contenir une commande 'disconnected'. Cette commande protège la puce PICAXE contre un scanning de la pin 'serial in' pour un nouveau téléchargement. Si vous n'avez pas ajouté cette commande, le PICAXE se réinitialisera lors d'un appui sur le bouton-poussoir.
3. Après qu'une commande 'disconnected' ait été utilisée, il sera nécessaire de réaliser une mise sous tension 'par une réinitialisation matériel'.
4. Le bouton-poussoir doit être ouvert pendant un nouveau téléchargement de programme.

En raison de ces exigences particulières, il est généralement préférable, si possible, de réserver la pin 'serial in' pour l'utilisation dédiée de programmation. A utiliser seulement comme une entrée à usage général lorsque toutes les autres pins sont déjà utilisées.

Circuit de Réinitialisation

Tous les PICAXE 18, 28 et 40 pins (et quelques 18 pin antérieures) ont une pin 'reset'. Cette pin doit être à un niveau Haut pour que le microcontrôleur PICAXE fonctionne. Si cette pin est laissée déconnectée, le microcontrôleur n'aura pas un fonctionnement fiable. Pour tirer cette pin vers un niveau Haut, connecter une résistance de 4,7kΩ entre la pin 'reset' et le V+ d'alimentation (ne pas connecter directement la pin, toujours utiliser une résistance). Un bouton 'reset' est optionnel mais fortement recommandé. Ce sera un 'poussoir à fermeture' et connecté entre la pin 'reset' et le 0V

Toutes les références PICAXE de 8, 14 et 20 pins (et 18M2) ne disposent pas d'une pin de réinitialisation. Par conséquent, pour réinitialiser le microcontrôleur l'alimentation doit être débranchée puis rebranchée. Notez que, lors de l'utilisation de condensateurs dans votre circuit d'alimentation, ces condensateurs peuvent détenir une charge suffisante pour maintenir le microcontrôleur alimenté pendant plusieurs secondes après que l'alimentation électrique ait été coupée.

Résonateur

Les différentes puces PICAXE ont un résonateur externe ou interne (ou les deux)

PICAXE	INTERNE	EXTERNE
08, 18	4	-
'A' parts	4,8	-
'M' parts	4,8	-
'X' parts	4,8	-
'M2' parts	4,8,16,32	-
20X2	4,8,16,32,64	-
Discontinued old parts:		
28A	-	4
28X	-	4,8,16
28X1	4,8	4,8,16
28X2-5V	4,8	4 (=16), 8 (=32), 10 (=40)
28X2-3V	4,8,16	4 (=16), 8 (=32), 10 (=40), 16 (=64)
40X	-	4,8,16
40X1	4,8	4,8,16
40X2-5V	4,8	4 (=16), 8 (=32), 10 (=40)
40X2-3V	4,8,16	4 (=16), 8 (=32), 10 (=40), 16 (=64)

Tous les PICAXE 28 et 40 pins peuvent utiliser un résonateur externe (le résonateur est interne pour les PICAXE 08, 14, 20 et 18 pins). Notez que le résonateur interne au sein du PICAXE 08, 14, 20 et 18 n'est pas tout à fait aussi précis qu'un résonateur externe. Bien que cela ne cause pas de problèmes avec la majorité des projets, si un projet spécialisé nécessite une très grande précision un PICAXE 28 ou 40 pins doit être utilisé.

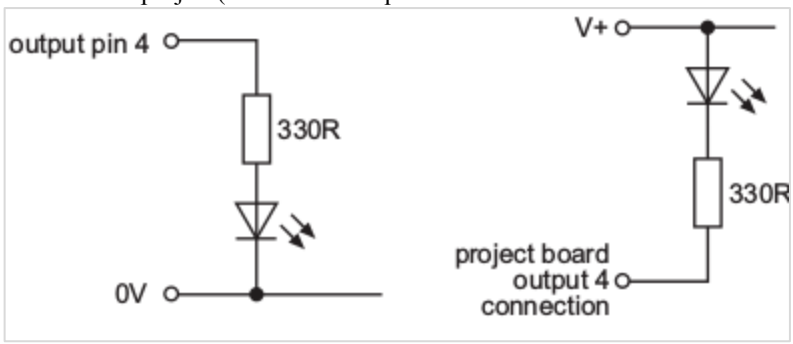
Un résonateur céramique 3 pins est recommandé en cas de besoin. Ce dispositif est constitué d'un résonateur et deux condensateurs de chargement dans un boîtier 3 pins seulement. La pin centrale est reliée à 0 V et les deux pins extérieures aux deux pins du résonateur (le résonateur PICAXE peut être utilisé d'une façon ou d'une autre).

Toutes les références fonctionnent par défaut avec un 4MHz interne, en dehors des références X2 qui fonctionnent par défaut avec un 8 MHz interne.

Le 28X2 et 40X2 contiennent un circuit interne PLL x4. Cela signifie que la fréquence de fonctionnement interne est 4x la fréquence du résonateur externe. La vitesse maximale de ces dispositifs est donc 64MHz (en utilisant un résonateur à 16 MHz).
Si on le désire un résonateur à 2 pins ou un cristal a deux pins, peuvent être utilisés avec les extensions X, X1 ou X2. Dans ce cas deux condensateurs de chargement appropriés doivent également être utilisés avec le résonateur / cristal. Voir la fiche technique du fabricant du cristal pour plus d'informations.

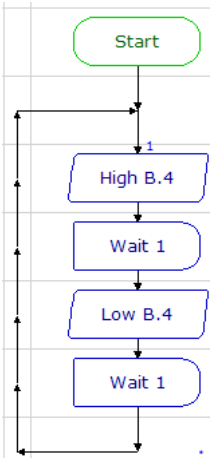
Test du système

Ce premier programme simple peut être utilisé pour tester votre système. Il exige la connexion d'une LED (et résistance 330Ω) à la pin de sortie 4. Si vous reliez la LED directement à une puce PICAXE sur un proto (ou circuit-maison), connecter la LED entre la pin de sortie et 0V. Lorsque vous utilisez les cartes de projet (par exemple, tel que fournies dans les packs de démarrage 18 et 28), connecter la LED entre V + et le connecteur de sortie, la sortie est chargée par la puce du Darlington sur la carte projet. (Assurez-vous que la LED est connectée dans le bon sens!).



1. Connecter le câble PICAXE au port USB/série de l'ordinateur. Notez quel port est connecté (exemple : COM1 ou COM6.
2. Démarrer le logiciel PICAXE Editor6
3. Sélectionner l'onglet puce PICAXE dans l'espace de travail
4. Sélectionner le type de PICAXE correct.
5. Cliquer sur l'onglet ' Port Série' et sélectionner le port série (port COM virtuel pour câble USB) auquel le câble PICAXE est connecté.
6. Taper le programme suivant :

```
main :    high 4
      a.  pause 1000
      b.  low 4
      c.  pause 1000
      d.  goto main
```



7. (NB noter le symbole (:) directement après le label 'main' et les espaces entre les commandes et les nombres.
8. Assurez-vous que le circuit PICAXE est connecté au câble série et que les piles sont connectées. Assurez- vous que la LED et la résistance de 330 Ohms sont connectées à la sortie B.4.
9. Sélectionnez PICAXE-> Programme
   Une barre de téléchargement doit apparaître comme pour les téléchargements de programmes. Lorsque le téléchargement est terminé, le programme devrait commencer à fonctionner automatiquement - la LED de sortie 4 devrait clignoter chaque seconde.

Si votre programme ne se télécharge pas utiliser la check-list et la procédure de réinitialisation matérielle décrite dans les deux sections suivantes pour isoler l'erreur.

Procédure Réinitialisation-matériel

Le processus de téléchargement demande au microcontrôleur PICAXE de vérifier régulièrement la ligne d'entrée pour détecter la présence d'un nouveau signal de téléchargement envoyé depuis l'ordinateur. Ceci est automatique et ne peut pas être notifié par l'utilisateur. Cependant il peut y avoir de rares occasions pour que le PICAXE ne vérifie pas la ligne d'entrée série assez rapidement pendant le déroulement de son programme. Ces situations peuvent inclure :

- Programme corrompu dans PICAXE (exemple si l'alimentation ou le câble enlevé pour un nouveau téléchargement
- Fréquence d'horloge incorrecte (définie par la commande *setfreq*)
- Pause ou attente de plus de 5 secondes utilisées dans le programme.
- L'utilisation de *serin, infrain* ou *keyin* au sein du programme

Heureusement il est très simple de résoudre ces problèmes, comme la première chose que toute puce PICAXE fait sur une réinitialisation de l'alimentation est de vérifier pour un nouveau téléchargement de l'ordinateur. Par conséquent, si vous réinitialisez PICAXE tandis qu'un téléchargement est en cours de démarrage par l'ordinateur, le nouveau téléchargement va toujours être reconnu. Ce processus est appelé un hard-reset.

Pour effectuer une réinitialisation matériel par l'utilisation de bouton Reset (PICAXE 28, 40 pins) :

1) Appuyez et maintenez enfoncé le bouton de réinitialisation vers le bas.
2) Cliquez sur le menu PICAXE-> Programme pour démarrer un téléchargement
3) Attendre jusqu'à ce que la barre de progression apparaisse à l'écran.
4) Attendre 1 seconde puis relâchez le bouton reset.

Pour effectuer une réinitialisation matériel par l'utilisation de l'alimentation (tout format) :

1) Déconnecter l'alimentation
2) Attendre jusqu'à ce que les capacités de découplages soient déchargées (peut prendre 30 secondes ou plus suivant les circuits).
3) Cliquez sur le menu PICAXE-> Programme pour démarrer un téléchargement.
4) Attendre jusqu'à ce que la barre de progression apparaisse à l'écran.
5) Rebrancher l'alimentation.

Checklist Téléchargement

Si vous ne pouvez pas télécharger votre programme, vérifier les éléments suivants. Rappelez- vous que tous les nouveaux PICAXE sont programmés et testés, toutefois si un nouveau circuit ne peut pas être chargé, la cause est souvent matérielle.
Si le programme échoue au cours d'un téléchargement c'est généralement un problème d'alimentation (ou la perte du câble de connexion). Essayer avec 3 nouvelles piles alcalines donnant exactement 4,5V.

Microcontrôleur PICAXE

- La puce Pixace est-elle bien insérée sur le support
- Est-ce une puce PICAXE qui est utilisée (non une puce PICAXE vierge non programmée)
- Est-ce une puce PICAXE endommagée qui est utilisée (ex : puce qui a subi un survoltage ou une inversion de polarité).
- Est-ce une tension continue entre 4,5 et 5v qui est correctement connectée. TESTER SUR LE CIRCUIT ACTUEL entre le + et le 0V avec un multimètre.
- Est-ce que la pin Reset est connectée au V+ via une résistance de 4,7kΩ (puces 18/28/40 Pins)
- Est-ce que la pin 3 du résonateur est connectée si besoin (puces 28/40 pins)
- Est-ce que les résistances de téléchargement série 10kΩ/22kΩ sont correctement connectées.

Logiciel

- Dernière version de PICAXE Editor 6 installée (voir la page logiciel à www.picaxe.co.uk pour les informations sur les mises à jour) version actuelle 6
- Le port série correctement sélectionné (menu espace de travail-->Configuration--> Port de communication).
- vitesse du résonateur correctement sélectionnée (si approprié) (menu View-->Options--> Mode).

- Aucun logiciel conflictuel sur le port série exécuté par l'ordinateur (en particulier logiciel PDA du type 'hotsync' et logiciel tableau blanc interactif).

Câble Téléchargement Série (AXE026)

- câble de téléchargement correctement câblé.
- Connecteur de téléchargement correctement câblé avec des résistances de 10kΩ/22kΩ.
- Toutes les pins du connecteur correctement soudées au circuit imprimé.
- Le câble de téléchargement correctement connecté entre le microcontrôleur et l'ordinateur.
- Le câble de téléchargement entièrement inséré dans le support.

Câble de Téléchargement USB(AXE027)

- Le câble USB configuré pour utiliser le bon port série
- Câble USB installé avec le bon driver (utilisateurs de Vista/XP- assurez-vous d'utiliser le bon driver spécifique XP (valide aussi pour Vista) disponible depuis la page logiciel www.picaxe.co.uk)

Adaptateur USB (USB010)

- L'adaptateur USB configuré pour utiliser le bon port série
- L'adaptateur USB installer avec le bon driver (utilisateurs de Vista/XP- assurez-vous d'utiliser le bon driver spécifique XP (valide aussi pour Vista) disponible depuis la page logiciel www.picaxe.co.uk)

Comprendre la mémoire PICAXE

La mémoire PICAXE est constituée de trois zones. La taille de la mémoire varie suivant le type de PICAXE

Mémoire Programme

La mémoire programme est la zone où le programme est stocké après un nouveau téléchargement, c'est une mémoire 'FLASH' réinscriptible qui peut être reprogrammée jusqu'à 100 000 fois (typique). Le programme n'est pas perdu quand l'alimentation est supprimée, ainsi le programme peut redémarrer quand l'alimentation est rebranchée.

Il n'est pas toujours requit d'effacer un programme, car chaque téléchargement écrase automatiquement la totalité de l'ancien programme. Cependant si vous voulez stopper un programme en cours d'exécution vous pouvez utiliser le menu PICAXE -->Effacer pour charger un programme 'vide' dans le PICAXE.

Sur les puces PICAXE standards (M2, X, X1) vous pouvez télécharger autour de 600 à 1000 lignes de code BASIC. Sur les références révisions A ou M vous pouvez charger autour de 80 lignes et les références éducatives autour de 40 lignes, les références X2 jusqu'à 4 programmes de 1000 lignes. Noter que ces valeurs sont approximatives car certaines commandes demandent plus de place mémoire. Pour vérifier votre mémoire utilisée, utilisez l'option menu PICAXE--> Vérifier

Sur les références X1et X2, 256 octets de mémoire programme peuvent aussi être réservés comme table de consultation (ex : pour des messages LCD) Voir les commandes *table/readtable* dans le manuel 2 pour plus de détails.

Mémoire Données

La mémoire donnée est un espace de stockage supplémentaire au sein du microcontrôleur. Les données ne sont pas perdues quand l'alimentation est coupée. Chaque téléchargement réinitialise les données à 0 sauf si la commande EEPROM a été utilisée pour des données 'préchargées' dans la mémoire donnée. Voir la description des commandes *read* et *write EEPROM* pour plus de détails.

Dans les PICAXE -08/08M/08M2/14M/20M/18/18M/18M2, la mémoire donnée est 'partagée' avec la mémoire programme. Par conséquent des programmes plus vastes se traduiront par une disposition de zone de mémoire de données plus petite
Dans tous les autres circuits PIXACE la mémoire donnée et la mémoire programme sont séparées.

RAM (Variables)

La mémoire RAM est utilisée pour stocker temporairement des données dans les variables durant l'exécution du programme. Toutes les données variables sont perdues lors de la coupure d'alimentation. Il y a quatre types de variables- usage générale, stockage, scratchpad et fonction spéciale.

Les variables sont des emplacements mémoires au sein du microcontrôleur PICAXE qui stockent les données pendant que le programme tourne. Toutes ces infos sont perdues quand le microcontrôleur est réinitialisé.

Pour info au sujet des variables mathématiques, voir l'information de la commande *'let'* dans le manuel 2.

Variables d'Usage Général

Il y a 14 ou plus d'octets de variables d'usage général. Ces octets de variables sont libellés b0, b1 etc. Les octets de variables peuvent stocker des nombres entiers entre 0 et 255. Les octets de variables ne peuvent pas utiliser des nombres négatifs ou fractionnels et seront en dépassement sans avertissement si vous excédez les bornes de valeurs 0 et 255 (ex : 254+3=1; 2-3=255).

Toutefois pour des grands chiffres, deux octets peuvent être combinés pour former un mot variable, qui est capable de stocker des entiers entre 0 et 65535. Ces mots variables sont libellés w0, w1, etc., et sont construits comme suit :

w0 = b1 :b0
w1 = b3;b2
w2 =b5 ;b4
w3 =b7 ;b6
w4 =b9 ;b8
w5 =b11;b10
w6 =b13 ;b12 etc.

Toutefois l'octet le plus significatif de w0 est b1, et le dernier octet significatif est b0.

En plus les octets b0 et b1 (w0) sont décomposés en bits individuels de variables. Ces bits variables peuvent être utilisés ou vous avez juste besoin d'une capacité de stockage d'un bit (0 ou 1).

b0 = bit7 : bit6 : bit5 : bit4; bit3; bit2 : bit1 : bit0
b1 = bit15 : bit14 : bit13 : bit12 : bit11 : bit10 : bit9; bit8

Les références M2, X1et X2 acceptent aussi les bits16- bit31 (b2-b3).

Vous pouvez utiliser n'importe quel mot, octet ou bit de variable au sein de toute session mathématique ou de commandes qui supportent les variables. Cependant prenez soin de ne pas répéter accidentellement l'usage du même octet ou bit de variable qui est déjà utilisé comme référence d'un 'mot' variable quelque part.

Toutes les variables d'usage général sont réinitialisées à 0 lors de la réinitialisation du programme.

Stockage des Variables

Les variables de stockage sont des emplacements de mémoire supplémentaires alloués pour le stockage temporaire d'octet de données. Ils ne peuvent pas être utilisés dans des calculs mathématiques mais peuvent être utilisés pour stocker temporairement des valeurs d'octets par l'usage des commandes *peek* et *poke*.

Le nombre d'emplacements de stockage disponible dépend du type de PICAXE. La table suivante donne le nombre d'octet de variables disponibles avec leurs adresses. Ces adresses varient suivant les spécifications techniques du microcontrôleur. Voir la description des commandes *peek* et *poke* pour plus d'infos.

PICAXE-18M2	228	28 à 255 ($1C à $FF
PICAXE-14M2/20M2	484	28 à 511 ($1C à $1FF)
Références M	48	80 à 127 ($50 à 7F)
Références A	48	80 à 127 ($50 à 7F)
Références X	96	80 à 127 ($50 à 7F), 192 à 239 ($C0 à $EF)
PICAXE-20X2	72	56 à 127 ($38 à 7F)
PICAXE-28X1	95	80 à 126 ($50 to $7E), 192 à 239 ($C0 à $EF)
PICAXE-28X2	200	56 à 255 ($38 à FF)
PICAXE-40X	112	80 à 127 ($50 à 7F), 192 à 255 ($C0 à $FF)
PICAXE-40X1	95	80 à 126 ($50 to $7E), 192 à 239 ($C0 à $EF
PICAXE-40X2	200	56 à 255 ($38 à FF)
PICAXE-08	aucun	

Scatchpad

PICAXE-20X2	128	0 à 127 ($00 à 7F)
PICAXE-28X1	128	0 à 127 ($00 à 7F)
PICAXE-28X2	1024	0 à 1023 ($00 à 3FF)
PICAXE-40X1	128	0 à 127 ($00 à 7F)
PICAXE-40X2	1024	0 à 1023 ($00 à 3FF).

Variables Fonctions Spéciales

Les variables fonctions spéciales disponibles pour utilisation dépendent du type de PICAXE

PICAXE-08 / 08M / 08M2 SFR

pins = le port entrée/sortie
dirs = la direction du registre de données (définit où les pins sont en entrées ou en sorties)
infra = autre terme pour la variable b13, utilisée au sein de la commande infrain2

la variable pins se décompose en bits variables individuels pour lecture des entrées individuelles avec une commande if...then. Seules les pins d'entrées valides sont implantées.

pins = x : x : x : pin4 : pin3 : pin2 : pin1 : x

La variable dirs est aussi décomposée en deux bits individuels. Seul les bits à configuration de bit bi-directionnel valide sont implantés.

dirs = x : x : x : dir4 : dir3 : dir2 : dir1 : x

PICAXE -14M2/ 18M2/20M2 SFR

pinsB	- pins entrée portB
outpinsb	- pins sortie portB
dirB	-direction registre de donnée portB
pinsC	- pins entrée portC
OutpinsC	- pins sortie portC
dirC	-direction registre de donnée portC
bptr	- pointeur de la RAM
@bptr	- valeur de l'octet RAM pointé par bptr
@bptrinc	- valeur de l'octet RAM pointé par bptr (après incrémentation)
@bptrdec	- valeur de l'octet RAM pointé par bptr (après décrémentation)
time	-l'heure actuelle
task	- la tâche actuelle

Quand utilisé à gauche d'une affectation 'pins' s'applique à la pins 'output' ex : let outpinsB = % 11000000 qui bascule les sorties 7,6 à 1 et les autres à 0.

