Ce document est une partie de la traduction du document trouvé sur cette page
http://www.picaxe.com/docs/pe6.pdf
Ce n'est pas une traduction officielle mais simplement une aide destinée aux » picaxiens » qui ne maitrisent pas bien la langue anglaise.
les images d'illustration sont en grande partie celle du document d'origine
le chemin « fichier options langue » permet d'afficher les menus en français

Section 2 : Fonctions clés

Barre de menu et barre d'outils

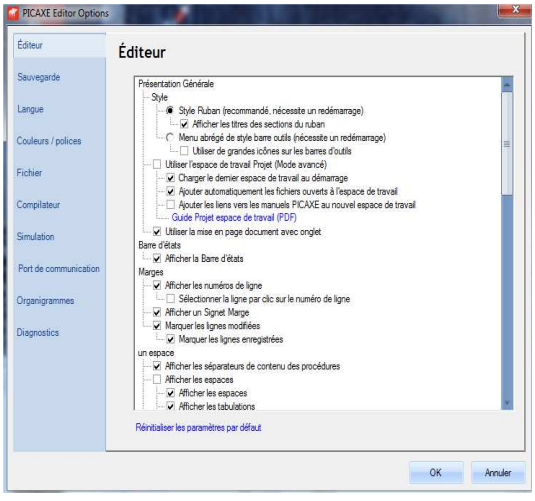
En mode Legacy, les barres de menu et d'outils traditionnels sont utilisés. La taille de l'icône Barre d'outils peut être standard (16x16 pixels) ou grande (32x32pixels). Les menus et barre d'outils peuvent être attachés ou flottant et positionnés au besoin. Cliquez droit sur une barre de menu pour montrer ou cacher les diverses barres d'outils.

Ribbon

Dans le nouveau mode Ribbon (rubans), les trois onglets de rubans (Home, Simulation et Picaxe) fournissent un accès rapide à toutes les fonctionnalités de l'application. Le QAT (Barre d'outils à Accès Rapide) et une fonction standard du ruban qui ajoute une petite barre d'outils à la barre de titre en haut de la fenêtre de l'application. Les boutons QAT par défaut comprennent annuler, refaire, et sauvegarder mais le QAT est également entièrement personnalisable : par exemple, vous pouvez ajouter Syntax Check et Program Button si vous le désirez.

En mode Ribbon, le menu Fichier utilise le format Windows standard qui fournit beaucoup plus d'informations que ce qui est disponible en mode Legacy (par exemple sur l'onglet Imprimer, la fonction Aperçu avant Impression est directement affichée)

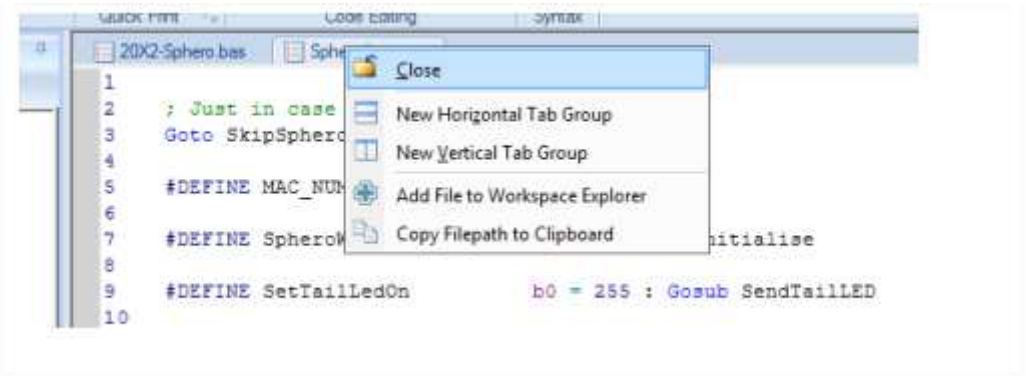
Option



La fenêtre de dialogue Options est accessible depuis le menu fichier > options et inclut une plus grande variété d'options qu'avant. Nous recommandons à tous les nouveaux utilisateurs de prendre un moment pour lire toutes les options de chaque onglet, car cela donne un très bon aperçu de toutes les nouvelles fonctionnalités de PE6.

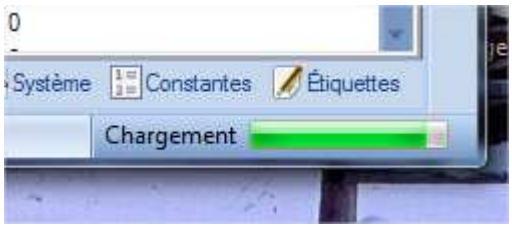
MDI (Multiple Document Interface)

Plusieurs documents peuvent être soit attachés ou flottants (en cascade). En mode attachés, plusieurs groupes verticaux et horizontaux peuvent être réarrangés comme voulu (faites un simple clic droit sur l'onglet pour ouvrir un menu vous permettant d'ajouter de nouveaux groupes verticaux / horizontaux.



Téléchargement de programmes

La programmation est maintenant faite en tâche de fond, pour qu'il soit possible de continuer à utiliser le logiciel pendant qu'un programme (ou un debuggage) est en cours. La barre de progression s'affiche dans la barre de statut (en bas à droite) et la fin d'un téléchargement est également notifiée via un pop-up de la barre des tâches.



PE6 et espace de travail
PE6 a maintenant deux modes opératoires : Simple et « Espace de travail de projets avancés ». Ce mode se sélection via l'onglet Fichier>Options>Editor

1) Mode Simple.

En mode simple les programmes PICAXE sont généralement contenus dans un seul fichier .bas (BASIC) ou .plf (organigramme). Cependant, plusieurs fichiers peuvent encore être ouverts en même temps, et la disposition des documents visibles est maintenue à chaque fois que PE6 est fermé ou redémarré. Les fichiers individuels sont accessibles via le menu File>Open>Save/Close

Ce mode est similaire dans son opération à PICAXE Programming Editor v5

2) Mode Advanced Project Workspace

Un « project workspace » est une collection de fichiers associés dans un projet, ainsi qu'un conteneur pour des réglages spécifiques (par exemple type de puce PICAXE). Un workspace peut être considéré comme un « index » (une méthode pour accéder rapidement à tous les fichiers requis dans ce projet).

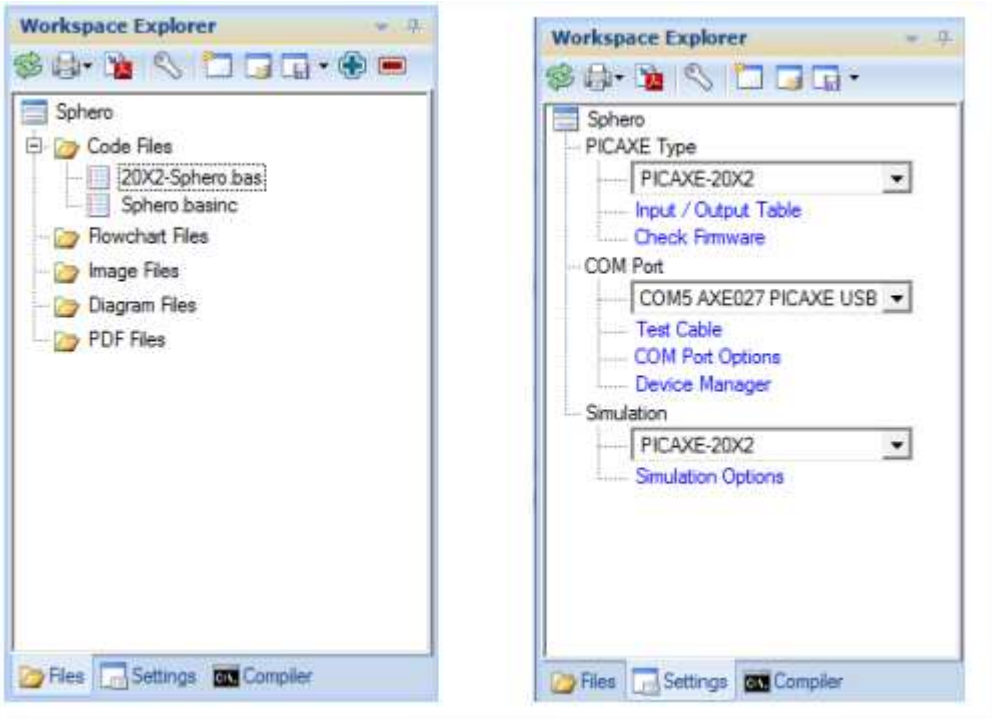
En mode project workspace on peut allouer à chaque projet sa propre disposition séparée. C'est très utile quand chaque projet contient plusieurs fichiers (par exemple un fichier . Bas principal, quelques fichiers .basinc et même des images telles que des dispositions PCB ou des diagrammes schématiques).