Quand utilisé à droite d'une affectation 'pins' s'applique à la pins 'input 'ex let b1 = pinsB qui charge b1 avec l'état actuel de la pin d'entrée sur le portB.

La variable pinsX est décomposée en bits de variables individuels pour lire les entrées individuelles avec une commande if...then. Seules les pins 'input' sont implémentées, ex :

pinsB = pinB.7 : pinB.6 : pinB.5 : pinB.4 :
pinB.3 : pinB.2 : pinB.1 : pinB.0

La variable outpinX est décomposée en bit de variables individuels pour écrire les sorties directement. Seules les pins output valides sont implémentées, ex :

outpinsB = outpinB.7 : outpinB.6 : outpinB.5 : outpinB.4 :
outpinB.3 : outpinB.2 : outpinB.1 : outpinB.0

La variable dirsX est décomposée en bit de variables individuels pour régler directement les entrées /sorties ex :

dirsB = dirB.7 : dirB.6 : dirB.5 : dirB.4 :
dirB.3 : dirB.2 : dirB.1 : dirB.0

Voir la section ' Variable Générale' dans le manuel 2 pour plus d'information au sujet de @bptr, @bptrinc, @bptrdec

PICAXE-14M/20M SFR (Non les références M2)

pins	= le port entré quand lecture depuis le port
(out) pins	= le port de sortie quand écriture sur le port
infra	= autre terme pour la variable b13, utilisée avec la commande infrain2
keyvalue	= autre nom pour infra, utilisée avec la commande keyin.

Notez que pins est une 'pseudo' variable qui peut s'appliquer à la fois au port d'entrée et de sortie.

Quand utilisé à gauche d'une affectation 'pins' s'applique à la pins 'output' ex : let pins =% 11000000 qui bascule les sorties 7,6 à 1 et les autres à 0.

Quand utilisé à droite d'une affectation 'pins' s'applique à la pins 'input 'ex : let b1 =pins qui charge b1 avec l'état actuel du port d'entrée
En complément, noter que : let pins =pins signifie laisser le port sortie égal au port entré

Pour éviter cette confusion, il est recommandé que le nom 'outpins' soit utilisé dans ce type de déclaration ex : let outpins =pins

la variable pins se décompose en bits variables individuels pour lecture des entrées individuelles avec une commande if...then. Seules les pins d'entrées valides sont implantées.

pins = x : x : x : pin4 : pin3 : pin2 : pin1 : pin0 (14M)
pins = pin7 : pin6 : pin5 : pin4 : pin3 : pin2 : pin1 : pin0 (20M)

La variable outpins est décomposée en bit de variables individuels pour écrire les sorties directement. Seules les pins output valide sont implémentées, ex :

outpins = x : x : outpin5 : outpin4 :
outpin3 : outpin2 : outpin1 : outpin 0 (14M)

outpins = outpin7 : outpin6 : outpin5 : outpin4 :
outpin3 : outpin2 : outpin1 : outpin0 (20M)

PICAXE-18 / 18A / 18M / 18X SFR (non 18M2)

pins	= le port entré quand lecture depuis le port
(out) pins	= le port de sortie quand écriture sur le port
infra	= autre terme pour la variable b13, utilisée avec la commande infrain2
keyvalue	= autre nom pour infra, utilisée avec la commande keyin.

Notez que pins est une 'pseudo' variable qui peut s'appliquer à la fois au port d'entrée et de sortie.

Quand utilisé à gauche d'une affectation 'pins' s'applique à la pins 'output' ex : let pins =% 11000000 qui bascule les sorties 7,6 à 1 et les autres à 0.

Quand utilisé à droite d'une affectation 'pins' s'applique à la pins 'input 'ex : let b1 =pins qui charge b1 avec l'état actuel du port d'entrée
En complément, noter que : let pins =pins signifie laisser le port sortie égal au port entré

Pour éviter cette confusion, il est recommandé que le nom 'outpins' soit utilisé dans ce type de déclaration ex : let outpins = pins

la variable pins se décompose en bits variables individuels pour lecture des entrées individuelles avec une commande if...then. Seules les pins d'entrées valides sont implantées.
pins = pin7 : pin6 : pin5 : pin4 : pin3 : pin2 : pin1 : pin0

La variable outpins est décomposée en bits de variables individuels pour écrire les sorties directement. Seules les pins output valides sont implémentées

PICAXE-28 / 28A / 28X SFR

pins	= le port entré quand lecture depuis le port
(out) pins	= le port de sortie quand écriture sur le port
infra	= autre terme pour la variable b13, utilisée avec la commande infrain2
keyvalue	= autre nom pour infra, utilisée avec la commande keyin.

Notez que pins est une 'pseudo' variable qui peut appliquer à la fois au port d'entrée et de sortie.

Quand utilisé à gauche d'une affectation 'pins' s'applique à la pins 'output' ex : let pins =% 11000000 qui bascule les sorties 7,6 à 1 et les autres à 0.

Quand utilisé à droite d'une affectation 'pins' s'applique à la pins 'input 'ex : let b1 =pins qui charge b1 avec l'état actuel du port d'entrée

En complément, noter que : let pins =pins signifie laisser le port sortie égal au port entré

Pour éviter cette confusion, il est recommandé que le nom 'outpins' soit utilisé dans ce type de déclaration ex : let outpins =pins

la variable pins se décompose en bits variables individuels pour lecture des entrées individuelles avec une commande if...then. Seules les pins d'entrées valides sont implantées.

pins = pin7 : pin6 : pin5 : pin4 : pin3 : pin2 : pin1 : pin0

La variable outpins est décomposée en bits de variables individuels pour écrire les sorties directement. Seules les pins output valides sont implémentées

La variable outpin est décomposée en bits de variables individuels pour écrire les sorties directement. Seules les pins output valide sont implémentées, ex :

outpins = outpin7 : outpin6 : outpin5 : outpin4 :
outpin3 : outpin2 : outpin1 : outpin 0

PICAXE-28X1 / 40X1 SFR

pins	= pins entrée port
outpins	= pins sortie port
ptr	= pointeur scratchpad (bloc note)
@ptr	= pointeur scratchpad pointé par bptr
@bptrinc	= valeur de l'octet RAM pointé par bptr (après incrémentation)
@bptrdec	= valeur de l'octet RAM pointé par bptr (après décrémentation)
flags	= système de drapeau

Quand utilisé à gauche d'une affectation 'pins' s'applique à la pins 'output' ex : let pins =% 11000000 qui bascule les sorties 7,6 à 1 et les autres à 0.

Quand utilisé à droite d'une affectation 'pins' s'applique à la pins 'input 'ex : let b1 =pins qui charge b1 avec l'état actuel du port d'entrée
la variable pins se décompose en bits variables individuels pour lecture des entrées individuelles avec une commande if...then. Seules les pins d'entrées valides sont implantées.

pins = pin7 : pin6 : pin5 : pin4 : pin3 : pin2 : pin1 : pin0

La variable outpin est décomposée en bit de variables individuels pour écrire les sorties directement. Seules les pins output valides sont implémentées

La variable outpins est décomposée en bits de variables individuels pour écrire les sorties directement. Seules les pins output valide sont implémentées, ex :

outpins = outpin7 : outpin6, outpin5 : outpin4 :
outpin3 : outpin2 : outpin1 : outpin0

La variable scratchpad pointer est décomposée en bits de variables individuels

ptr = ptr7 : ptr6 : ptr5 : ptr4 : ptr 3 : ptr 2 : ptr1 : ptr0

Voir la section 'Variable-Scratchpad' pour de plus amples informations sur @ptr, @ptrinc, @ptrdec

L'octet système 'flag' est décomposé en bits de variables individuels. Si la fonction matérielle spéciale du drapeau n'est pas utilisée dans un programme, le drapeau individuel peut être utilisé librement comme un bit drapeau utilisateur défini.

Nom	Spécial	Fonction spéciale
flag0	-	réservé pour un usage futur
flag1	-	réservé pour un usage futur
flag2	-	réservé pour un usage futur
flag3	-	réservé pour un usage futur
flag4	-	réservé pour un usage futur
flag5	hserflag	hserial background receive has occurred
flag6	hi2cflag	hi2c écriture a eu lieu
flag7	toflag	drapeau de débordement timer

PICAXE-20X2 / 28X2 / 40X2 SFR

pinsA - pins entrée PortA
dirsA - donnée direction registre portA
pinsB - pins entrée PortB
dirsB - donnée direction registre portB
pinsC - pins entrée PortC
dirsC - donnée direction registre portC
pinsD - pins entrée PortD
dirsD - donnée direction registre portD
bptr - le pointeur de RAM
@bptr - valeur de l'octet RAM pointé par bptr
@bptrinc - valeur de l'octet RAM pointé par bptr (après incrémentation)
@bptrdec - valeur de l'octet RAM pointé par bptr (après décrémentation)
ptr - pointeur scratchpad (ptrh : ptrl)
@ptr - valeur de scratchpad pointé par ptr
@ptrinc - valeur de scratchpad pointé par ptr (après incrémentation)
@ptrdec - valeur de scratchpad pointé par ptr (après décrémentation)
flags -drapeau système

Quand utilisé à gauche d'une affectation 'pins' s'applique à la pins 'output' ex : let pins =% 11000000 qui bascule les sorties 7,6 à 1 et les autres à 0.

Quand utilisé à droite d'une affectation 'pins' s'applique à la pins 'input 'ex : let b1 =pins qui charge b1 avec l'état actuel du port d'entrée

la variable pinsX se décompose en bits variables individuels pour lecture des entrées individuelles avec une commande *if...then*. Seules les pins d'entrées valides sont implémantées.

pinsB = pin7 : pin6 : pin5 : pin4 :
pin3 : pin2 : pin1 : pin0 :

La variable outpinX est décomposée en bits de variables individuels pour écrire les sorties directement. Seules les pins output valide sont implémentées, ex :

outpins = outpin7 : outpin6 : outpin5 : outpin4 :
outpin3 : outpin2 : outpin1 : outpin0

La variable dirsX est décomposée en bits de variables individuels pour régler directement les entrées /sorties ex :

dirsB = dirB.7 : dirB.6 : dirB.5 : dirB.4 :
dirB.3 : dirB.2 : dirB.1 : dirB.0

voir la section 'Variables- Générale' dans le manuel 2 pour de plus amples information sur @bptr, @bprtinc, @bptrdec

Le pointeur scratchpad est décomposé en bit de variables individuels

ptrl = ptrl7 : ptrl6 : ptrl5 :ptrl4 : ptrl 3 : ptrl 2 : ptrl1 : ptrl0
ptrh = xxxx : xxxx : xxxx :xxxx : xxxx : xxxx : ptr9 : ptr8

voir la section 'Variables-Scratchpad' dans le manuel 2 pour de plus amples information sur @ptr, @prtinc, @ptrdec

L'octet système 'flag' est décomposé en bit de variables individuels. Si la fonction matériels spéciale du drapeau n'est pas utilisée dans un programme le drapeau individuel peut être utilisé librement comme un bit drapeau utilisateur défini.

Nom	Spécial	Fonction spéciale
flag0	hint0flag	interruption matérielle sur la pin INT0
flag1	hint1flag	interruption matérielle sur la pin INT1
flag2	hint2flag	interruption matérielle sur la pin INT2
flag3	hintflag	interruption matérielle sur n'importe quelle pin 0,1,2
flag4	compflag	interruption matérielle sur comparaison
flag5	hserflag	hserial background receive has occurred
flag6	hi2cflag	hi2c écriture a eu lieu
flag7	toflag	drapeau de débordement timer

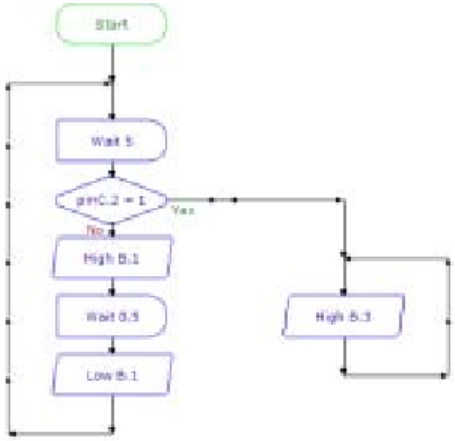
Traitement de Tâche Parallèle

Le circuit PICAXE éducatif M2 supporte l**e** traitement de tâche parallèle.

Un microcontrôleur PICAXE est un contrôleur monocœur et peut donc seulement traiter une instruction à un moment donné. La seule exception à cette règle est quand le circuit contient aussi un circuit périphérique séparé qui peut s'exécuter indépendamment du cœur principal. L'exemple principal d'un périphérique séparé est le périphérique pwm, qui est activé par la commande pwmout. Celui -ci génère des impulsions complètement séparées de la tâche du processeur principal, et ainsi ne nécessite aucun temps de traitement du cœur principal pour continuer à travailler en arrière-plan.

Cependant la nouvelle gamme de microcontrôleur PICAXE éducatif (réf M2) peut maintenant simuler un 'Traitement parallèle' par la commutation à plusieurs reprises entre un certain nombre de tâches à une vitesse très rapide. Ceci est rendu possible par les vitesses d'exploitation accrues des références les plus récentes- pour l'instant exécution de 4 tâches à 16Mhz approximativement, équivalent à une tâche exécutée à la vitesse de 4Mhz. Toutes les tâches donc 'semblent' être exécutées en parallèle. Les tâches en parallèle sont destinées à une utilisation éducative pour simplifier la programmation par de jeunes étudiants. La meilleure preuve par l'exemple.

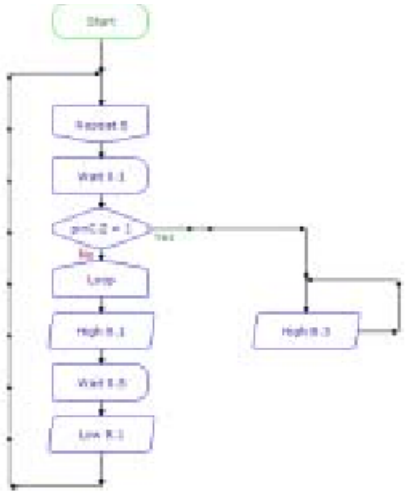
```
start0:
      pause 5000
      if pinC.2 = 1 then alarm
      high B.1
      pause 500
      low B.1
      goto start0
alarm:
      high B.3
      goto alarm
```



Un étudiant veut construire une alarme de vélo. Toutes les 5 secondes, une LED flash brièvement pour indiquer que l'alarme est active. Toutefois, lorsque la chaîne (fils) est coupée, un buzzer doit immédiatement sonner.
La première tentative de programmation de l'élève est montrée dans l'organigramme 1 ci-dessus. Ceci est la solution la plus évidente, mais ne fonctionne pas comme prévu. Ceci parce que l'alarme ne sonne pas immédiatement lorsque le fil est coupé - l'entrée est vérifiée uniquement une fois tous les cinq secondes, il peut y avoir jusqu'à un 'décalage' de 5 secondes avant que l'alarme sonne.

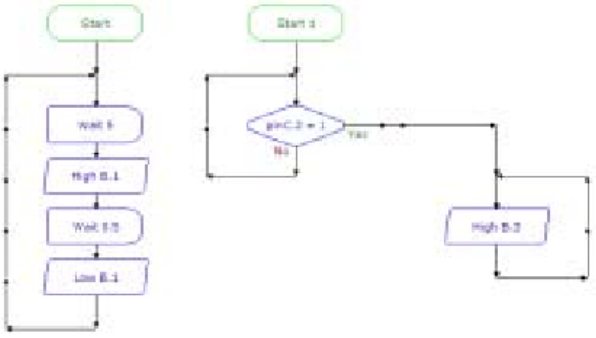
La bonne solution mono tâche est de briser le grand délai de 5 secondes en plusieurs 'morceaux' plus petits, par exemple 50 boucles de 0,1 s (100 ms), comme indiqué dans le schéma 2 ci-après.
Par conséquent, l'entrée est contrôlée beaucoup plus fréquemment et l'alarme sonne presque instantanément. Cependant cette solution n'est pas si facile à comprendre et la plupart des étudiants ne sauront pas atteindre cette solution sans aide de l'enseignant.

```
start0:
      for b4 = 1 to 50
         pause 100
         if pinC.2 = 1 then alarm
      next b4
      high B.1
      pause 500
      low B.1
      goto start0
alarm:
      high B.3
      goto alarm
```



L'organigramme 3 ci-dessous montre un moyen plus simple pour atteindre le résultat correct à l'aide des tâches parallèles. La tâche 1 fait clignoter tout simplement la LED, la tâche 2 vérifie le commutateur. Dans cette solution, l'entrée est détectée encore plus vite que la solution de travail unique ci-dessus. Cette solution de tâches en parallèle est aussi généralement plus facile à comprendre pour les étudiants que l'organigramme 2.

```
start0:
      pause 5000
      high B.1
      pause 500
      low B.1
      goto start0

start1:
      if pinC.2 = 1 then alarm
      goto start1
alarm:
      high B.3
      goto alarm
```



Comment Opèrent les Tâches Parallèles

Toutes les références M2 peuvent fonctionner en mode de tâche unique ou mode tâche parallèle. En mode monotâche la partie M2 fonctionne comme une partie traditionnelle de PICAXE et suit la séquence de programme (ligne par ligne) comme prévu.
Par l'inclusion d'étiquettes supplémentaires 'start' au sein de l'organigramme utilisateur / programme BASIC, le compilateur bascule automatiquement le programme téléchargé en mode tâche parallèle à la place. En mode de tâches en parallèle, il y a deux (ou plusieurs) positions de départ du programme et la puce PICAXE commence toutes les tâches après une réinitialisation.
Les commandes sont traitées d'une manière circulaire, par exemple avec deux tâches, la première commande dans la tâche 0 est traitée, puis la première commande dans la tâche 1 est traitée, puis la seconde commande dans la tâche 0 est traitée et ainsi de suite. Donc

Le noyau de traitement 'cycles' entre les différentes tâches. Au cours de temps 'mort' du traitement de la tâche (par exemple lors des retards tels qu'une commande pause) le noyau réalise automatiquement qu'il n'y a pas de traitement à effectuer dans cette tâche et se déplace immédiatement sur la prochaine tâche. Par conséquent, la réponse d'une autre tâche n'est pas affectée par un délai de pause.

Toutes les variables / RAM utilisateur / EEPROM sont partagées entre toutes les tâches. Par conséquent, si nécessaire, les tâches peuvent interagir et influencer le comportement de l'autre par le transfert de données dans des octets particuliers.

Étiquettes en mode Multitâche

Chaque tâche peut avoir n'importe quelle longueur. La seule restriction est que toutes les tâches doivent tenir dans la totalité de la zone mémoire de programme de cette puce M2.

Le début (′top′) du programme est toujours la tâche 0. Cette tâche est la première tâche qui est démarrée lorsque la puce est réinitialisée. Si vous le souhaitez une étiquette en option ' START0 *:* ' peut être utilisée, mais cela est également implicite par défaut si non utilisé. Si utilisé, ' START0 *:* 'doit toujours être sur la première ligne de code de programme.

Dans les programmes de base, la deuxième tâche, la tâche 1, est indiqué par l'utilisation de l' étiquette 'start1 :". De même la tâche 2 est indiqué par 'start2 :' et ainsi de suite. Dans les organigrammes Logicator une nouvelle cellule 'start' est simplement glissée dans l'organigramme pour indiquer les nouvelles positions de départ.