La principale différence avec le mode simple est que lorsque vous ouvrer un projet précédent, vous ouvrez le fichier project workspace (.wsp), par des fichiers individuels de code. Donc, l'action principale quand on passe de projets en projet se faire avec les boutons Ouvrir / Sauvegarder Workspace, pas avec les boutons Ouvrir / Sauvegarder fichiers.

Le processus d'ouverture d'un fichier workspace .wsp ouvre donc tous les autres fichiers individuels contenus dans le workspace. Tout fichier associé avec le projet (affichés dans l'arbre de fichiers dans l'explorateur de Workspace) peut aussi être accédé facilement en double-cliquant sur le nom du fichier.

De la même manière, les réglages tels que le type de PICAXE, le port COM préféré, l'image de simulation etc. sont aussi maintenus dans le fichiers workspace .wsp. Par conséquence, par exemple, lorsque vous ré-ouvrez un Workspace le type de PICAXE sélectionné changera automatiquement pour le type précédemment sauvegardé pour ce projet en particulier.

Panneau d'exploration de Workspace



L'explorateur de Workspace a trois onglets, bien que le le premier onglet « files » n'est pas requis en mode « simple » et n'est donc seulement visible lorsque le mode « avancé » est sélectionné.

L'onglet « Files » montre un index de tous les fichiers associés avec ce « project workspace ». Les fichiers peuvent facilement être ajoutés ou retirés en utilisant les boutons de la barre des tâches. Ouvrir un fichier via le menu Fichier ne l'ajoute pas automatiquement à l'index du Workplace, bien que vous puissiez rapidement le faire via le menu « add to workspace » disponible sur l'onglet de ce fichier.

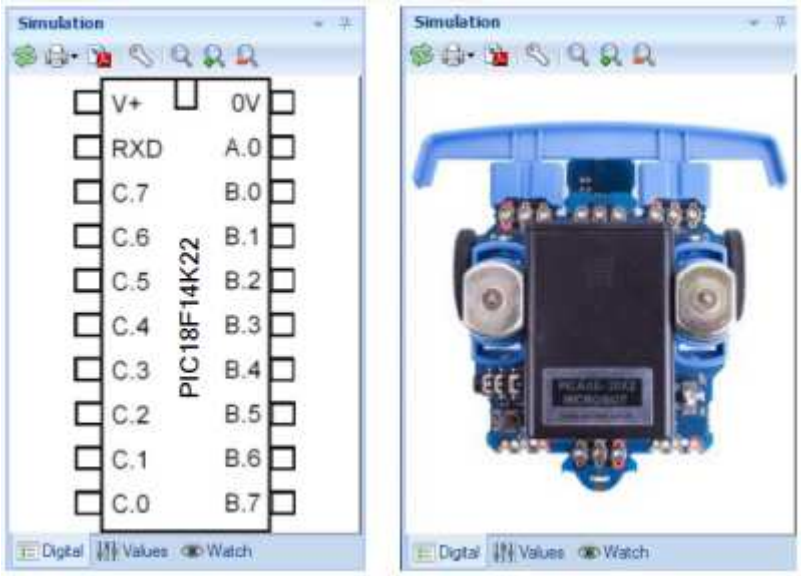
Les fichiers sont regroupés par leur extension : fichiers de code BASCI, fichiers Organigramme, fichiers Images (par exemple la photo ou l'image d'une disposition PCB), les fichiers de diagramme PICAXE (par exemple un diagramme de circuit), et enfin les feuilles de donnée en PDF. Double-cliquez sur un fichier dans l'arbre d'index l'ouvrira.

L'onglet « Settings » est utilisé pour régler le type de puce PICAXE, le port COM et le diagramme de simulation utilisé par ce projet. Ces valeurs sont sauvegardée dans le workspace pour que, par exemple, le bon type de puce PICAXE soit automatiquement sélectionnée la prochaine fois que le workspace est ouvert.