Le compilateur reconnaît automatiquement les étiquettes supplémentaires 'start' et passe donc le circuit en mode multi-tâches sur le nouveau téléchargement de programme.
Chaque tâche a son propre compteur de programme et de pile. Par conséquent, des sous-procédures peuvent être partagées entre différentes tâches si nécessaire, mais cela n'est généralement pas recommandé. Il y a une interruption (commande *setin*) qui interrompt toutes les tâches actuelles.

Suspensions des Tâches

Il est possible de désactiver les tâches au cours du programme par l'utilisation de la commande '*suspend*'. Une tâche suspendue peut ensuite être reprise par une commande '*resume*' dans une tâche différente.

La tâche en cours de traitement est stockée dans une variable utilisateur spéciale appelée variable 'task'. La variable 'task' est mise à jour au moment où le cœur commute sur une nouvelle tâche. Par conséquent, la commande '*suspend task*' suspendra la tâche en cours. Pour avoir une tâche particulière suspendue lors de la réinitialisation veillez simplement à ce que '*suspend task'* soit la première commande au sein de cette tâche.

Prenez soin de ne pas suspendre toutes les tâches en même temps, ou aucun traitement n'aura lieu! Une tâche particulière peut également être redémarrée en utilisant la commande '*restart*'. Notez que '*restart*" ne réinitialise pas la puce entière (utilisez la commande '*reset*' pour ce faire), de sorte que les variables etc. ne sont pas effacées par la commande '*restart*".

Les commandes '*sleep*' et '*nap*' coupent le noyau et placent la puce en mode faible puissance. Par conséquent, une commande sleep ou nap au sein de toutes les tâches suspendront toutes les tâches.

Simulation BASIC

Lors de la simulation du programme BASIC le logiciel organigramme 'PICAXE Editor 6' exécute toutes les tâches en parallèle comme prévu, mais normalement 'trace' (surlignés sur l'écran) le code BASIC pour une tâche à la fois pour éviter toute confusion. La tâche 0 est tracée par défaut, cependant une tâche différente peut être retracée en incluant une directive «#simtask X 'dans le programme. Ceci dirige le logiciel pour retracer la tâche X à la place. '#simtask all» est également acceptée et va rapidement mettre en évidence la ligne en cours de traitement dans chaque tâche, ce qui peut être déroutant à regarder.

Limitations

Le traitement en parallèle est principalement conçu pour des projets éducatifs simples. Il fonctionne très bien en utilisant des commandes d'entrée/sortie simples et des programmes qui contiennent des commandes de pause dans les tâches. Cela couvre la grande majorité des projets éducatifs à l'école.

Le traitement en parallèle n'est pas conçu pour des tâches parallèles complexes où chaque tâche va essayer d'utiliser différentes fonctionnalités avancées simultanément par exemple en essayant d'utiliser des protocoles de communication série/infrarouge/1 fil simultanément dans 3 différentes tâches! Dans cette situation, l'utilisateur final doit utiliser un programme de base unique pour traiter chaque fonctionnalité avancée à son tour

Les commandes qui nécessitent une utilisation totale du cœur pour maintenir l'intégrité de synchronisations critique (par exemple *readtemp*, *sertxd*, *debug*, *serin, irin* etc.) 'bloquera' l'attribution des tâches en parallèle jusqu'à ce que la commande soit finie/expirée. Par conséquent, les autres tâches apparaîtront 'bloquée' à un moment lors du traitement de cette commande.

En raison de la tâche cyclique, le temps entre chaque commande dans une tâche particulière ne peut pas être garanti, car les différentes longueurs des commandes dans les autres tâches seront traitées dans l'intervalle. Cela signifie également que la précision des commandes de pause sera légèrement diminuée. Si un programme exige spécifiquement une grande précision de synchronisation une seule tâche devrait être utilisée à la place.

La commande '*setfreq*' n'est pas disponible dans les programmes de tâches parallèles, car le noyau basculera automatiquement la fréquence de façon à maintenir une grande vitesse de traitement de la tâche en parallèle. Cependant la plupart des commandes apparaîtront comme fonctionnant à 4 MHz, de sorte que des commandes telles que *pulsout*/*pulsin, serout*/*serin, count* etc. doivent être calibrées comme pour un fonctionnement à 4 MHz. Les pulsations d'asservissement en toile de fond continueront à fonctionner, mais peuvent avoir une précision altérée avec les 'transitoires' occasionnels. Le changement dans la fréquence de fond peut également affecter les générations d'impulsions de fond pwm, il est donc recommandé que le mode de tâche unique soit utilisé pour les programmes contenant des commandes pwmout.

Le traitement des tâches multiples est beaucoup plus complexe qu'une seule tâche et ainsi en mode de tâches en parallèle le noyau nécessite l'utilisation de mémoire RAM supplémentaire dédiée. Par conséquent, sur le 18M2 en mode tâche parallèle seulement, les octets 128 à 255 de la RAM utilisateur sont réservés au titre de la RAM supplémentaire pour une utilisation par le noyau. 0-127 octets sont encore disponibles pour l'utilisateur final via des commandes *peek*/*poke*. En mode de tâches en parallèle le pointeur d'octets (de bptr) sur la 18M2 sera de retour à 0 après 127 (il déborde à 255 en mode monotâche). Cela ne vaut pas pour d'autres puces M2 (par exemple 14M2, 18M2 +, 20M2), le silicium de ces puces développées plus tard a été conçu pour inclure plus de RAM à cet effet.

Organigramme (Flowchart) ou BASIC?

Les logiciels PICAXE Editor 6 et AXEpad supporte la programmation en BASIC textuel.

Le logiciel PICAXE Editor 6 fournit une méthode graphique de programmation par organigrammes. Logicator, largement utilisé dans l'enseignement, est maintenant intégré dans PICAXE Editor 6. Ainsi PICAXE Editor remplace maintenant 'Logicator', une seule application fournit les deux méthodes de programmation,

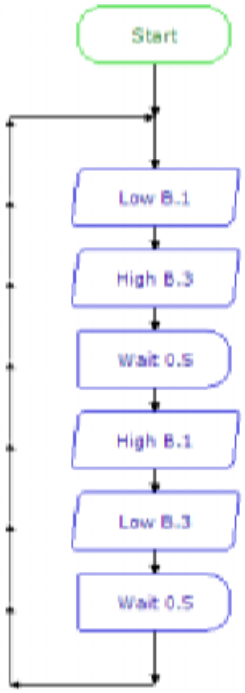
Toutes les méthodes de programmation utilisent les mêmes commandes BASIC et la même syntaxe. La méthode d'organigramme fournit simplement un moyen graphique de rejoindre les commandes BASIC ensemble, pour éviter de taper des programmes. Les organigrammes utilisent un sous-ensemble de commandes de base plus petit, et est généralement utilisé par des élèves plus jeunes dans le milieu scolaire.

Un avantage de la programmation par organigramme est la simulation graphique à l'écran. Cela permet aux élèves de 'voir' leur programme en fonctionnement avant de télécharger le PICAXE.

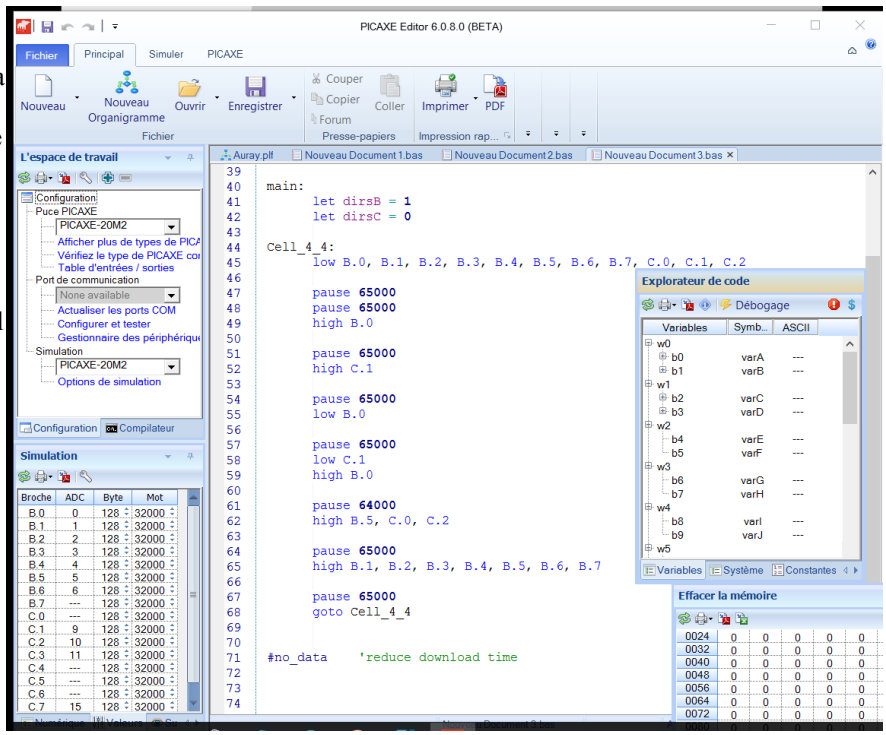
La plupart des amateurs et utilisateurs expérimentés dans l'éducation préfèrent la méthode de programmation BASIC textuel. Il est beaucoup plus puissant que des organigrammes, qui peuvent devenir très compliqués pour les grands programmes.

Tous les organigrammes sont automatiquement convertis en programmes BASIC avant de télécharger le microcontrôleur PICAXE. Par conséquent, le principal objectif de ce manuel est la programmation en BASIC textuel.

```
main:
      low B.1
      high B.3
      pause 500
      high B.1
      low B.3
      pause 500
      goto main
```

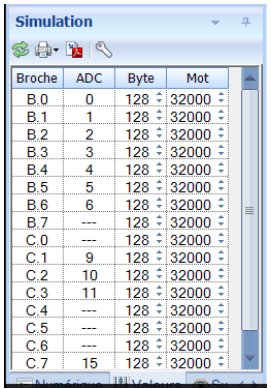


Simulation BASIC



Quand un programme BASIC a été saisi, la simulation est lancée en cliquant sur le menu Simuler-> Exécuter (ou en appuyant sur <Ctrl> + <F5>)
Le panneau principal de simulation est toujours affiché lors d'une simulation, mais varié en apparence pour correspondre au mode de puce PICAXE utilisée (Espace de travail→Configuration → Puce PIXCAXE).
Différents schémas d'animation de simulation peuvent être sélectionnés

Les **Entrées/Sorties** sont colorés respectivement en jaune et vert. Quand ils sont d'une couleur sombre, ils sont off (= 0), quand ils ont leur propre couleur, ils sont on (= 1).



Pour modifier une condition d'entrée il suffit de cliquer sur la zone colorée sur le dessin de simulation jaune vif indique un niveau logique 1 (on).

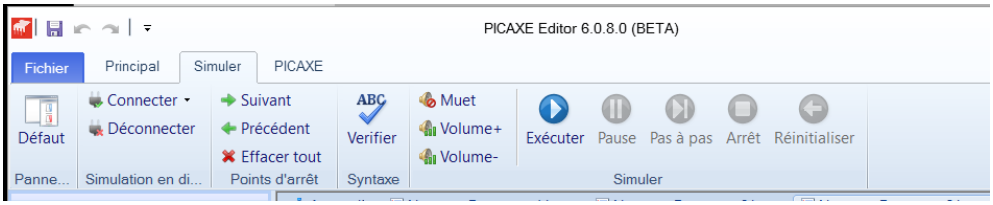
Les valeurs d'entrée analogiques sont présentées dans une grille sur le deuxième onglet. et peut

être modifié par le bouton de défilement haut/bas boutonsvers le haut ou en tapant sur la valeur directement (0255). Byte sont des valeurs utilisées par la commande 'readadc' et d'autres commandes qui utilisent une valeur d'octet.

La valeur de 'mot' fonctionne d'une manière similaire (0-65535) et est utilisé par les commandes suivantes en tant que valeur d'entrée: count, pulsin, readadc10, readtemp, readtemp12 etc.

Contrôle de Flux du Programme et les Points d'Arrêt

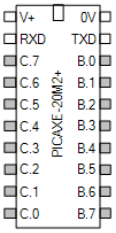
Trois boutons sur le panneau principal de simulation sont des boutons de raccourci pour les fonctions du menu Simulation.



Les points d'arrêt peuvent être placés ou (retirés) dans le programme en cliquant simplement sur le numéro de ligne dans la marge. Alternativement, le menu Simulation-> Basculer point d'arrêt peut être utilisé pour insérer retirer un point d'arrêt à la position actuelle du curseur. Les points d'arrêt sont indiqués par une barre rouge dans la marge.

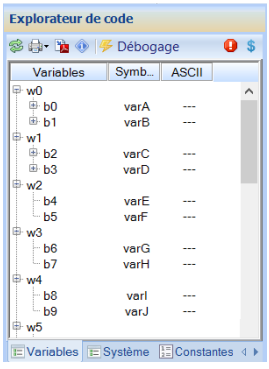
Pour un seule étape dans un programme placer un point d'arrêt sur la première ligne que vous voulez étudier, puis cliquez sur Exécuter. De ce point en utilisant } vous avancerez à travers le programme.

L'affichage des autres panneaux disponibles (lors du démarrage de la simulation) est déterminé par Simuler-> Options Panneaux de Simulation.



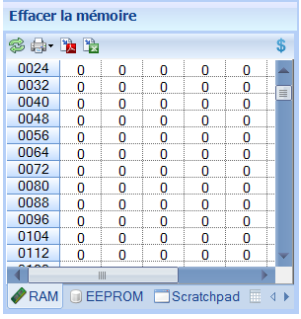
Pour modifier une valeur d'entrée, cliquez sur le 'Switch' à côté de la disposition des pins. Pour changer une valeur analogique utilisez le curseur pour modifier la valeur analogique.

La valeur 'générique' est utilisée pour entrer des données pour des commandes telles que count, pulsin etc.



Panneau Variables
Le panneau de variables montre les valeurs de l'octet variable actuel (b0, b1, etc.) ou un mot (w0, w1, etc.). Pour modifier une valeur variable double-cliquez sur la variable tandis que le programme est en pause. Vous pouvez alors entrer une nouvelle valeur.

Panneau de Mémoire
Le panneau de la mémoire affiche les valeurs actuelles de l'EEPROM de données ou SFR ou zones de mémoire scratchpad. Pour les puces PICAXE qui stockent également le programme utilisateur dans l'EEPROM de données, les emplacements de mémoire actuellement utilisés par le programme sont indiqués par un 'P'.



Panneau de Sortie de Série
Le panneau de sortie de série affiche la sortie avec les commandes *serout* et *sertxd*.
Les commandes Debug ne sont pas simulées car les valeurs des variables sont toujours disponibles dans le Panneau 'variables'.

Délai d'Exécution de la Simulation
Le curseur (en bas à droite de l'écran, dans la barre d'état principal) définit le délai d'exécution entre chaque ligne de simulation

Options de Simulation

Utilisez Configuration -->Simulation pour sélectionner les options variées de simulation.

Délai de Simulation
Ce curseur fixe le délai entre chaque ligne du programme simulé.

Étiquettes en Surbrillance
Cette option met en évidence et retarde les étiquettes qui sont elles-mêmes sur une ligne. Cela ralentit la simulation, mais permet à l'utilisateur de voir clairement quelle étiquette a été sautée..

Simulation LCD
La commande *serout* sur la sortie sélectionnée affichera une simulation d'écran LCD. Cette simulation correspond à la commande série standard AXE033 ou FRM010
(les caractères personnalisés, l'horloge AXE033 et les fonctions d'alarme AXE033 ne sont pas simulés).

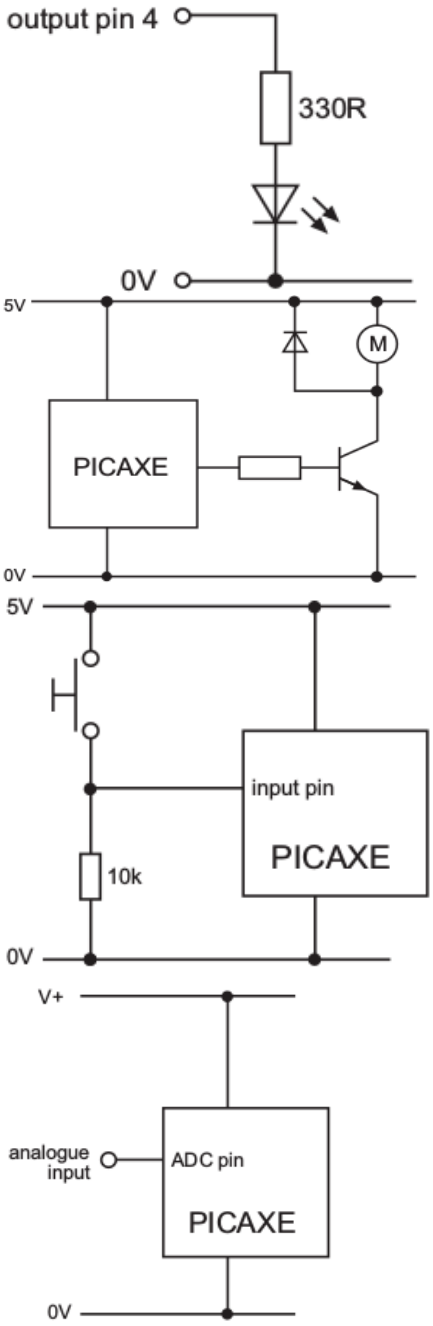
Simulation EEPROM
Ajoute une EEPROM 24LC16B ou 24LC256 simulée pour les commandes d'i2c.

Simulation DS1307 RTC
Ajoute une horloge DS1307 simulée en temps réel pour les commandes d'i2c. Le registre heure et date lue utilise les valeurs de l'horloge interne de l'ordinateur. Les écritures pour changer ces registres de date/heure sont ignorées dans la simulation.

Récapitulatif Circuit d'Interfaçage

Cette section fournit un très bref aperçu de l'interfaçage des entrées/sorties des microcontrôleurs PICAXE.

Pour plus d'explications et de détails, voir le manuel 3.
Le manuel 3 sur les Circuits d'interfaçage fournit des diagrammes détaillés de connexion et des exemples de programmes pour la plupart des entrées/sorties des capteurs communs.



Sorties Digitales

Le microcontrôleur peut absorber ou fournir 20mA sur chaque pin de sortie, 90mA maximum par puce.
Donc les dispositifs courants faibles tels que les LED peuvent s'interfacer directement avec les pins de sortie.
Les dispositifs à courant plus élevés peuvent être reliés par l'intermédiaire d'un transistor, ou FET, ou d'un réseau de pilote Darlington,

Entrée Digitale

Les commutateurs d'entrées numériques peuvent être interfacés avec une résistance de 10kΩ pour tirer vers le bas.
La résistance est essentielle car elle empêche l'entrée d'être 'flottante' lorsque l'interrupteur est en position ouvert.
Ceci donnerait un mauvais fonctionnement.
Notez que la résistance de 10kΩ est pré-équipée sur les entrées de la carte projet.

Entrées Analogiques

Les entrées analogiques peuvent être raccordées à un diviseur de potentiel entre V + et 0V.
La référence analogique est la tension d'alimentation, et le signal analogique ne doit pas dépasser la tension d'alimentation.

Tutoriel 1 - Comprendre et utiliser le Système PICAXE

La puce PICAXE, le 'cerveau' du système PICAXE, lorsque fournie neuve sans un programme de contrôle, ne fait rien! L'utilisateur doit écrire un programme de contrôle sur l'ordinateur, puis télécharger ce programme dans la puce PICAXE.

Par conséquent, le système PICAXE se compose de trois éléments principaux :

Le logiciel 'PICAXE Editor 6 '

Ce logiciel s'exécute sur un ordinateur et vous permet d'utiliser le clavier de celui-ci pour saisir des programmes dans un langage BASIC simple. Les programmes peuvent aussi être générés par des dessins organigrammes. Trois applications de logiciels tiers peuvent être utilisés en alternance (par exemple 'Flowol"ou les logiciels (Yenka' peuventt être utilisé pour simuler des circuits électroniques complets PICAXE, programmés par l'intermédiaire des organigrammes).

Le câble de Téléchargement USB AXE027

C'est le câble qui raccorde l'ordinateur au système PICAXE. Le câble doit être seulement raccordé pour télécharger les programmes. Il ne doit pas être connecté lors de l'exécution du programme PICAXE car celui-ci est stocké en permanence dans la puce PICAXE- même quand l'alimentation est déconnectée

La puce PICAXE et le circuit

La puce microcontrôleur PICAXE 'exécute' le programme qui a été téléchargé. Cependant la puce doit être montée sur une carte électronique qui permet une connexion de la puce au microcontrôleur.

La carte électronique peut être conçue par l'utilisateur sur un morceau de circuit imprimé, une interface préfabriquée ou un circuit d'apprentissage qui peut être utilisé pour la rapidité et la facilité de mise en œuvre.

Résumé - Procédure de programmation

1) Écrire le programme sur l'ordinateur en utilisant le logiciel de PICAXE Editor 6
2) Branchez le câble de téléchargement à partir de l'ordinateur vers le PICAXE.
3) Connectez l'alimentation à la carte PICAXE.
4) Utilisez le logiciel PICAXE Editor 6 pour télécharger le programme. Le câble de téléchargement peut alors être retiré après le téléchargement.

Le programme démarrera sur le PICAXE automatiquement. Cependant, le programme peut également être relancé à tout moment en appuyant sur le bouton de réinitialisation (si disponible) ou par la coupure de l'alimentation.

Convention d'Appellation des Pins Entrée/Sortie

Les premières puces PICAXE ont un maximum de 8 pins en entrée et de 8 en sortie, donc il n'y avait pas besoin d'un schéma de dénomination du port, car il y avait un seul port d'entrée par défaut et un port de sortie par défaut pour chaque puce.

Par conséquent les pins d'entrée et de sorties ont été simplement appelées par leur numéro de pin. Par exemple :

Les commandes de sortie	Commandes d'entrée
High 1	count 2, 100, w1
rsound 2, (50,50)	pulsin 1, 1, w1
serout 3, N2400, (b1)	serin 0, N2400, b3

Toutefois, plus tard sur les références, PICAXE M2 et X2 plus de flexibilité a été ajoutée en permettant à la quasi-totalité des pins d'être configurées comme entrées ou sorties comme vous le souhaitez.
Cela crée plus de 8 entrées ou sorties et un schéma de dénomination modifiée est donc nécessaire. Par conséquent, les pins sur ces références sont désignées par la nouvelle notation de PORT.PIN. Jusqu'à 4 ports (A, B, C, D) sont disponibles, en fonction du nombre de pins de la puce Par exemple :

Les commandes de sortie	Commandes d'entrée
High 1	count A.2, 100, w1
sound C.2, (50,50)	pulsin B.1, 1, w1
serout A.3, N2400, (b1)	serin C.0, N2400, b3

Dans le cas des expressions *if ... then* qui vérifient l'état de la variable de la pin d'entrée, la convention de nommage de ces variables de pins d'entrée ont changé dans un style proche de

si pin1 = 1 alors …

à

si pinC.1 = 1 alors ...

Le nom de l'octet des pins d'entrée de chaque port est changé de

pins

à

pinsA, pinsB, pinsC, pinsD

Le nom de l'octet des pins de sortie de chaque port est changé de

outpins

à

outpinsA, outpinsB, outpins !

Le nom de la direction de registre de données pour chaque port est changé de

dirs

à

dirsA, dirsB, dirsC, dirsD

Ce manuel utilise généralement le format de PORT.PIN plus récent dans les exemples, sauf un exemple qui est spécifiquement pour une référence plus ancienne.
S'il vous plaît voir les schémas de brochage pour la puce que vous utilisez. On notera que les numéros de pins entrée/sortie utilisés dans les commandes ne sont pas les mêmes que la patte physique.

Téléchargement d'un programme BASIC

Le programme suivant bascule la sortie 4 on et off toutes les secondes. Quand vous téléchargez ce programme, la LED devra flasher au rythme d'une seconde

```
main  :
        high b.4
        pause 500
        low b.4
        pause 500
        goto main
```

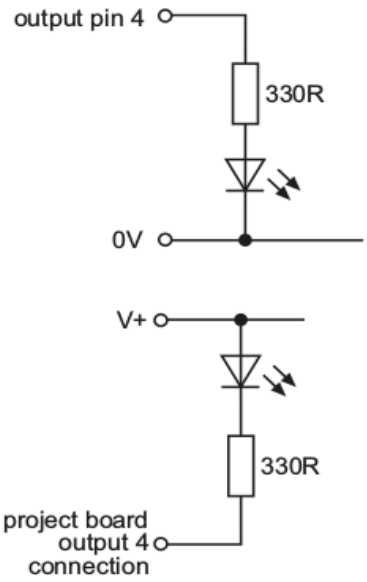
Ce programme utilise les commandes **high** et **low** pour contrôler la sortie 4; et utilise la commande **pause** pour introduire un retard de (1000ms = 1 seconde)

La dernière commande **goto main** fait revenir le programme à l'étiquette **main** : au départ du programme. Ceci signifie que le programme boucle en permanence. Noter que la première fois le label est suivi d symbole (:) Ceci dit à l'ordinateur que ce mot est un nouveau label.

Instructions détaillées :

1) Connecter le câble PICAXE au port série/USB de l'ordinateur. Noter quel port COM est connecté.
2) Démarrer le logiciel 'PICAXE Editor 6 '
3) Sélectionner espace de travail-->Configuration
4) Sélectionner la puce PICAXE appropriée.
5) Cliquer sur l'onglet 'Port' et sélectionner le port série auquel le câble PICAXE est raccordé. Cliquer 'OK'
6) Entrer le programme suivant :

```
main  :
        high b.4
        pause 500
        low b.4
        pause 500
```

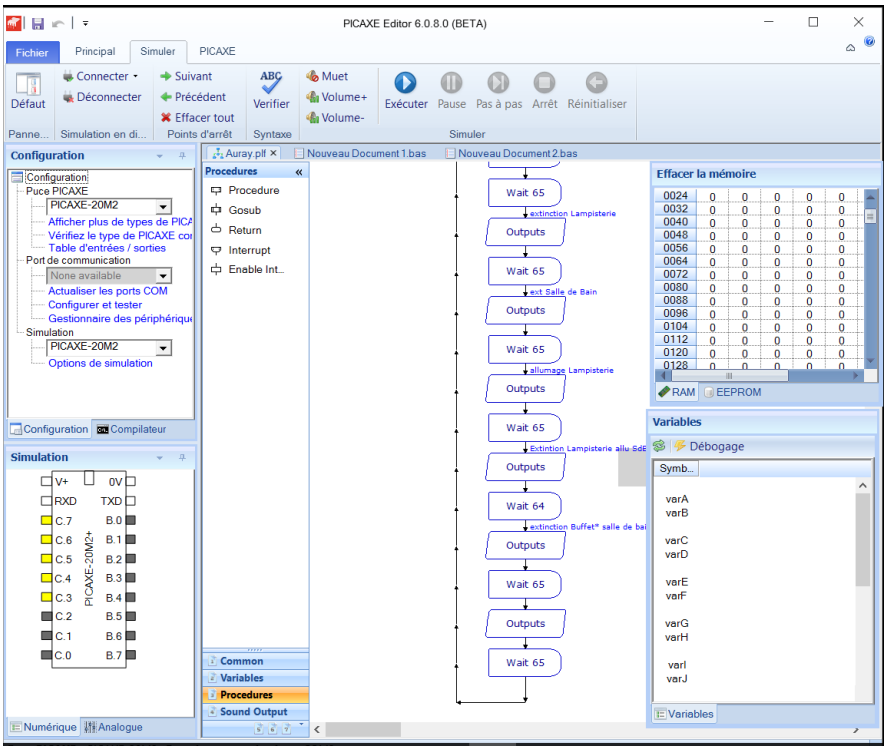


```
goto main
```

NB le symbole (:) directement après 'main' et les espaces entre les commandes et les numéros.

7) Connecter une LED (et une résistance de 330Ω) à la pin de Sortie 4. Si la LED est connectée directement à une puce PICAXE sur un **proto** (ou un circuit fait par vous-mêmes) connecter la LED directement **entre la pin de sortie et le 0V**. Quand vous utilisez le circuit **projet** (comme ceux fournis avec les packs de démarrage 14, 18, et 28) connecter la LED **entre le V+ et la sortie** du connecteur, car la sortie est amplifiée par un circuit darlington sur la carte projet (assurez-vous que la LED est connectée de la bonne façon!).
8) Soyez sûr que le circuit PICAXE est connecté au câble série et que les batteries soient connectées.
9) Sélectionner PICAXE-->Exécuter. Une barre de téléchargement devrait apparaître pendant que le programme se charge. Quand le téléchargement est terminé, le programme s'exécute automatiquement, la LED doit flashée toutes les secondes.

Simuler un Programme BASIC



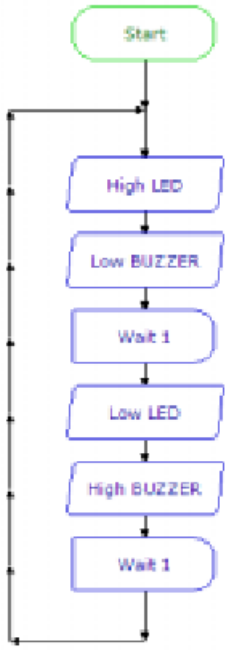
Pour simuler le programme cliquer simplement sur le bouton Simuler--> Exécuter.

Chaque ligne du code BASIC sera surlignée lors de son exécution et un graphique sur écran montre l'état de toutes les pins d'entrées et de sorties.

Pour changer l'état d'une entrée, cliquer simplement sur le bouton d'entrée qui est à côté de la patte correspondante sur le graphique.

Tutorial 2 – Utilisation de Symboles, Commentaire et Espace-vide

Quelquefois il est difficile de se rappeler quelle pin est raccordée à quel appareil. La commande 'symbol' peut alors être utilisée au début du programme pour renommer les entrées et les sorties




```
symbol LED   = B.4          ; rename output4 'LED'
symbol buzzer = B.2          ; rename output2 'buzzer'

main:                        ; make a label called 'mai
      high LED               ; LED on
      low buzzer             ; buzzer off
      pause 1000             ; wait 1 second (1000 ms)
      low LED                ; LED off
      high buzzer            ; buzzer on
      wait 1                 ; wait 1 second
      goto main              ; jump back to the start
```

Rappelez-vous que les **commentaires** (une explication après le symbole ;) peut être sur chaque ligne du programme une façon plus aisée de comprendre. Ces commentaires sont ignorés par l'ordinateur quand il télécharge le programme dans le PICAXE.

Une étiquette (ex : **main :** dans le programme ci-dessus) peut être n'importe quel mot (en dehors de mot comme 'switch') mais doit commencer par une lettre. Quand l'étiquette est la première à être définie, elle doit être suivie de symbole (:). Celui-ci indique à l'ordinateur que le mot est une nouvelle étiquette.

L'ordinateur utilise la commande **wait.** Les commandes **wait** et **pause** créent des délais. Cependant **wait** peut seulement être utilisée avec des secondes entières et **pause** avec des délais plus courts (mesurée en millisecondes, ex 25 /1000).

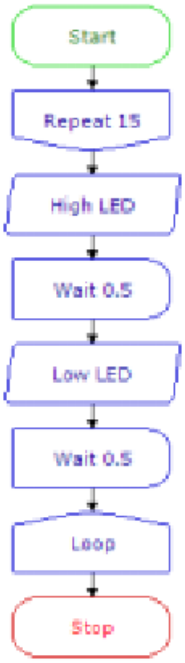
Wait peut être suivi par un nombre comprit ente 1 et 65.
Pause peut être suivit par un nombre entre 1 et 65535.

Il est aussi une bonne technique de programmation d'utiliser des tabulations (ou des espaces vides) au début des lignes sans étiquettes de sorte que toutes les commandes soient parfaitement alignées. Le terme '**white-space' est**t utilisé par les programmeurs pour définir des tabulations, des espaces et des lignes vides, et l'usage correct des espaces vides peut rendre le listing du programme plus facile à lire et comprendre. Voir l'exemple sur ma page suivante, où le code mes commandes *for...next* est aussi indenté avec une tabulation pour plus de clarté.

Note :
Certaines langages BASIC parmi les premiers utilisaient les **numéros de lignes** plutôt que les **étiquettes** pour la commande '*goto*'. Malheureusement, ce système de numéro de ligne peut ne pas être pratique à utiliser, parce que si vous modifiez votre programme en ajoutant plus tard, ou en retirant, des lignes de code vous avez alors par conséquent à modifier tous les numéros de ligne dans les commandes *goto.* Le système de l'étiquette, tel qu'il est utilisé dans la plupart des langages BASIC modernes, surmonte ce problème automatiquement.

Tutoriel 3 – Boucles For...Next

C'est souvent utilisé pour répéter la même part de programme un nombre de fois, par exemple pour faire clignoter une LED. Dans ces cas, une boucle **for...next** peut être utilisée.



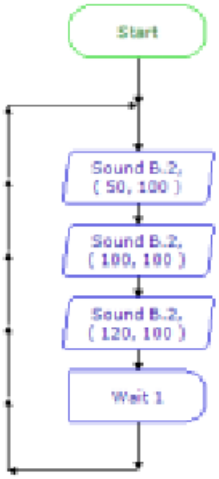
Ce programme fait clignoter la LED connectée à la sortie 1, 15 fois. Le nombre de fois que chaque code est répété est stocké dans la RAM d'usage général du PICAXE en utilisant la variable b1 (le PICAXE contient 14 octets de variable libellés b0 à b13) Ces variables peuvent aussi être renommées en utilisant la commande symbol pour rendre celles-ci plus faciles à mémoriser.

```
symbol counter = b1     ; define the variable b1 as "counter"
symbol LED = B.4        ; define pin 4 with the name "LED"

main:
      for counter = 1 to 15         ; start a for...next loop
            high LED                ; switch pin 4 high
            pause 500               ; wait for 0.5 second
            low LED                 ; switch pin 4 low
            pause 500               ; wait for 0.5 second
      next counter                  ; end of for...next loop

      end                           ' end program
```

Noter encore comment l'espace vide a été utilisé pour montrer clairement toutes les commandes contenues entre les commandes **for** et **next**.

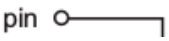
Tutoriel 4 – Faire des Sons

Les buzzer fabriquent des sons à fréquence fixe quand ils sont activés. Cependant le système PICAXE peut automatiquement créer des bruits de différentes fréquences par l'utilisation de commande de son, jeu, mélodie avec un élément acoustique Piezo.
Toutes les puces PICAXES acceptent la commande son, qui est destinée à faire des bips d'avertissement, etc. Ceci est recommandé au lieu d'utiliser les buzzers.

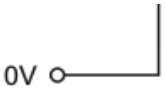


Pour jouer de la musique, la commande son n'est pas recommandée. Quelques puces PICAXE acceptent les commandes jouer, syntoniser, spécialement désignées pour jouer de la musique. Se référer à la commande '*Tune*' dans le manuel 2 pour plus de détails.

Exemple de programme son :




```
main:
      sound B.2,(50,100)      ; freq 50, length 100
      sound B.2,(100,100)     ; freq 100, length 100
      sound B.2,(120,100)     ; freq 120, length 100
      pause 1000              ; wait 1 second
      goto main               ; loop back to start
```



Pour tester ce programme vous devez ajouter un piézo entre la pin de sortie (dans ce cas sortie 2) et 0V. Noter que la platine projet (fournie avec les packs de démarrage PICAXE-14, 18, 28) est muni d'un driver Darlington, le piezo doit être relié directement à la pin de sortie PICAXE (pas la sortie du driver).

Le premier nombre indique le numéro de la pins (dans ce cas, la sortie 2). Le prochain nombre (entre parenthèses) est le ton, suivi de la durée. (noter que les seuls tons valides vont de 0 à 127).
Le programme suivant utilise une boucle *for ... next* pour produire 120 sons différents.
Le nombre stocké dans la variable b0 augmente de 1 à chaque boucle (1-2-3 etc.) Par conséquent, en utilisant le nom de la variable b0 dans la position de tonalité, la note peut être changé sur chaque boucle.

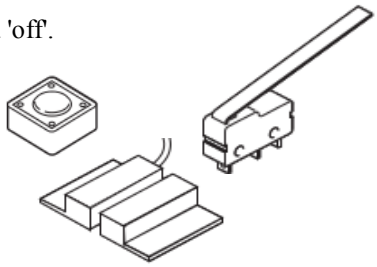
```
main:
      for b0 = 1 to 120       ; start a for...next loop
         sound B.2,(b0,50)    ; make a sound, freq from b0
      next b0                 ; next loop
      end
```

Le programme suivant fait la même tâche, mais à reculons, en utilisant la valeur d'incrément de - 1 (plutôt que la valeur par défaut de 1 ci-dessus).

```
main:
      for b0 = 120 to 1 step -1     ; count down in loop
         sound B.2,(b0,50)          ; make a sound. freq from b0
      next b0                       ; next loop
      end
```

Tutoriel 5 – Utilisation des Entrées Digitales

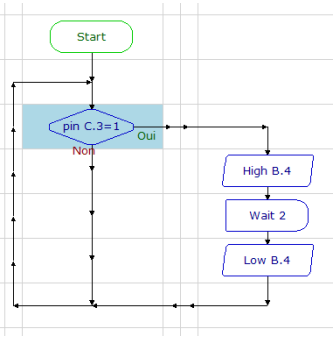
Un capteur digital est un simple capteur du type contact qui peut seulement être 'on' ou 'off'.



Des exemples communs de capteurs digitaux sont :

- microswitches
- poussoirs et interrupteurs à bascule
- contacts reed

Ce programme ci-dessous montre comment réagir avec des poussoirs. Dans ce programme la sortie 4 flashe à chaque fois que le poussoir sur l'entrée 3 est actionné. Noter que si vous utilisez une puce 18 pattes, vous devrez sélectionner une pin différente (ex : pin0, car la pin 3 n'existe pas sur ce modèle)




```
main:                           ; make a label called 'main'
      if pinC.3 = 1 then flash  ; jump if the input is on
      goto main                 ; else loop back around

flash:                          ; make a label called 'flash'
      high B.4                  ; switch output 4 on
      pause 2000                ; wait 2 seconds
      low B.4                   ; switch output 4 off
      goto main                 ; jump back to start
```

Dans ce programme les trois premières lignes forme une boucle continue. Si l'entrée est off (=0) le programme boucle juste tour après tour. Si le switch est on (=1) le programme saute à l'étiquette appelée **'flash'.** Le programme alors fait flasher la sortie 4 pendant deux secondes avant de retourner à la boucle principale.

Notez soigneusement l'orthographe dans la ligne **if ... then**, **pin3** est un seul mot (sans espace). En effet, pin3 est le nom d'une variable qui contient les données de la pin d'entrée. Notez également que seule l'étiquette est placée après la commande **then**.

Deux commutateurs (ou plus) peuvent être combinés par les mots clés AND ou OR.