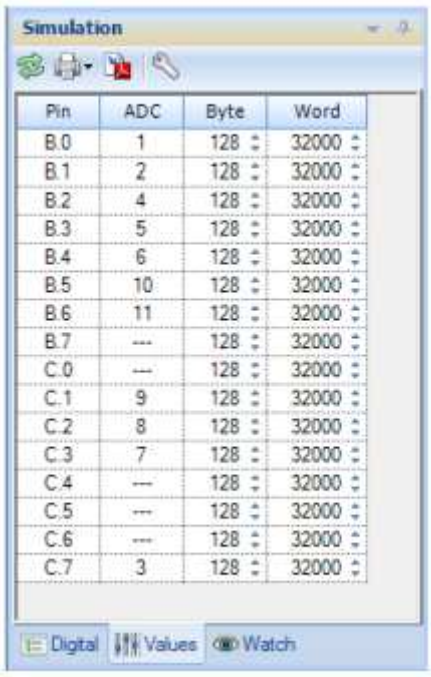
Sont également inclus dans « Settings » diverses fonctions communes de liens raccourcis. Une nouvelle fonctionnalité disponible ici est le « Input/Output Table », qui est une fonction souvent requise qui affiche instantanément la patte de sortie sélectionnée de la puce PICAXE et le sommaire de la fonctione de la patte directement dans l'éditeur.

L'onglet « Compiler » affiche un message de feedback brut des compilateurs et pré-processeurs PICAXE.

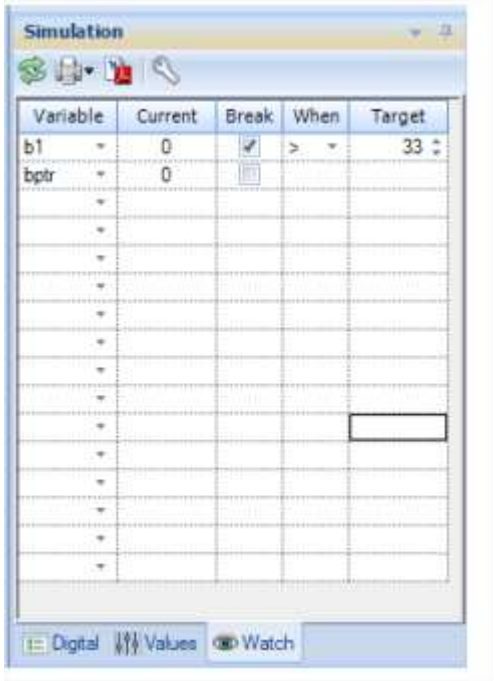
Simulation Panel



Le panneau de simulation affiche un diagramme interactif de simulation. Celui-ci affiche par défaut le pinout de la puce PICAXE sélectionnée, mais peut être changée en tableau de projet interactif ou même en photo de votre propre projet (les utilisateurs peuvent construire leur propre simulation interactive au besoin). Le diagramme de simulation montre l'état des sortir durant une simulation, et permet aussi au entrées d'être changées via quelques clics sur l'entrée appropriée. Au besoin le diagramme de simulation peut être détaché pour devenir une fenêtre flottante via un clic droit.