Une porte AND 2 entrées est programmée comme suit :

```
if pinC.2 = 1 and pinC.3 = 1 then flash
```

Une porte OR 3 entrées est programmée comme suit :

```
if pinC.1 = 1 or pinC.2 = 1 or pinC.3 = 1 then flash
```

Pour lire l'ensemble du port d'entrée utilisez la commande suivante

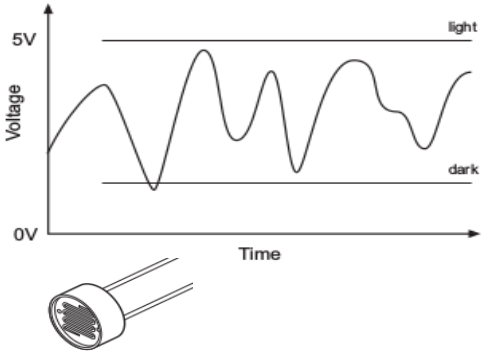
```
let b1 = pinsC
```

Pour isoler des pins individuelles (par exemple 6 et 7) dans le port, masquer les pins variables avec une déclaration de & (ET logique)

```
let b1 = pinsC & %11000000
```

Tutoriel 6 – Utilisation d'Entrées Analogiques

Un capteur analogique mesure un signal continu tel une lumière, une température ou une position.
Le capteur analogique fournit un signal de tension variable. Cette tension peut être représentée par un nombre dans la gamme de 0 à 255. (ex : sombre =0, lumière = 255).



Des exemples de capteurs analogiques courants sont :

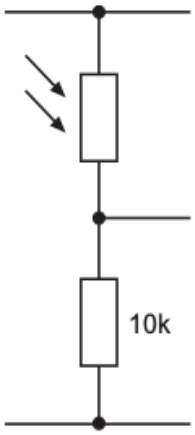
- LDR (résistance variant avec la lumière)
- Thermistance (résistance variant avec la température)
- Potentiomètres

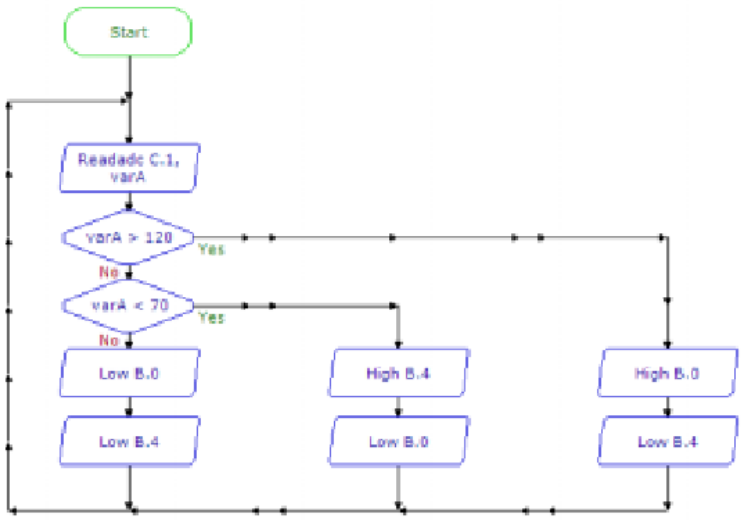
Utilisation d'une LDR

La LDR est un exemple de capteur analogique. Elle est connectée à l'entrée ADC par l'intermédiaire d'un diviseur de tension (ex : entrée 1). Noter que toute les entrées n'ont pas la propriété ADC – voir le diagramme de brochage pour de plus amples informations.

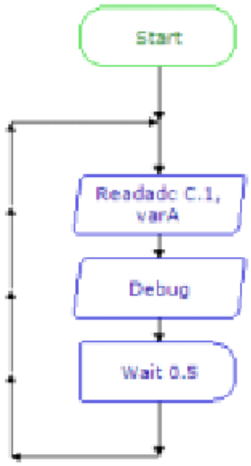
La valeur d'une entrée analogique peut facilement être copiée dans une variable par l'usage de la commande *'readadc'*. La valeur variable (0 à 255) peut alors être testée. Le programme suivant commute une LED si la valeur est plus grande que 120 et une LED différente si la valeur est inférieure à 70. Si la valeur est entre 70 et 120, les deux LED sont éteintes.

```
main:                           ; make a label called ,main
      readadc C.1,b0            ; read ADC1 into variable b0
      if b0 > 120 then top      ; if b0 > 120 then do top
      if b0 < 70 then bot       ; if b0 < 70 then do bot
      low B.0                   ; else switch off 0
      low B.4                   ; and switch off 4
      goto main                 ; jump back to the start

top:                            ; make a label
      high B.0                  ; switch on 0
      low B.4                   ; switch off 4
      goto main                 ; jump back to start

bot:                            ; make a label
      high B.4                  ; switch on 4
      low B.0                   ; switch off 0
      goto main                 ; jump back to start
```





Tutoriel 7 – Utilisation de Debug



Quand l'utilisation de capteurs analogiques est souvent nécessaire pour calculer les valeurs 'threshold' (de seuils) pour le programme (ex : les valeurs 70 et 120 dans le programme du tutoriel 6), la commande *debug* fournit un méthode aisée pour voir en 'temps réel' la valeur d'un capteur ainsi la valeur de seuil peut être calculée par expérimentation.

```
main:                   ; make a label called main
      readadc C.1,b0    ; read channel 1 into variable b0
      debug b0          ; transmit value to computer screen
      pause 500         ; short delay
      goto main         ; jump back to the start
```

Après que ce programme ait été exécuté une fenêtre 'debug' montrant la valeur de la variable b0 apparaîtra sur l'écran de l'ordinateur. Comme la lumière tombant sur le capteur de la LDR est modifiée, la valeur de la variable indiquera la lecture actuelle du capteur.

La fenêtre de débogage s'ouvre automatiquement après un téléchargement, mais peut aussi être ouverte manuellement à tout moment via le bouton Débogage dans l'Explorateur de Code

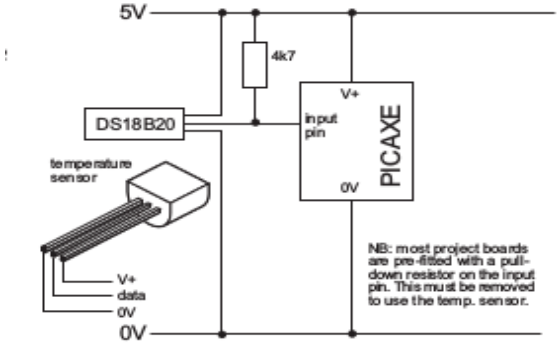
Tutoriel 8 – Utilisation d'un Terminal Série avec Sertxd

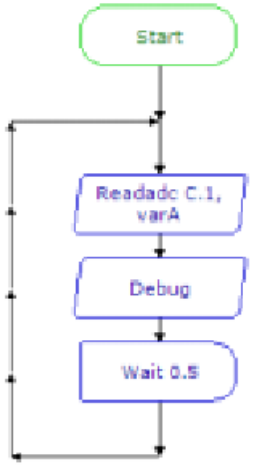
Toutes les variantes PICAXE acceptent la commande de débogage
Toutefois, les références M et X prennent également en charge les messages de débogage série plus complexes par l'utilisation de la commande *sertxd,* qui envoie une chaîne série définie par l'utilisateur à l'ordinateur (à la vitesse de transmission 4800). Cela peut être affiché par la fonction Terminal Série incluse (menu PICAXE--> Terminal).
Le terminal série peut également être ouvert automatiquement à chaque fois qu'un téléchargement se fait par l'onglet Principal--> Exécuter.

```
main:                   ; make a label called main
      readtemp C.1,b0   ; read input 1 into variable b0
      sertxd ("The value is ",#b0,cr,lf)
      pause 500         ; short delay
      goto main         ; jump back to the start
```

La commande *sertxd* transmet la chaîne "La valeur est" suivie par la chaîne de caractères ASCII de la valeur actuelle de la variable b1 (le préfixe # devant la variable indique une chaîne de caractères ASCII représentant la valeur correcte à transmettre). Les constantes CR et LF sont des valeurs prédéfinies (13 et 10) qui font que le terminal série affiche une valeur pour chaque nouvelle ligne afin que l'affichage se mette à jour correctement.

Ce programme utilise la commande *readtemp* pour lire la température d'un capteur de température numérique DS18B20 connecté à l'entrée 1.





Tutoriel 9 - Systèmes numériques

Un microcontrôleur opère en effectuant un grand nombre de commandes dans un très court laps de temps par traitement des signaux électroniques. Ces signaux sont codés dans le système binaire - le signal étant soit haut (1) soit bas (0).

Le système de comptage utilisé dans les activités de tous les jours est le système décimal. Ce système de numérotation utilise les dix chiffres familiers 0-9 pour définir la grandeur du nombre.

Cependant lorsque l'on travaille avec des microcontrôleurs il est parfois plus facile de travailler en binaire. Cela est particulièrement vrai lorsque vous essayez de contrôler plusieurs sorties en même temps.

Un chiffre binaire unique est renvoyé à un **bit** (chiffre binaire). Les systèmes PICAXE utilisent 8 bits (**1 octet= byte**), avec le bit le moins significatif (LSB), bit 0, sur le côté droit et le bit le plus significatif (MSB), le bit 7, sur le côté gauche.

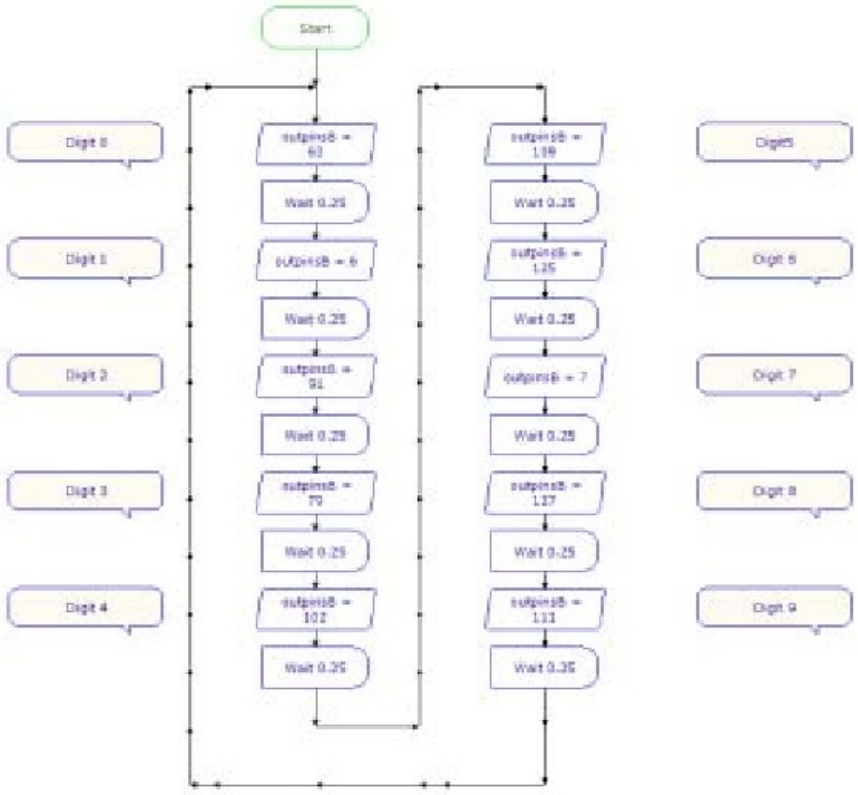
Par conséquent, le nombre binaire % 11001000 signifie mettre les bits 7,6,3 sont haut (1) et les autres bas (0). Le signe % indique à l'ordinateur que vous travaillez en binaire au lieu de décimale.

Cela signifie que l'ensemble des 8 sorties peuvent être commandées en même temps, au lieu de plusieurs commandes haut et bas.

```
main:
      let pinsB = %00111111   ; digit 0
      pause 250               ; wait 0.25 second
      let pinsB = %00000110   ; digit 1
      pause 250               ; wait 0.25 second
      let pinsB = %01011011   ; digit 2
      pause 250               ; wait 0.25 second
      let pinsB = %01001111   ; digit 3
      pause 250               ; wait 0.25 second
      let pinsB = %01100110   ; digit 4
      pause 250               ; wait 0.25 second
      let pinsB = %01101101   ; digit 5
      pause 250               ; wait 0.25 second
      let pinsB = %01111101   ; digit 6
      pause 250               ; wait 0.25 second
      let pinsB = %00000111   ; digit 7
      pause 250               ; wait 0.25 second
      let pinsB = %01111111   ; digit 8
      pause 250               ; wait 0.25 second
      let pinsB = %01101111   ; digit 9
      pause 250               ; wait 0.25 second
      goto main
```

Le programme suivant montre comment afficher sur un afficheur sept segments AXE050 un comptage 0-9.

Chaque lignes 'let pins= ' change le nombre de barres qui sont allumées sur l'afficheur sept segments sur la carte de tutoriel. Ceci est plus rapide et plus efficace pour la mémoire, que l'utilisation de plusieurs commandes 'haut' et 'bas'.



Tutoriel 10 – Sous procédures

Une sous procédure est un 'mini-programme' qui peut être appelé depuis le programme principal. Une fois que la sous-procédure a été effectuée le programme principal se poursuit.
Les sous-procédures sont souvent utilisées pour séparer le programme en petites sections plus facile à comprendre. Les sous-procédures qui terminent les tâches courantes peuvent également être copiées d'un programme pour gagner du temps.
Les microcontrôleurs PICAXE série X supportent 255 sous-procédures. Toutes les autres séries supportent 15 sous-procédures.

Le programme suivant utilise deux sous-procédures pour séparer les deux sections principales du programme ('flash' et 'noise').

```
symbol LED = B.4              ; rename output4 'LED'
symbol buzzer = B.2           ; rename output2 'buzzer'
symbol counter = b1           ; define a counter using variable b1

main:                         ; make a label called 'main'
      gosub flash             ; call the sub-procedure flash
      gosub noise             ; call the sub-procedure noise
      goto main               ; loop back

      end                     ; end  of the main program


flash:                        ; make a sub-procedure called flash
      for counter = 1 to 25   ; start a for…next loop
         high LED             ; LED on
         pause 50             ; wait 0.05 second
         low LED              ; LED off
         pause 50             ; wait 0.05 second
      next counter            ; next loop
      return                  ; return from the sub-procedure

noise:
      high buzzer             ; buzzer on
      pause 2000              ; wait 2 seconds
      low buzzer              ; buzzer off
      return                  ; return from the sub-procedure
```

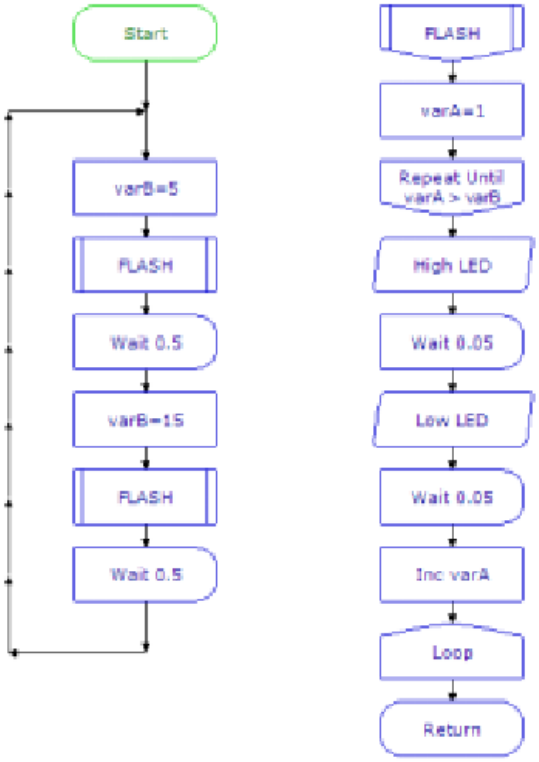
Ce deuxième programme montre comment une variable peut être utilisée pour transférer des informations dans une sous-procédure. Dans ce cas la variable b2 est utilisée pour indiquer au microcontrôleur de clignoter 5 fois, puis 15.

```
symbol LED = B.4             ; rename output4 'LED'
symbol counter = b1          ; define a counter using variable b1

main:                        ; make a label called 'main'
      let b2 = 5             ; preload b2 with 5
      gosub flash            ; call the sub-procedure flash
      pause 500              ; wait a while
      let b2 = 15            ; preload b2 with 15
      gosub flash            ; call the sub-procedure flash
      pause 500              ; wait a while
      goto main              ; loop back

      end                    ; end  of the main program


flash:                       ; make a sub-procedure called flash
      for counter = 1 to b2  ; start a for…next loop
         high LED            ; LED on
         pause 250           ; wait 0.25 second
         low LED             ; LED off
         pause 250           ; wait 0.25 second
      next counter           ; next loop
      return                 ; return from the sub-procedure
```

Tutoriel 11 – Utilisation des Interruptions

Une interruption est un cas particulier d'une sous-procédure. La sous-procédure se produit immédiatement après qu'une entrée particulière (ou combinaison d'entrées) est activée.

Une interruption interrogée est un moyen plus rapide de réagir à une combinaison d'entrée particulière. Il est le seul type d'interruption disponible dans le système PICAXE. Le port d'entrées est vérifié entre l'exécution de chaque ligne de commande dans le programme, entre chaque note d'une commande de mise au point, et de façon continue au cours d'une commande de pause. Si la condition particulière des entrées est vraie, une '*gosub* ' à la sous-procédure d'interruption est exécutée immédiatement. Lorsque la sous-procédure a été effectuée, l'exécution du programme se poursuit à partir du programme principal.



SETINT avec la valeur 0 comme l'octet de masque.

L'état des entrées d'interruption est tout motif de '0' et '1' sur le port d'entrée, masquée par l'octet ' mask'. Par conséquent tous les bits masqués par un '0' dans l'octet mask seront ignorés.

Exemples :

Pour interruption seulement sur entrée 1 niveau haut

setint %00000010,%0 0000010

Pour interruption seulement sur entrée 1 niveau has

setint %00000000,%0 0000010

Pour interruption entrée0 haute, entrée1 haute et entrée2 basse

setint %00000011,%0 0000111

etc.

Seule un masque d'entrée est autorisé à tout moment. Pour désactiver l'interruption, exécuter une commande

Notes :
1) Chaque programme qui utilise la commande *setint* doit avoir une interruption correspondante : sous-procédure (terminé par une commande de retour) au bas du programme.
2) Lorsque l'interruption se produit, l'interruption est définitivement désactivée. Par conséquent, pour réactiver l'alarme (si désiré) une commande SETINT doit être utilisée au sein de l'interruption : sous-procédure elle-même. L'interruption ne sera pas activée tant que la commande "retour" ne sera pas exécutée
3) Si l'alarme est réactivée et l'état d'interruption n'est pas effacé dans la sous-procédure, une deuxième interruption peut se produire immédiatement après la commande de retour.
4) Une fois le code d'interruption exécuté, l'exécution du programme se poursuit à la ligne suivante du programme dans le programme principal. Dans le cas de la pause interrompue, attendre, jouer ou la commande de mise au point, tout retard de temps qui reste est ignoré et le programme se poursuit avec la ligne suivante du programme.

Plus d'explications détaillées SETINT.
Le SETINT doit être suivi de deux nombres - un pour 'comparer avec la valeur '(entrée) et un pour le 'masque de saisie '(masque) dans cet ordre. Il est normal d'afficher ces numéros en format binaire, car les pins 'actives' sont plus en évidence. En format binaire l'input7 est sur la gauche et input0 est sur la droite.

Le second nombre, le «masque de saisie», définit les pins qui doivent être vérifiées pour voir si une interruption doit être générée ...
-% 00000001 vérifiera pin d'entrée 0
-% 00000010 vérifiera pin d'entrée 1
-% 01000000 vérifiera pin d'entrée 6
-% 10000000 vérifiera pin d'entrée 7
- etc
Ceux-ci peuvent également être combinés pour vérifier un certain nombre de pins d'entrée en même temps ...
-% 00000011 vérifiera pins d'entrée 1 et 0
-% 10000100 vérifiera pins d'entrée 7 et 2
Ayant décidé quelle pins vous souhaitez utiliser pour l'interruption, le premier nombre (valeur des entrées) indique si vous souhaitez que l'interruption se produise lorsque ces entrées particulières sont sur on (1) ou off (0).
Une fois qu'un SETINT est actif, le PICAXE surveille les pins que vous avez spécifiées dans "masque de saisie ', où '1' est présent, en ignorant les autres pins

Un masque de saisie de % 10000100 vérifiera les pins 7 et 2 et créera une valeur de % a0000b00 où le bit 'a' sera 1 si la pin 7 est élevé et 0 si faible, et le bit 'b' sera 1 si la pin 2 est élevé et 0 si faible.
Le 'comparer avec la valeur ', le premier argument de la commande SETINT, est que cette valeur créée est comparée, et si les deux correspondent, l'interruption se produit, si elles ne correspondent pas alors l'interruption ne se produira pas.
Si le «masque de saisie" est% 10000100, les pins 7 et 2, puis le résultat de 'comparer avec la valeur' peut être l'un des suivants ...

% 00000000 Pin 7 = 0 et la pin 2 = 0
% 00000100 Pin 7 = 0 et la pin 2 = 1
% 10000000 Pin 7= 1 et la pin 2 = 0
% 10000100 Pin 7= 1 et la pin 2 = 1

Donc, si vous voulez générer une interruption lorsque la pin 7 est haute et la pin 2 est basse, le 'masque de saisie' est% 10000100 et le 'comparer avec la valeur'est 10000000%, ce qui donne une commande de SETINT de ..

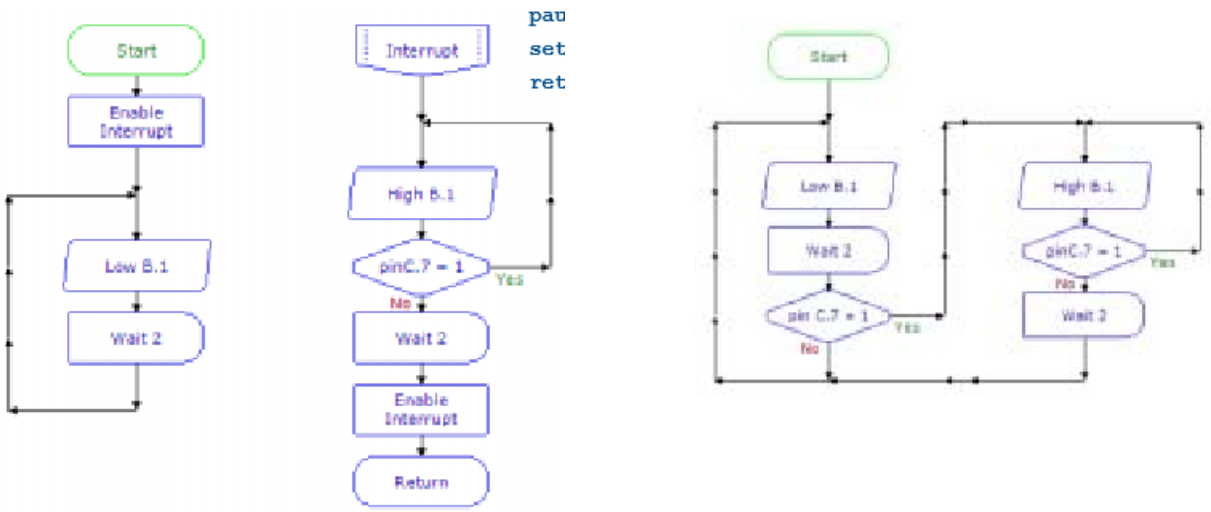
- SETINT% 10000000, 10000100%

L'interruption se produit alors quand, et seulement quand, la pin 7 est haute et la pin 2 est basse. Si la pin 7 est basse ou la pin 2 est haute, l'interruption ne se fera pas tant que deux pins sont 'examinées' dans le masque.

Exemple :

```
setint %10000000,%10000000
' activate interrupt when pinC.7 only goes high

main:
      low B.1                     ; switch output 1 off
      pause 2000                  ; wait 2 seconds
      goto main                   ; loop back to start

interrupt:
      high B.1                    ; switch output 1 on
      if pinC.7 = 1 then interrupt ; loop here until the
                                  ; interrupt cleared
      pause 2000                  ; wait 2 seconds
      setint %10000000,%10000000  ; re-activate interrupt
      return                      ; return from sub
```

Dans cet exemple, une LED sur la sortie 1 va s' allumer immédiatement quand l'entrée est commutée haute. Avec if pin 7 = 1 then .... Type déclaration, le programme pourrait prendre jusqu'à deux secondes pour allumer la LED comme si la déclaration n'est pas traités pendant le temps de pause de 2000 dans la boucle principale du programme (programme standard indiqué ci-dessous en comparaison).




```
main:
      low B.1                ; switch output 1 off
      pause 2000             ; wait 2 seconds
      if pinC.7 = 1 then sw_on
      goto main              ; loop back to start

sw_on:
      high B.1               ; switch output 1 on
      if pinC.7 = 1 then sw_on
                 ; loop here until the condition is cleared
      pause 2000             ; wait 2 seconds
      goto main              ; back to main loop
```

Prochaine étape – votre propre projet PICAXE

Vous devriez maintenant avoir une bonne idée sur la façon dont le système PICAXE fonctionne et devriez être en mesure de commencer à concevoir votre propre projet.

Assurez-vous que vous avez aussi étudié les manuel 2 (commandes BASIC) et 3 (circuits d'interfaçage de microcontrôleurs) pour plus d'informations.

Il existe un large éventail d'idées de projets et d'exemples dans les fichiers d'aide du logiciel de l'éditeur de programmation. L'étude de ces projets fournira d'autres idées, comme en regardant le forum très actif au sein de la section du support technique du site Web principal de PICAXE (www.picaxe.forumco.uk).

Le forum a une très grande communauté de passionnés de PICAXE qui peut toujours donner un coup de main si vous êtes aux prises avec un projet!

Il n'y a aucune limite à la créativité des utilisateurs PICAXE ! Aller à votre propre projet, vous pourriez être surpris de la rapidité avec laquelle vous pouvez commencer à développer des projets électroniques excitants basés sur des microcontrôleurs !

Annexe A – Commandes Sommaires BASIC

Cette annexe fournit un aperçu des commandes disponibles. Se référer au manuel 2 pour plus d'informations spécifiques et d'exemple pour chaque Commande BASIC.

Entrée	high, low, toggle, pulsout, letpins=,
Sortie	if...then, if portA...then, if portCthen..., pulsin, button
Comptage	count,
ADC	readadc, readadc10
Config portC	letdirsc=
sortie portc	high portc, low portc, let pinsc=
PWM	pwmout
RAM	peek, pook
Son	sound
Série	serin,serout
Déroule du progr	goto, gosub, return, branch
Boucles	for...next
Mathématiques	let (+, -, *, /, //, max, min, &, | , ^, &/, |/, ^/)
Variables	if ...then, random, lookdown, lookup
Mémoire Donnée	eeprop, write, read
Tempo	pause, wait, nap, sleep, end
Divers	symbol, debug
Interruption	setint
Commande Servo	servo
Infrarouge	infrain
Température	readtemp, readtemp12
1-wire Serial No	readowsn
Clavier	keyin, keyled
Scratchpad	put, get, @ptr, @ptrinc, @ptrdec
ADC	calibadc, calibadc10
Serie	hsersetup, hserout, hserin, serrxd
SPI	spiin, spiout, hspisetup, hspiin, hpsiout
I2C	hi2csetup, hi2cin, hi2cout
One -Wire	owin, owout
PWM	hmpwm
Timer	settimer
Comd Alimentation	hibernate, enablebod, disablebod

Annexe B – Surcadencement à haute fréquence

Toutes les principales fonctions de PICAXE sont basées sur une fréquence de résonateur de 4 MHz (8MHz réf X2). Cependant, l'utilisateur peut choisir de 'Surcadencer' certaines réf PICAXE pour obtenir un fonctionnement plus rapide

Pour changer la fréquence sur les références courantes

Toutes les puces 8, 14, 18 et 20 pins
Télécharger un programme contenant la commande la commande *setfreq m4* (pour 4Mhz) ou *setfreq m8* (pour 8Mhz). Si aucune commande *setfreq* n'est utilisée dans un programme, la fréquence sera par défaut 4Mhz (8Mhz pour les références X2), Noter que la nouvelle fréquence est effective immédiatement après l''exécution de la commande.

PICAXE -28X1/28X2 et PICAXE -40X1/ 40X2
Souder les résonateurs les résonateurs céramiques trois pattes appropriés sur le circuit. Utiliser la commande *setfreq* pour basculer entre la fréquence externe et interne,

Pour changer la fréquence sur les anciennes (obsolètes) références
Avec le -08,-18 le résonateur interne est fixé à 4Mhz et ne peut pas être changé,
Avec le -08M, -14M, -18A, -18X, le résonateur interne a une valeur par défaut de 4Mhz, Cependant il peut être augmenté à 8Mhz par l'utilisateur via l'utilisation de la commande *setfreq m8*.

Lors de téléchargement de nouveau programmes sur des références obsolètes, assurez-vous que la fréquence correcte (#freq directive) est utilisée pour correspondre au dernier programme tournant sur la puce PICAXE. Dans le doute faire un Réset matériel à 4Mhz.

Avec le -28X1 / -40X1, le résonateur interne a une valeur par défaut de 4 MHz.
Cependant, il peut être augmenté par l'utilisateur à 8 MHz via l'utilisation de la commande '*setfreq M8* '

Avec le -28 et le -28A un résonateur 4Mhz externe doit être utilisé,
Avec le-28X/ -40X un résonateur céramique trois pattes est normalement utilisé, mais il est aussi possible d'utiliser un résonateur plus rapide (8 ou 16Mhz), bien que cela aura une incidence sur le fonctionnement de certaines des commandes.

Avec les -28X1 /400X1 le résonateur interne a par défaut une valeur de 4Mhz, Cependant il peut être augmenté à 8Mhz par l'utilisateur via l'utilisation de la commande *setfreq m8* ou par un résonateur céramique trois pattes 16/20Mhz, via l'utilisation de la commande *setfreq em16 (em20)*.

Le logiciel PICAXE Editor n'accepte que les résonateur de fréquence 4, 8 ou 16 MHz .Aucune autre fréquence n'est recommandées, Si une autre fréquence est utilisée il peut ne pas être possible de télécharger un nouveau programme dans le microcontrôleur PICAXE.

PICAXE-28X et PICAXE-40X
Souder les résonateurs les résonateurs céramiques trois pattes appropriés sur le circuit projet

Téléchargement de Programmes à 4, 8, 16 MHz (vieilles références obsolètes seulement)
Après avoir modifié la fréquence, vous devez sélectionner la bonne fréquence . Si la mauvaise fréquence est sélectionnée, le programme ne sera pas téléchargé Ceci n'est pas requis pour les références M2, X1 et X2 car elles sont par défaut sur le résonateur interne pour le téléchargement.

Commandes affectées par la fréquence du résonateur
Plusieurs commandes sont affectées par un changement de fréquence du résonateur. Un sommaire de ces commandes affectées est donnée plus loin (voir le manuel 2 Commandes BASIC pour plus de détails sur la syntaxe et information)

Lors de l'utilisation des appareils avec un résonateur interne, rappelez-vous qu'il est parfois possible de revenir à 4 MHz pour exécuter la commande dépendant de cette vitesse. Exemples :

```
setfreq m4
readtemp 1, b1
setfreq M8
```

Cela est impossible avec les appareils dotés d'un résonateur externe. Ce processus est automatique sur les références M2, X1 et X2.

Commandes pour laquelle le fonctionnement est affecté par le changement de la vitesse du résonateur :

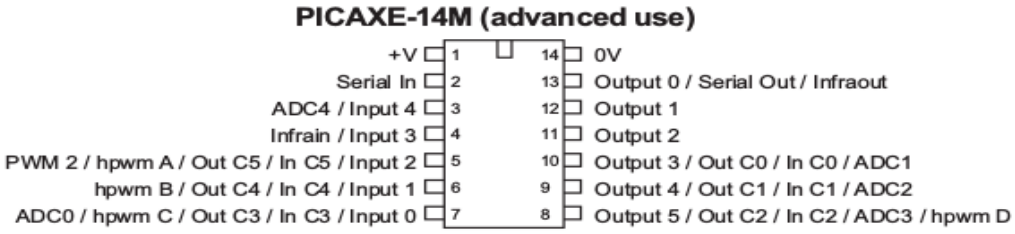
- count
- dbug
- readi2c, writei2c, i2cin, i2cout
- pause, wait
- pulsin, pulsout
- pwm. pwmout
- serin, serout, sertxd, sertxd, usersetup, serin, hserout
- sound

Notez que nap, doze et lsleep ne sont pas affectés par la vitesse du résonateur car ils utilisent leur propre horloge interne, séparée.

Les commandes suivantes ne fonctionneront pas à 8 ou 16 MHz en raison de problèmes de synchronisation avec le dispositif externe indiqué. Notez que les références M2, X1 et X2 passent automatiquement en mode 4MHz interne pour traiter ces commandes, de sorte que la fréquence externe peut être plus élevée.

- irin, infrain, infrain2, irout, infraout (récepteur infrarouge)
- kbin, keyin (clavier)
- kbled, keyled (clavier)
- readtemp / readtemp12 (capteur température DS18B20)
- readowsn, owin, owout (1-wire device)
- servo (servo)
- play, tune (music)
-

Annexe C -Configuration des Entrée-Sortie du PICAXE- 14M



Le PICAXE-14M est un appareil très polyvalent. Dans son état par défaut, qui est conçu principalement pour un usage éducatif, il a simplement les 'entrées à gauche', et les 'sorties à
droite '.
Cependant les utilisateurs plus avancés peuvent reconfigurer les 3 pins du bas de chaque côté pour être des entrées ou des sorties. Cela a des avantages supplémentaires comme suit :
- Quantité plus souple des entrées et sorties
- Plus de canaux CAN deviennentt disponibles
- La possibilité d'utiliser pwmout via les commandes pwmout et hpwm.

Le diagramme ci-dessus montre la fonction avancée de chaque pin. Les repères 6 sont disposés dans un 'port' (portC) avec les bits marqués C0-C5. On notera que le nombre de bits portC ne correspond pas aux numéros normaux d'entrée/sortie (ou même les numéros de patte!). Étudier le schéma de brochage très attentivement!

Utilisation des pins portc comme sorties

Toutes les pins du portc peuvent être configurées pour être utilisées comme une sortie numérique.

Pour convertir la pin C3 en sortie et la rendre haute
high portc 3
Pour convertir la pin C3 en sortie et la rendre faible
low portc 3

Pour convertir toutes les pins en sorties
let dirsc =% 00111111
Pour convertir toutes les pins sur les entrées
let dirsc =% 00000000

Il est impossible d'accéder aux pins C3-C5 du portc avec tout autre type commandes 'sortie'
(serout, pulsout etc). Par conséquent, lorsque utilisées comme sorties ces pins devrait être réservées comme des sorties aussi simple que on / off . Rappelez-vous que C0-C2 sont des
sorties (3-5) normales, de toute façon, et peuvent donc être utilisées avec une commande de sortie.

Utilisation de portc comme entrées digitales

Les pins C0, C1, C2 du portc sont par défaut, configurées en sorties. Elles peuvent cependant être reconfigurées comme entrées, mais vous devez vous assurer que votre conception matériel tient compte du fait que la pin sera une sortie à la mise sous tension. Une simple résistance 1k en série avec la pin résoudra normalement ce problème.

Pour faire de la pin une entrée, vous devez utiliser 'let dirsc= 'comme décrit ci-dessus.

La syntaxe suivante est utilisée pour tester la condition d'entrée : if portc pin0 = 1 then jump
A savoir, le mot-clé supplémentaire 'portc' est inséré après la commande *'if'*.

pour tester si deux entrées (ou plus) sur portc sont on
if portc pin0 = 1 AND pin1 = 1 then jump
pour tester si l'un des deux (ou plusieurs) entrées PORTC sont sur
if portc pin0 = 1 OR pin1 = 1 then jump
Notez la commande portc est nécessaire une seule fois après la commande *'if'*.
Il est impossible de tester les entrées sur deux ports différents dans le même *if ... then*.

Il est impossible d'accéder aux pins de portc avec d'autres commandes de type 'entrée' (count, pulsin etc). Par conséquent, ces pins doivent être réservées comme de simple interrupteurs marche / arrêt.

Notez que 'dirsc' utilise la notation commune BASIC 0 pour l'entrée et 1 pour la sortie.

Utilisation portc comme entrées analogiques
Trois pins ADC supplémentaires, ADC1,2,3, sont disponibles APRÈS que la pin correspondante ait Pinété convertie en une entrée. Vous devez vous assurer que votre conception matériel tient compte du fait que la pin sera une sortie à la mise sous tension. Une simple résistance 1k en série avec la pin résoudra normalement ce problème.

Utilisation portc comme sorties PWM
C5 peut être utilisé avec la commande *pwmout*, mais fera de cette pin une sortie. Pins C2-5 (hpwm A-D) peuvent tous être utilisées avec la commande *hpwm*, mais fera également des pins correspondantes des sorties.

Note spéciale - Pin sortie 0
Pin 0 (patte 13) est utilisée pendant le téléchargement du programme, mais peut également être utilisée comme une sortie normale une fois le téléchargement terminé. Par conséquent, vous devez vous rappeler que votre périphérique de sortie passera rapidement on et off si le téléchargement a lieu (pas un problème avec sorties simples comme les LED, mais pourrait causer des problèmes avec d'autres appareils tels que les moteurs)

Annexe D – Configurations des Entrées/Sorties PICAXE -08/08M/08M2

Le microcontrôleur PICAXE-08 dispose de 5 pins d'entrée / sortie. Contrairement à la plus grande partie des microcontrôleurs PICAXE (où les pins sont pré-définies), l'utilisateur peut choisir si certaines pins sont utilisées comme entrées ou comme sorties.

La pin 0 doit toujours être une sortie, et la pin 3 doit toujours être une entrée (ceci est dû à la construction interne du microcontrôleur). Les 3 autres pins peuvent être sélectionnées pour être entrées ou sorties, et ainsi l'utilisateur peut sélectionner une combinaison d'entrée/sortie dans les limites de 1 entrée-4 sorties et 4 entrées-1 sortie.

De plus la pin 1 contient également un convertisseur analogique-numérique faible résolution et ainsi peut être utilisé comme une pin d'entrée analogique si nécessaire.

Important - Ne vous trompez pas!
Les numéros de pins d'entrée / sortie NE sont PAS les mêmes que les numéros de la 'patte' externe, ainsi la numérotation des pins d'entrée/ sortie suit l'allocation des port des fabricants de microcontrôleurs. Pour éviter toute confusion ce manuel parle toujours des 'pattes' faisant référence à l'emplacement physique externe de la pin d'entrée/sortie.

Patte	Description	Notes
1	Positive Supply, V	Utiliser une batterie ou alim de 3V à 5V
2	Serial In	Utilisée pour le téléchargement
3	Pin 4	Entrée ou Sortie
4	Pin 3	Entrée seulement
5	Pin 2	Entrée ou Sortie
6	Pin 1	Entrée ou Sortie
7	Pin 0 / Serial Out	Sortie seulement. Utilisée pour le téléchargement
8	Ground, G	Connecter à l'alimentation (0V)

Note spéciale - Pin sortie 0
Pin 0 (patte 7) est utilisée pendant le téléchargement du programme, mais peut également être utilisée comme une sortie normale une fois le téléchargement terminé. Sur les circuits d'essai un cavalier permet que la patte du microcontrôleur soit connectée à la prise de téléchargement (position PROG) ou à la sortie (position OUT). Rappelez-vous de déplacer le cavalier dans la position correcte lorsque vous testez votre programme!