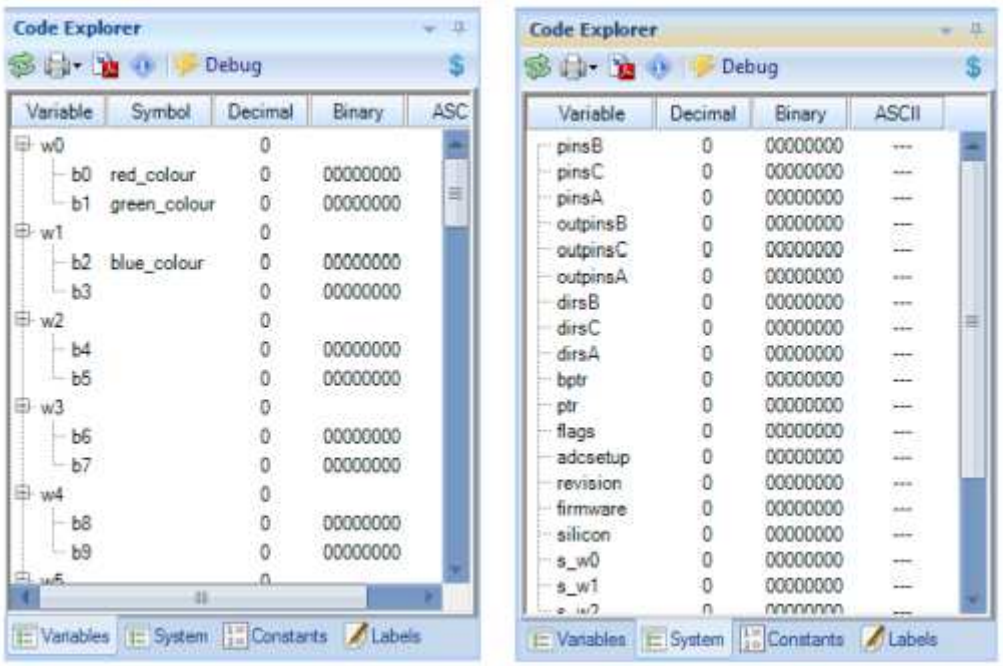


Le deuxième onglet « valeurs » permet pendant une simulation des commandes telles que « readadc » et « pulsin » etc. Les commandes orientées Byte (par exemple readadc) utilise la valeur dans la colonne bute, les commandes orientées mot (par exemple pulsin) utilise la valeur dans la colonne « mots ». Ce nouvau system byte/mot dépasse les limitations d'un seule boite d'entrée « générique » dans PE5.



Le troisième onglet « watch » est une nouvelle fonctionnalité qui permet à des points de rupture d'être réglé sur n'importe quel variable RAM ou System RAM. C'est également utile pour comparer les valeurs cote à cote lorsqu'une simulation est en marche, par exemple « b23 » et « pinsB » peuvent être placés l'un à côté de l'autre pour faire des comparaisons.

Panneau Explorateur de Code



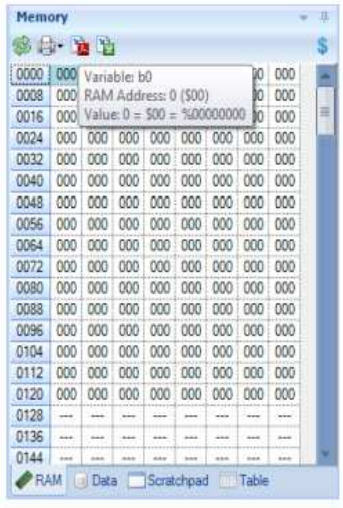
Le panneau d'explorateur de code montre les valeurs des variables, les variables du système principal et les constantes définies par l'utilisateur. Les labels utilisés dans le programme sont également affichés.

Dans PE5 la fenêtre « debug » était une fenêtre modale, ce qui veut dire qu'aucun programme d'édition ne pouvait être entrepris pendant qu'une session de debuggage était en cours. Dans PE6 la fonction debugger a été entièrement intégrée dans l'explorateur de code pour qu'il agisse comme une tâche de fond. Cliquez simplement sur le bouton « debug » pour commencer la session de debuggage (ou télécharger un programme contenant la fonction debugger) puis continuer à travailler normalement.

De plus, la nouvelle directive #no_debug peut être utilisée pour désactiver toutes les commandes de débuggage dans votre programme, pour qu'elles soient simplement ignorées et non incluses dans aucun téléchargement.

Si un conflit de variables est identifié dans un programme (par exemple l'utilisation de b1 et w0 comme deux symboles différents) un résumé d'alerte peut être imprimé depuis le bouton « Variable Clash » sur la barre d'outils.

Memory Panel



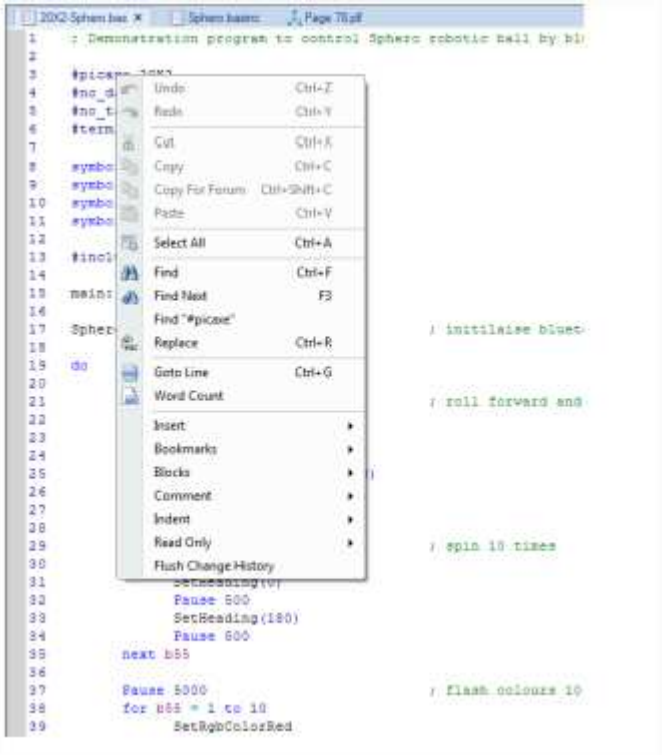
Le panneau « mémoire » montre toutes les valeur de RAM, données EEPROM, scratchad RAM et aires de tables de mémoire. L'affichage est par défaut sur décimales, mais peut être changé en hexadécimal en cliquant sur le bouton « $ ».