Si vous faites votre propre circuit, vous pouvez inclure un cavalier similaire ou un petit interrupteur, ou vous pouvez préférer connecter la patte du microcontrôleur à la fois à la sortie et au connecteur de téléchargement en même temps. Dans ce cas, vous devez vous rappeler que votre périphérique de sortie passera rapidement on et off lors d'un téléchargement (pas un problème avec sorties simples comme des LED, mais pourrait causer des problèmes avec d'autres appareils tels que des moteurs).

Sélection entrées ou sorties.
Lorsque de la première mise sous tension des PICAXE-08, toutes les pins sont configurées comme des pins d'entrée (sauf pin0, qui est toujours une sortie). Il existe trois méthodes de réglage des autres pins pour être des sorties (si nécessaire)

Méthode 1 - utiliser une commande qui demande que la pin devienne une sortie.
Ceci est la méthode la plus simple, utilisée par la plupart des utilisateurs dans l'enseignement. Dès qu'une commande qui implique une pin de sortie (tels que high, low,toggle, serout ou sound) est utilisée, le microcontrôleur PICAXE-08 convertit automatiquement la pin en une sortie (et laisse la pin en sortie).
Par conséquent, la façon la plus simple de configuration en sortie est juste de mettre une commande 'low' au début du programme pour chaque pin de sortie. Cela indique au microcontrôleur que la pin devient une sortie, et de veiller que la sortie est la condition low (off).

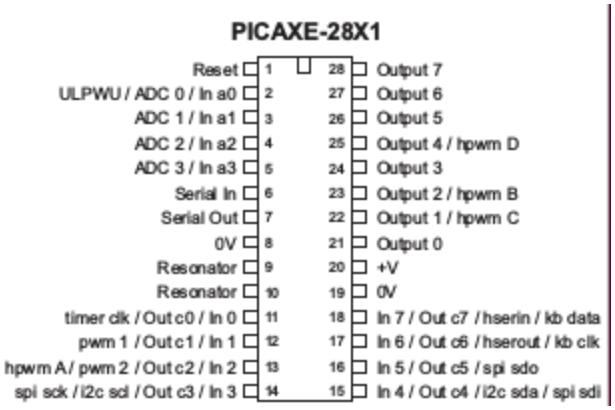
Méthode 2 - utiliser la commande d'entrée et de sortie.
La commande *'output?*' (où ? Est le numéro de pins) peut également être utilisée pour indiquer à la pin d'être une sortie au début d'un programme. De même, la commande '*input'* peut être utilisée pour définir la pin en entrée, bien que ce soit normalement pas nécessaire car la plupart des pins sont définies comme des entrées par défaut. Notez que la commande de sortie ne définit pas la pin comme étant dans un état haut ou bas connu, il est souvent préférable d'utiliser la commande '*low'* à la place.
Les commandes d'entrée et de sortie sont sans effet sur la pin 0 (sortie) et la pin 3 (input), qui ne peuvent être modifiée.

Méthode 3 - (avancé) utiliser la commande let dirs =
La commande ' *let dirs =% 000100111* ' peuvent être utilisée pour définir simultanément toutes les pins en même temps. Cela est plus rapide que l'utilisation de plusieurs commandes d'entrée/sortie, mais nécessite une compréhension des bits binaires (expliqué dans le tutoriel 9).
Placer un 0 pour le numéro de la pin fera de la pin correspondante une entrée, un 1 fera de la pin une sortie. La valeur de bits 0,3,5,6,7 peut être soit 0 ou 1 car ils sont sans effet sur le microcontrôleur et sont tout simplement ignorés.

Sélection pins pour être une entrée analogique.
L'utilisation de la commande *readadc* configure automatiquement la pin en une entrée analogique donc utiliser la commande '*readadc* 1, b2' chaque fois que vous souhaitez prendre une lecture analogique (en supposant l'utilisation de la variable b2 pour stocker la lecture analogique).

Annexe E – Configuration Pins Entrée/sortie PICAXE -28x /28x1

Pour assurer une plus grande flexibilité, la configuration des pin d'entrée / sortie du PICAXE- 28X peut être modifiée par l'utilisateur.
Les paramètres d'alimentation par défaut jusqu'à sont les mêmes que les autres PICAXE-28 (8 entrées, 8 sorties, 4 analogiques).



PORTA (pattes 2 à 5) fournissent 4 entrées analogiques (par défaut) ou jusqu'à 4 entrées numériques

PORTB (pattes 21-28) fournira 8 sorties fixes

PORTC (patte11 à 18) fournira 8 entrées numériques (par défaut) ou jusqu'à 8 sorties.

Cela donne un maximum de 12 entrées numériques, 16 sorties et 4 entrées analogiques

PORTA Fonctions

Patte	Fonctions par Défaut	Seconde Fonction
2	analogique0	porta entrée0
3	analogique1	porta entrée1
4	analogique2	porta entrée2
5	analogique	porta entrée3

PORTB Fonctions
les pins du PORTB sont fixées comme sorties et ne peuvent être changées.

PORTC Fonctions

Patte	Défaut	Seconde Fonction	Fonctions spéciales
11	entrée0	portc sortie0	infrared (entrée)
12	entrée1	portc sortie1	pwm 1 (sortie)
13	entrée2	portc sortie2	pwm 2 (sortie)
14	entrée3	portc sortie3	i2c sd clock (entrée)
15	entrée4	portc sortie4	i2c sda data (entrée)
16	entrée5	portc sortie5	
17	entrée6	portc sortie6	clavier clock (entrée)
18	entrée7	portc sortie7	clavier data (entrée)

Les pins du portC peuvent être utilisées comme entrées par défaut, changées pour sorties, ou utilisées avec leur fonction spéciale via l'utilisation de la commande infrain, keyin, i2c esclave ou pwmout le cas échéant.

Utilisation du porta comme entrées digitales

Les pins 0 à 3 (pattes 2 à 5) du porta sont, par défaut, configurées comme entrées analogiques.
Cependant, elles peuvent également être utilisées en tant qu'entrées numériques simples.

La syntaxe suivante est utilisée pour tester la condition d'entrée :

```
if portA pin 0 = 1 then jump
```

exemple : le mot clé supplémentaire 'portA' est inséré après la commande *'if'* .

pour tester si deux (ou plusieurs) entrées du porta sont on

```
if portA pin 0 = 1 AND pin1= 1  then jump
```

pour tester si l'un des deux (ou plusieurs) entrées porta sont on

```
if portA pin 0 = 1 OR pin1= 1  then jump
```

Notez que la commande porta est nécessaire une seule fois après la commande '*if*.
Il est impossible de tester les entrées sur deux ports différents dans le même *if ... then*.

Il est impossible d'accéder aux pins Porta avec d'autres commandes de type 'input' (count, pulsin etc). Par conséquent, ces pins doivent être réservées pour de simples interrupteurs marche / arrêt.

Utilisation du portc en tant que sorties

Les pins du portc sont, par défaut, des pins d'entrée numériques.
Cependant, elles peuvent également être configurées pour être utilisées en tant que sorties numériques.
Pour convertir la pin en sortie et la rendre high

```
high portc 1
```

Pour convertir la pin en sortie et de la rendre low

```
low portc 1
```

Pour convertir toutes les pins en sorties

```
let dirsc =% 11111111
```

Pour convertir toutes les pins en entrées

```
let dirsc  =% 00000000
```

Notez que '*dirsc'* utilise la notation commune BASIC 0 pour entrée et 1 pour sortie.

(Avancé- Si vous êtes plus familier avec la programmation du code assembleur vous pouvez préférer utiliser la commande 'l*et trisc*=' à la place, car celle-ci utilise notation inversée de l'assembleur
- 1 pour l'entrée et pour la sortie 0. Ne pas essayer de '*poke*' directement le registre de TRISC (commande *poke*) car le bootstrap PICAXE rafraîchit le réglage du registre régulièrement).

Pour commuter toutes les sorties du portc à haut

```
let pinsc =% 11111111
```

(Ou) `let portc =% 11111111`

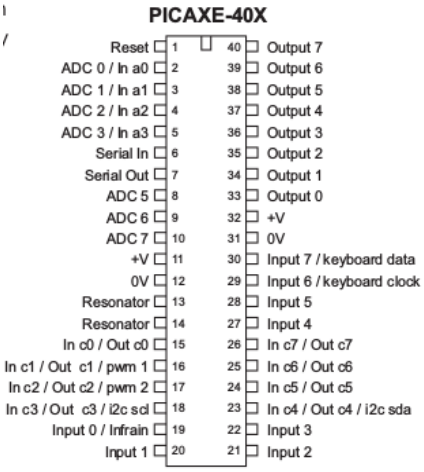
Pour passer toutes les sorties du portc à bas

```
let pinsc =% 00000000
```

(Ou) `let portc =% 00000000`

Annexe F – Configuration Pins Entrée/sortie PICAXE -40x /40x1

Pour assurer une plus grande flexibilité, la configuration des pins d'entrée sortie du PICAXE-40X peut être modifiée par l'utilisateur.



PORTA (pattes 2 à 5) fournit 4 entrées analogiques (par défaut) ou jusqu'à 4 entrées numériques.

PORTB (patte 33-40) fournit 8 sorties fixes.

PORTC (patte 15-18. 23-26) fournit 8 entrées numériques (par défaut) ou jusqu'à 8 sorties.

PORTC (patte 19-22, 27-30) fournissent 8 entrées numériques

PORTE (patte 8-10) fournir 3 entrées analogiques

Cela donne un maximum de 20 entrées numériques, 16 sorties, 7 entrées analogiques

PORTA Fonctions

Patte	Fonction par Défaut	Seconde Fonction
2	analogique0	porta entrée0
3	analogique1	porta entrée1
4	analogique2	porta entrée2
5	analogique	porta entrée3

Fonctions PORTB / PORTE

Les pins du PORTB sont fixées comme sorties et ne peuvent pas être modifiées.
Les pins du PORTE sont fixées comme entrées analogiques et ne peuvent pas être modifiées.

PORTC Fonctions

Patte	Défaut	Seconde Fonction	Fonction Spéciale
15	entrée portc0	sortie portc0	
16	entrée portc1	sortie portc1	pwm 1 (sortie)
17	entrée portc2	sortie portc2	pwm 2 (sortie)
18	entrée portc3	sortie portc3	i2c sd clock (entrée)
19	entrée portc4	sortie portc4	i2c sda data (entrée)
23	entrée portc5	sortie portc5	
24	entrée portc6	sortie portc6	
25	entrée portc7	sortie portc7	
26	entrée portc8	sortie portc8	

Les pins du portc peuvent être utilisées comme entrées par défaut, changé en sorties, ou utilisées avec leur fonctions spéciale par l'utilisation des la commande i2cslave ou pwmout.

PORTCD Fonctions

Patte		Fonction parDéfaut	Fonction Spéciale
19	entrée0	sortie portc0	ifrared (entrée)
20	entrée1	sortie portc1	
21	entrée2	sortie portc2	
22	entrée3	sortie portc3	
27	entrée4	sortie portc4	
28	entrée5	sortie portc5	
29	entrée6	sortie portc6	keyboard clock (entrée)
30	entrée7	sortie portc7	keyboard data (entrée)

Utilisation du porta comme entrées digitales

Les pins 0 à 3 (pattes 2 à 5) du porta sont, par défaut, configurées comme entrées analogiques.
Cependant, elles peuvent également être utilisées en tant qu'entrées numériques simples.
La syntaxe suivante est utilisée pour tester la condition d'entrée :

```
if portA pin 0 = 1 then jump
```

exemple : le mot clé supplémentaire 'portA' est inséré après la commande '*if*' .

pour tester si deux (ou plusieurs) entrées du porta sont on

```
if portA pin 0 = 1 AND pin1= 1  then jump
```

pour tester si l'un des deux (ou plusieurs) entrées porta sont on

```
if portA pin 0 = 1 OR pin1= 1  then jump
```

Notez que la commande porta est nécessaire une seule fois après la commande '*if*.
Il est impossible de tester les entrées sur deux ports différents dans le même *if… then*.

Il est impossible d'accéder aux pins Porta avec d'autres commandes de type 'input' (count, pulsin etc). Par conséquent, ces pins doivent être réservées pour de simples interrupteurs marche / arrêt.

Utilisation portc intrants numériques

Sur le PICAXE-40X le **portd** sont les entrées standards, et donc utilisent la norme
if pin0 = command. Par conséquent, pour les entrées du **portc la** saisie du mot-clé supplémentaire portc doit être utilisée (comme dans l' exemple ci-dessus if porta pin0 =).

Utilisation du portc en tant que sorties

Les pins du portc sont, par défaut, des pins d'entrée numériques.
Cependant, elles peuvent également être configurées pour être utilisées en tant que sorties numériques.
Pour convertir la pin en sortie et la rendre high

```
high portc 1
```

Pour convertir la pin en sortie et de la rendre low

```
low portc 1
```

Pour convertir toutes les pins en sorties

```
let dirsc =% 11111111
```

Pour convertir toutes les pins en entrées

```
let dirsc  =% 00000000
```

Notez que '*dirsc*' utilise la notation commune BASIC 0 pour entrée et 1 pour sortie.

(Avancé- Si vous êtes plus familier avec la programmation du code assembleur vous pouvez préférer utiliser la commande *'let trisc=*' à la place, car celle-ci utilise la notation inversée de l'assembleur
- 1 pour l'entrée et pour la sortie 0. Ne pas essayer de '*poke*' directement le registre de TRISC (commande *poke*) car le bootstrap PICAXE rafraîchit le réglage du registre régulièrement).

Pour commuter toutes les sorties du portc à haut

```
let pinsc =% 11111111
```

(Ou) `let portc =% 11111111`

Pour passer toutes les sorties du portc à bas

```
let pinsc =% 00000000
```

(Ou) `let portc =% 00000000`

Annexe G – FAQ

Où puis-je acheter des microcontrôleurs PICAXE?
Tous les microcontrôleurs peuvent être achetés à partir de la section PICAXE de la boutique en ligne www.tech-supplies.co.uk ou de nos distributeurs (voir www.picaxe.co.uk)

Quel câble - série ou USB?
De nombreux ordinateurs modernes ne disposent pas d'un port série de 9 pins et ainsi nous recommandons toujours le téléchargement USB partie de câble AXE027.
Toutefois, le câble série AXE026 est une option plus économique pour les ordinateurs qui ont toujours des ports série - par exemple dans une salle informatique de l'école.

Il semble y avoir deux câbles PICAXE de téléchargement série - qui dois-je utiliser?
Le câble PICAXE série standard (réf AXE026) se termine par un jack de style stéréo 3,5 mm.
Si vous fabriquez votre propre circuit, nous recommandons ce câble stéréo moins cher car il est de meilleure qualité, et notre circuit modèle utilise ce connecteur (réf CON039).
Le câble original PICAXE-28 (réf AXE025) est terminé par un connecteur 3 pins en ligne, mais ce câble n'est plus utilisé sur l'un de nos circuits projet ou exemples de circuit

J'ai construit un second circuit (sans le circuit de téléchargement) et le programme de PICAXE ne s'exécute pas!
Si vous programmez une puce PICAXE dans un autre circuit, puis déplacez la puce sur un autre circuit sans le circuit de téléchargement, vous devez vous assurer que la patte 'serial in' est liée à la masse (0V) sur la deuxième carte pour un fonctionnement fiable.

Je l'ai acheté quelques circuits vierges et ils ne fonctionnent pas dans le système de PICAXE!
Le microcontrôleur PICAXE n'est pas un microPIC vide! C'est un microcontrôleur qui a été pré-programmé avec un programme de «bootstrap» qui permet le téléchargement via la liaison directe par câble (le programme d'amorçage indique au microcontrôleur comment interpréter les commandes directes de programmation par câble).
Par conséquent, vous devez acheter des microcontrôleurs 'PICAXE', plutôt que de microcontrôleurs vierges, pour les utiliser avec le système PICAXE.
Cependant, nous vendons des microcontrôleurs PICAXE à env. le même prix que les composants vierges, donc il y a très peu de différence de prix pour l'utilisateur final, surtout si vous achetez les multi-packs.

J'ai programmé un microcontrôleur PICAXE l'aide d'un programmateur classique et désormais il ne fonctionne pas dans le système de PICAXE!
Vous avez écrasé, et donc supprimé, le programme d'amorçage PICAXE (voir ci-dessus).
Le microcontrôleur ne peut plus être utilisé comme un microcontrôleur PICAXE, mais vous pouvez naturellement continuer à l'utiliser avec votre programmeur classique.

Pouvez-vous reprogrammer des microcontrôleurs (que je ai accidentellement effacées) avec le programme d'amorçage?
Non, nous n'acceptons pas de microcontrôleurs provenant de sources inconnues en raison des procédures de stockage/manutention correctes exigées par ces dispositifs.
Nous utilisons des équipes de programmeurs coûtant plusieurs milliers de livres pour programmer le code d'amorçage dans les microcontrôleurs vierges, et ainsi devons protéger cet équipement coûteux contre les dommages.
Il est également probable que si nous ne proposons pas ce service, c'est que le coût du transport finirait par être plus cher que les nouveaux microcontrôleurs PICAXE de toute façon !

Pouvez-vous fournir le programme d'amorçage pour que je puisse faire mon propre PICAXE?
Non, la petite redevance reçue sur chaque puce PICAXE vendue est le seul avantage financier pour notre société pour soutenir le système PICAXE - le logiciel est gratuit et les câbles/kits de développement sont vendus à très bas prix.
Par conséquent nous ne permettons pas à quelqu'un d'autre de fabriquer des microcontrôleurs PICAXE.

Puis-je voir le code assembleur qui est téléchargé dans le PICAXE?
Si vous possédez un Programmeur PIC Serie Révolution vous pouvez convertir les programmes BASIC PICAXE en code assembleur, pour programmer des PIC vierges ou simplement apprendre le code assembleur en travaillant par 'désassemblage'.
Toutefois, certaines des commandes les plus complexes (ex : programmes multitaches M2) ne sont pas prise en charge, et le programme de code assembleur généré est optimisé pour l'apprentissage séquentiel (pas optimisé pour la compacité comme avec le système PICAXE) et ainsi le code n'est pas 100% identique à celui téléchargé par le PICAXE.

Pouvez-vous modifier la pin agencement d'entrée / de sortie du microcontrôleur PICAXE?
Les PICAXE-08 / 08M, et toutes les réf X2 et M2 ont des pins configurables.
Les autres réf ont essentiellement des i/o fixes, bien que certaines pins peuvent être modifiées - voir les annexes à la fin du manuel 1 pour plus de détails.

Quelle longueur de programme puis-je télécharger dans le microcontrôleur PICAXE?
Cela varie selon les commandes utilisées, car toutes les commandes n'utilisent pas la même quantité de mémoire.
En règle générale, vous pouvez télécharger :
40-110 lignes de code dans PICAXE-18.8
80-220 lignes de code dans PICAXE-08M / 14M / 20M / 18A / 18M / 28 / 28A
600-1800 lignes de code dans PICAXE-14M2 / 18M2 / 20M2 / 18X / 28X / 40X
2000-3200 lignes de code dans PICAXE-20X2 / 28X1 / 28X2 / 40 X 1 / 40X2
Cependant, certaines commandes, telles que *sond* et *serout* utilisent plus de mémoire et ainsi réduira ce nombre.
Dans notre expérience, la plupart des programmes d'enseignement qui sont trop longs à télécharger sont généralement mal composés, et peuvent être considérablement réduit en taille par l'utilisation de sous-procédures, etc.

Ai-je besoin d'effacer l'appareil?
Comment puis-je arrêter un programme dans le microcontrôleur PICAXE courir?
Chaque téléchargement remplace automatiquement l'ensemble du programme précédent.
Il n'est généralement pas nécessaire d'effacer la mémoire à tout moment.
Toutefois, si vous voulez arrêter un programme en cours d'exécution, vous pouvez sélectionner le menu 'Effacer de la mémoire matérielle' pour télécharger un programme'vide' dans la mémoire PICAXE.