Sur les grilles RAM et scratchpad une des cellules sera toujours surlignées d'une couleur plus sombre. C'est la cellule actuellement pointée par la variable bptr/ptr, et ainsi la valeur affichée dans cette cellule surlignée est la valeur actuelle à laquelle @bptr/@ptr aura accès.

Pendant une simulation vous pouvez double-cliquez sur n'importe laquelle des valeurs RAM pour changer sa valeur.

Éditeur de Code

L'éditeur de code comprend beaucoup de fonctionnalités nouvelles et améliorées. Pratiquement toutes les fonctionnalités peuvent être activées/désactivées via la fenêtre de dialogue File>Options



Cliquez droit en éditant pour afficher le menu contextuel étendu, qui fournit un accès rapide aux fonctionnalités principales d'édition, dont les nouvelles fonctionnalités d'édition telles que le bloc commenter/ne pas commenter.

Marge

La marge affiche le numéro de la ligne, les favoris et les points de rupture. Un bande colorée verticale optionnelle surligne aussi les changements faits au code depuis la dernière ouverture et depuis la dernière sauvegarde.

Infobulles interactives

Les infobulles interactives fournissent maintenant de nouvelles informations additionnelles sur tous les types de contenu de programme BASIC.

- Commandes – l'infobulle fournit une commande de structure basique
- Variables – l'infobulle montre la valeur de la variable en cours en décimale, hexadécimal et binaire.
- Constantes – l'infobulle montre la valeur constante en décimal, hexadécimal et binaire.
- Nombres – l'infobulle montre la valeur dans d'autres formats (par exemple décimal et binaire pour un nombre en hexadécimal)
- Série – l'infobulle montre les valeurs ASCII des caractères dans cette série
- Broches – l'infobulle montre les fonctions principales de la broche

Support de navigation assistée avec barres de défilement améliorées et souris intelligente

Fonction « trouver et remplacer » améliorée, avec toute les instances trouvées en favori

Points de rupture et favoris sauvegardés avec le fichier.

De nombreux raccourcis claviers sont maintenant supportés, par exemple Ctrl-V et Shift-Ins pour « coller ».