Combien de fois le microcontrôleur PICAXE peut être reprogrammé?
puces PICAXE peuvent être reprogrammées au moins 100.000 fois.
Noter que ce sont les valeurs minimales et les valeurs réelles peuvent être beaucoup plus grande.

Puis je commander des servos par l'utilisation de PICAXE
Oui, de nombreuses références ont une commande 'servo' qui permet de contrôler jusqu'à 8 servomoteurs (un sur chaque sortie).

Puis-je contrôler un écran LCD?
Oui, le PICAXE supporte les modules séries LCD (comme le module Serial LCD / Horloge AXE033) via la commande *serout*.
A noter que le module AXE033 peut également être pré-programmé avec jusqu'à huit messages pour réduire l'utilisation de la mémoire du microcontrôleur PICAXE.

À quelle vitesse le PICAXE fonctionne?
Les microcontrôleurs PICAXE-08/18 ont un résonateur de 4 MHz interne et le PICAXE-28/40 utilise un résonateur céramique de 4 MHz externe. Cela signifie que le microcontrôleur traite 1 million de commandes assembleur à la seconde, ce qui équivaut à environ environ 1.000 commandes BASIC par seconde.
Les références M et X peut être surcadencées à 8 ou 16 MHz (multiplie la vitesse en x2 ou x4).

Est-ce que les interruptions sont supportées par PICAXE?
Oui. De nombreuses références prennent en charge une interruption testée sur le port d'entrée.
Utilisez la commande *'setin'* ou *'setpaintflag*s' pour configurer le paramètre du port d'interruption souhaitée.

Comment puis-je créer temps retarde de plus de 65 secondes?
La meilleure façon de créer de longs retards est de faire des retards en minute avec une boucle, par exemple attendre une heure (60 minutes)

```
for b2 = 1 to 60      'start a for..next loop
    pause 60000       'wait 1 minute
    next b2           'next loop
```

Le microcontrôleur PICAXE travaille à 4MHz nominal, mais en raison des tolérances de fabrication, il est susceptible d'avoir une dérive de quelques secondes sur de longues périodes de temps (par exemple par jour).
Notez que le module Serial LCD / Clock (AXE033) a une horloge de précision et la fonction 'clock' peut être utilisée pour déclencher le PICAXE à intervalle prédéfini ou à certains durée/dates avec beaucoup plus de précision.
Les références X peuvent également être liées à l'horloge en temps réel I2C DS13097.

Mon programme est trop long! Que puis-je faire?
Conseils pour réduire la durée du programme (voir les commandes de base du fichier d'aide pour plus de détails) :
1) l'utilisation *'let pins* = ' à la place de multiples commandes high/low
2) Utilisez des sous-procédures pour le code répété
3) Essayez de réduire l'utilisation des commandes *sound* et *serout* qui utilisent une grande quantité de Mémoire
4) Si vous utilisez un écran LCD, stocker les messages dans le module LCD série AXE033 plutôt
que dans le programme
5) Utiliser EEPROM et de lire des commandes pour stocker les messages dans la mémoire de données (voir la prochaine page)
6) restructurer votre programme en visant à réduire le nombre de commandes *'goto'*
7) Utilisez une puce PICAXE avec la plus grande mémoire (référence X1 ou X2)

Vous pouvez utiliser le menu 'PICAXE-> Vérifier la syntaxe' pour tester la durée de votre programme sans téléchargement.

Les symboles augmentent-ils la durée du programme?
Non, tous les symboles sont reconvertis en 'nombres' par le logiciel de l'ordinateur avant d'être télécharger et n'ont donc aucun effet sur la durée du programme. Vous pouvez utiliser autant de symboles commandes que vous le souhaitez.

Quelles sont les notes générées par la commande du son?
La commande du son génère différents 'bip' retentit pour les valeurs 1-127.
Les commandes mélodie et jouer du PICAXE-08M sont spécifiquement conçus pour jouer des airs. Voir la commande de mise au point dans le manuel 2 pour plus de détails.

Je veux plusieurs sorties - que dois- je faire?
Utilisez les PICAXE-28X/40X ou 28X1/40X1 qui peuvent avoir jusqu'à 16 sorties. Ou connecter une seule sortie (par exemple sortie7) d'une première puce PICAXE à input0 d'un deuxième PICAXE-18 puce. Programmez le deuxième PICAXE-18 avec ce simple programme :

```
main :    serin 0, N2400, b1
          let pins = b1
goto main
```

Les huit sorties de la seconde puce peuvent maintenant être contrôlés avec un serout 7, N2400, (b2) commande par la première puce, où b2 contient la valeur 'pins' 0 à 255) souhaitée sur la deuxième puce. Cela vous donne un total de 15 sorties utilisables.

Je veux plusieurs entrées - que dois- je faire?
Utilisez un PICAXE-28X1 ou 40X1, qui peut être configuré pour avoir un grand nombre d'entrées.
Rappelez-vous que les entrées analogiques peuvent également être utilisés comme entrées numériques si nécessaire, voir juste si la valeur 'de readadc' est supérieur ou inférieur à 100.
Dans de nombreuses applications les commutateurs peuvent également être connectés en parallèle sur une pin d'entrée unique.

Comment puis-je tester plus d'une entrée à la fois?
Utilisez la commande suivante pour tester deux entrées ensemble

```
if pin0 = 1 AND pin1 = 1 then ...
```

ou l'une des deux entrées

```
if pin0 = 1 OR pin1 = 1 then ...
```

Annexe I - Information Techniques Avancées et FAQ

Cette annexe fournit des données techniques avancées pour les utilisateurs qui souhaitent comprendre données techniques plus avancées sur les microcontrôleurs PICAXE. Ces informations ne sont pas requises pour l'utilisation normale de PICAXE. Ces notes supposent que l'utilisateur est familier avec les microcontrôleurs PIC, leurs paramètres de fusibles de configuration et la programmation en code assembleur.

Qu'est-ce qu'un microcontrôleur PICAXE?
Un microcontrôleur PICAXE est un microcontrôleur Microchip PIC qui a été préprogrammé avec le code d'amorçage PICAXE. Le code d'amorçage permet au microcontrôleur d'être reprogrammé sans la nécessité d'un (coûteux) programmeur classique, ce qui rend l'ensemble du système abordable, un câble série très simple, un faible coût de téléchargement!
Le code d'amorçage contient également des routines communes (telles que la façon de générer une pause ou une sortie audio), de sorte que chaque téléchargement ne perd pas de temps à télécharger ces données couramment requises. Cela rend le temps de téléchargement beaucoup plus rapide.

Pourquoi utiliser PICAXE à la place de l'assembleur /C?
Le PICAXE utilise un langage BASIC simple (ou organigrammes) et les étudiants plus jeunes
peuvent commencer à générer des programmes en moins d'une heure lors de la première utilisation. Il est beaucoup plus facile à apprendre et à déboguer que le code assembleur ou le C.

Le second avantage est la méthode de téléchargement directe par câble. Le logiciel est gratuit et donc le seul coût par ordinateur est un câble de téléchargement à faible coût. Cela permet aux étudiants d'acheter leur propre câble et pour les écoles d'équiper chaque ordinateur avec un câble de téléchargement. D'autres systèmes qui nécessitent un programmateur coûteux sont généralement trop onéreux à mettre en œuvre dans ce sens.
Enfin, comme la puce PICAXE ne quitte jamais la carte, toutes les cassures de pattes (qui peuvent se produire lorsque la puce est déplacée d'avant en arrière à partir d'un programmeur) sont éliminées.

La manière dont le programme est stocké dans le microcontrôleur?
Le programme est stocké soit dans la mémoire de données ou de programme en fonction du type de microcontrôleur.
Le tableau suivant montre comment program, read,/write/ EEPROM data et readmem/writemem est stockée.

Programme	Read/Write	Readmem/Writemem
PICAXE-08M2Program/data	Data (256)	N/A (use i2c)
PICAXE-14MProgram	Data (256)	N/A (use i2c)
PICAXE-18M2Program/data	Data (256)	N/A (use i2c)
PICAXE-20M2Program	Data (256)	N/A (use i2c)
PICAXE-18X Program	Data (256)	N/A (use i2c)
PICAXE-28X Program	Data (128)	N/A (use i2c)
PICAXE-28X1Program	Data (256)	N/A (use readtable or i2c)
PICAXE-28X2Program	Data (256)	N/A (use readtable or i2c)
PICAXE-40X Program	Data (128)	N/A (use i2c)
PICAXE-40X2Program	Data (256)	N/A (use readtable or i2c)

Le programme et la mémoire lire/écrire sont écrasés par chaque téléchargement. Utilisez la commande *eeprom* pour précharger les données (dans le programme) pour les commandes de lecture /écriture. La mémoire readmem / writemem ne change pas pendant un téléchargement.

Combien de fois le microcontrôleur peut-il être reprogrammé?
Les puces PICAXE peuvent être reprogrammées au moins 100.000 fois. Noter que ce sont les valeurs minimales et les valeurs réelles peuvent être beaucoup plus grande.

Comment un téléchargement est lancé?
Lorsque l'ordinateur démarre un téléchargement une interruption est générée sur la pin d'entrée série sur le PICAXE. Cela interrompt le programme principal et met le PICAXE dans un état pour recevoir un nouveau téléchargement.
Par conséquent, vous devez vous assurer que la pin 'serial in' est reliée à la masse (0V) via les résistances 22kΩ/10kΩ sur tous les circuit d'essai pour un fonctionnement fiable du microcontrôleur (pour éviter des 'pins flottantes' et des signaux d'interruption indésirables).

Quelles sont les caractéristiques électriques de la PICAXE (par exemple exploitation plage de tension, etc.)?
Les caractéristiques électriques du microcontrôleur PICAXE dépendent du microcontrôleur PIC de base qui est programmé avec le code d'amorçage pour créer le microcontrôleur PICAXE.
Donc voir la fiche Microchip (de www.microchip.com) pour les caractéristiques des microcontrôleurs appropriées.
La plus faible tension de fonctionnement recommandée par ces fiches est 3V (Notez que ceci est la 'tension de fonctionnement' seulement. Vous pouvez avoir besoin d'une tension plus élevée (4,5 V minimum recommandé) tout en faisant le téléchargement série actuel de l'ordinateur pour assurer la programmation de la mémoire précise de la puce).
Les références X2 sont également disponibles en 1.8V variantes spéciales de 3.3V.

Le PICAXE met le chien de garde en service?
Oui, le chien de garde est défini et utilisé dans un certain nombre de commandes telles que *sleep* et *nap* .
L'utilisateur ne peut pas modifier ses paramètres.

Le PICAXE régler la minuterie fusible de la mise sous tension?
Oui.

Est-ce que le PICAXE met la protection sous-tension en service?
Oui pour les références M, M2, X1 et X2, pas pour les autres. Un effet secondaire malheureux du fusible sur les autres références est qu'il restreint le fonctionnement à la plus basse tension du microcontrôleur soit environ 4,2V.
Comme de nombreux utilisateurs souhaitent utiliser des batteries 3V, le fusible n'est pas réglé sur les microcontrôleurs PIC avec un seuil de 4.2V.
La commande *enablebod/disable* permet d'activer/désactiver la fonction de protection sous-tension sur les réf2rences M, M2, X1 et X2.

Comment le PICAXE fait les conversions ADC (analogique-numérique) ?
Le (interrompu) PICAXE-08 et PICAXE-18 ont utilisé le comparateur interne pour faire une comparaison avec un ADC à faible résolution, fournissant 16 valeurs analogiques discrètes.
Les autres microcontrôleurs PICAXE utilisent tous l'ADC interne pour faire une pleine conversion 256 niveaux (8 bits). Bien que les microcontrôleurs soient techniquement capables de 10 conversions de bits, ceci est converti par la commande *readadc* dans l'octet (8 bits) des valeurs pour faciliter l'utilisation via l'octet (b1 etc.) des variables, ce qui rend les mathématiques plus facile pour les étudiants.
Cela donne une résolution d'environ 0.02V (à 5V) qui est suffisant pour la plupart des projets éducatifs. La plupart des références ont aussi une option read adc 10 bits séparé (1024 niveaux), via la commande readadc10.

Pouvez-vous fournir le programme d'amorçage pour que je puisse faire mon propre PICAXE?
Non, le petit redevance effectué sur chaque puce PICAXE vendue est le seul avantage financier pour notre société pour soutenir le système PICAXE - le logiciel est gratuit et les câbles/kits de développement sont vendus à très bas prix.
Par conséquent nous ne permettons pas à quelqu'un d'autre de fabriquer des microcontrôleurs PICAXE.

Puis-je mélanger assembleur avec le code de base?
Non, le programme et le code d'amorçage ne peuvent pas être mixés avec le code assembleur, ce ne sont pas les bonnes pratiques de programmation.
Cependant, vous pouvez atteindre le même but par la conversion de votre BASIC en code assembleur en utilisant la fonction de conversion automatique, puis éditer le programme converti de code assembleur (voir ci-dessous).

Puis-je voir le code assembleur qui est téléchargé dans le PICAXE?
Si vous possédez un Programmeur Serie PIC Révolution (ref BAS800), vous pouvez convertir des programmes PICAXE BASIC en code assembleur, pour programmer des PIC vierges ou simplement apprendre comment fonctionne le code assembleur en 'désassemblant'.
Toutefois, certaines des commandes plus complexes (par exemple serin) ne sont pas pris en charge, et le programme de code assembleur généré est optimisé pour l'apprentissage séquentiel (pas optimisé pour la compacité comme avec le système PICAXE) et ainsi le code n'est pas identique à celui téléchargé sur le PICAXE.

Pouvez-vous modifier la pin agencement d'entrée/de sortie du microcontrôleur PICAXE?
Le PICAXE-08 dispose de 5 pins qui peuvent être configurées comme vous le souhaitez.
Les PICAXE 28 et 40 pins peuvent également être modifiés pour donner plus d'entrées ou de sorties.
Les dispositions des18 pins d'entrée/sortie sont fixes et ne peuvent pas être modifiés.

Quelle longueur de programme puis-je télécharger dans le microcontrôleur PICAXE?
Cela varie sur les commandes utilisées, car toutes les commandes n'utilisent pas la même quantité de Mémoire.
Il n'y a pas de formule 'byte' fixe pour l'utilisation de la mémoire par exemple pause de 5, 50 et pause 500 prendront tous différentes quantités d'espace de mémoire !
Pour calculer l'utilisation de la mémoire, utiliser l'option 'Vérifier la syntaxe 'dans le menu PICAXE. Cela rendra compte de la quantité de mémoire utilisée.

Les symboles augmentent-ils la durée du programme?
Non, tous les symboles sont reconvertis en 'nombres' par le logiciel de l'ordinateur avant de télécharger et n'ont donc aucun effet sur la durée du programme.
Vous pouvez utiliser autant de symboles des commandes que vous le souhaitez.

Ai-je besoin d'effacer l'appareil?
Comment puis-je arrêter un programme dans le microcontrôleur PICAXE?
Chaque téléchargement remplace automatiquement l'ensemble du programme précédent.
Il n'est généralement pas nécessaire d'effacer la mémoire à tout moment. Toutefois, si vous voulez arrêter un programme en cours d'exécution, vous pouvez sélectionner le menu 'Effacer de la mémoire matérielle'pour télécharger un programme 'vide' dans la mémoire PICAXE.

Pourquoi un programme 'vide' est différent de 0 octets?
Chaque programme téléchargé contient des données de configuration, et une commande *'end*' est toujours ajoutée automatiquement à la fin de chaque programme téléchargé. Par conséquent, un programme 'vide' à l'écran ne sera pas générer par un programme de zéro octet. Pour éviter la fin automatique utiliser la directive #no_end.

Les microcontrôleurs sont-ils vulnérables?
Les microcontrôleurs ont un niveau élevé de protection statique intégré dans chaque pin et ainsi généralement les manipuler sans aucune protection statique personnelle dans un environnement éducatif (non-production) est acceptable.

Puis-je utiliser EEPROM I2C avec le PICAXE?
Les références M2, X, X1 et X2 supportent toutes les références i2c via les commandes *hi2cin* et *hi2cout*.

Le PICAXE peut compter des impulsions?
Les références M, M2, X, X1 et X2 prennent en charge la commande de comptage qui peut compter le nombre d'impulsions dans une période définie. Toutes les références prennent en charge la commande *pulsin* pour mesurer la longueur d'une impulsion.

Puis-je contrôler des servos en utilisant le PICAXE?
Puis-je faire une commande PWM d'un moteur utilisant le PICAXE?
Les références M, M2, X, X1 et X2 ont une commande *pwmout* dédiée qui agit sur une ou deux des pins pour le plein contrôle PWM.
Ces références ont également une commande '*servo*' qui permet de contrôler jusqu'à 8 servomoteurs (un sur chaque sortie). Les commandes *servo* utilisent l'horloge interne et une interruption, de sorte que les impulsions sont maintenues 'en arrière-plan' tout le temps que le PICAXE exécute le programme principal.
La commande *servo* produit une impulsion de 0.01ms à 2,55 ms de longueur environ toutes les 20 ms.
Par conséquent, il peut également être utilisé comme une sortie simple PWM d'arrière-plan avec PWM marque : rapports spatiaux entre 1 : 2000 et 1 : 8 (environ).

À quelle vitesse le PICAXE fonctionne?
Puis-je overclocker le PICAXE?
Toutes les références ont un résonateur 4 MHz/8 MHz interne, et la famille PICAXE-28/40 peut éventuellement aussi utiliser un résonateur céramique externe. Cela signifie que le microcontrôleur traite 1 million de commandes assembleur par seconde, ce qui équivaut à environ 1000 commandes BASIC par seconde.
Différentes commandes prennent des moments différents pour exécuter en fonction de la complexité de leur -code assembleur.
Toutes les références peuvent être surcadencées jusqu'à 64 MHz (voir l'Annexe de Surcadençage pour les restrictions).

Pourquoi le PICAXE ne soutient-elle jusqu'à 4800 vitesse de transmission sur les commandes serout /de Serin? Puis-je envoyer et recevoir des données en série par l'intermédiaire du câble de téléchargement?
Les vitesses maximales ont été initialement sélectionnées pour un fonctionnement fiable avec des microcontrôleurs à résonateur interne.
Les premiers résonateurs internes ne sont pas aussi précis qu'un dispositif externe, et une vitesse de transmission plus lente assure un fonctionnement fiable.
Les références M2, X1 et X2 soutiennent des vitesses bien plus élevées via l'équipement EUSART à l'aide de la commande *serout*.
De nombreuses références peuvent envoyer des données via le câble de téléchargement via une commande *'sertxd*' et recevoir des données via la commande '*sertxd'*.

Est-ce que les interruptions sont acceptées par PICAXE?
Le PICAXE utilise les interruptions de microcontrôleurs internes pour certains de ses commandes basiques (par exemple l'asservissement). Par conséquent, les interruptions internes ne sont pas disponibles pour une utilisation générale.
Toutefois, les références A, M et X prennent toutes en charge une seule demande interruption sur le port d'entrée.
Utilisez la commande BASIC '*setint*' pour configurer les paramètres du port d'interruption souhaité pour permettre l'interruption demandée. L'interruption demandée scanne le port d'entrée entre chaque commande BASIC (et en permanence pendant les commandes de pause), et s'active ainsi très rapidement.

Versions Logiciels

La dernière version PICAXE Editor 6 et tous les autres titres peuvent être chargés depuis le site web suivant :
www.picaxe.co.uk

Un forum très actif sur les discussions sur les projets PICAXE, et pour des supports techniques, existe aussi à :
www.picaxeforum.co.uk
http ://www.picaxeforum.co.uk/forumdisplay.php?44-Le-forum-officiel-PICAXE-francophone

Adresse Contact
Revolution Education Ltd
http ://www.rev-ed.co.uk/

Remerciements
Révolution Education voudrait remercier :

Clive Seager
John Bown
LTScotland
Higher Still Development Unit
UKOOA