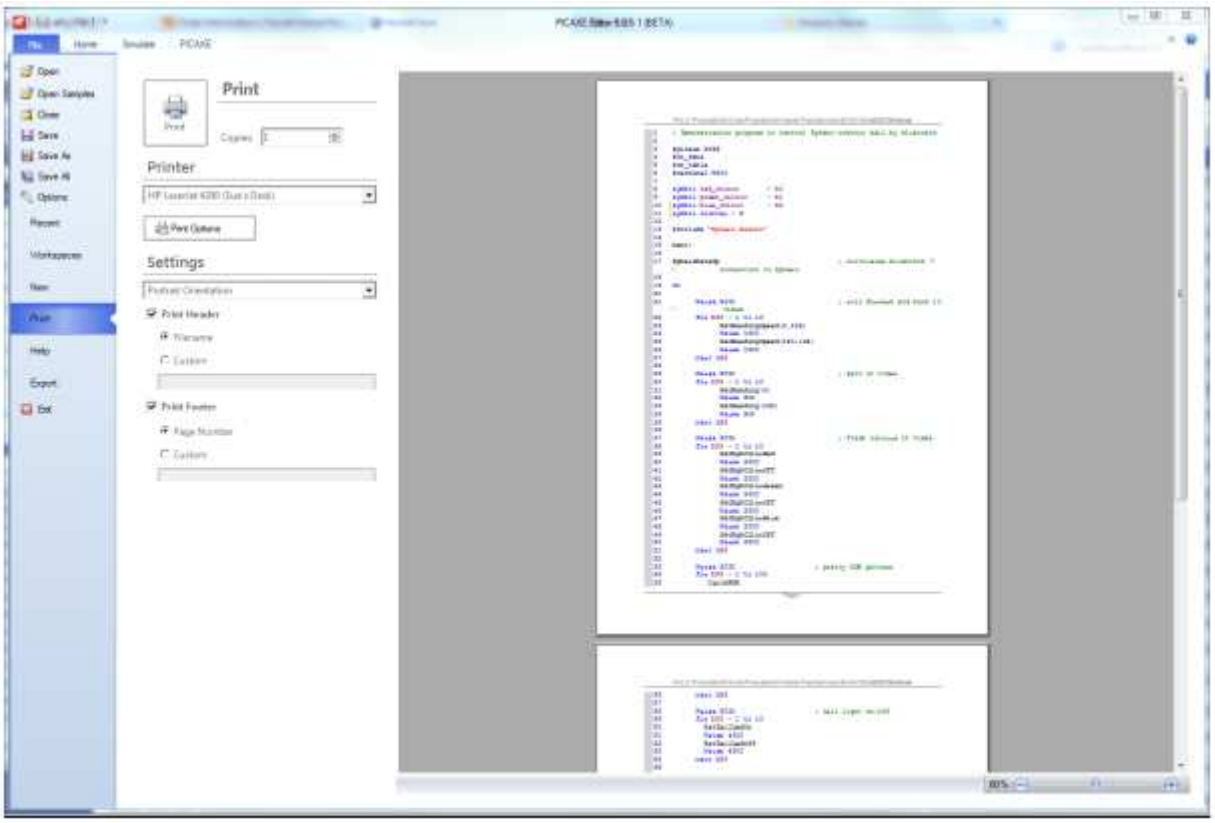
Visionneuse d'images

La fenêtre d'image peut être utilisée pour afficher les types d'image les plus communs (jpg, png etc) qui seront en fait des photos du projet, des organigrammes PCB, schémas, etc.

Visionneuse de diagrammes

La fenêtre « diagrammes » est utilisée pour afficher les diagrammes interactifs PICAXE. Cela peut être un digramme de simulation, un organigramme de platine d'expérimentation ou un diagramme de circuit, etc. Voire le Picaxe Diagram Builder (en développement) pour plus de détails.

Imprimer et Exporter en PDF

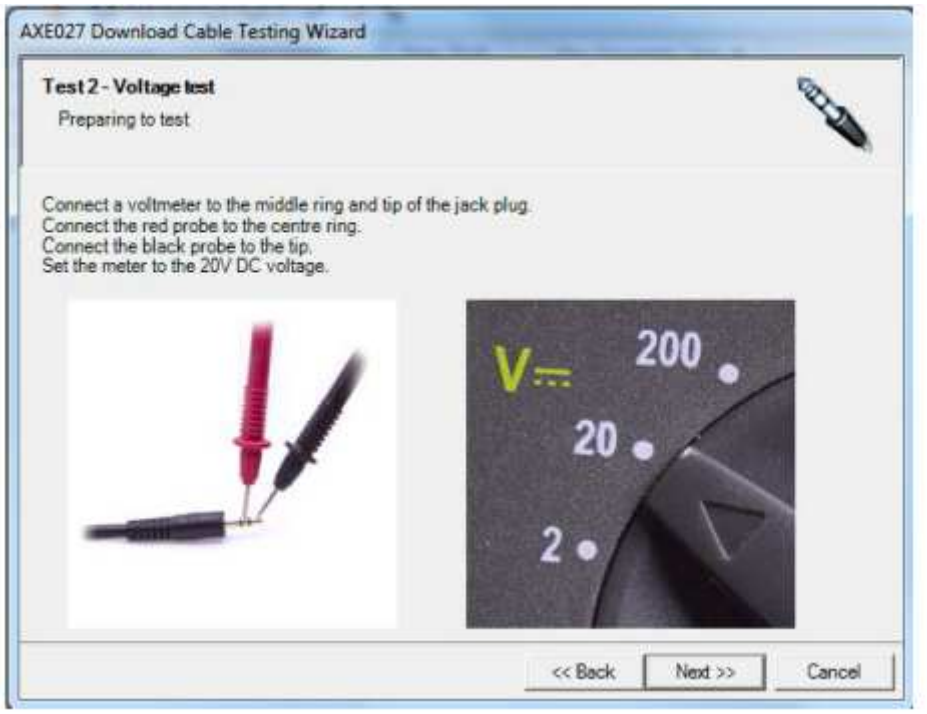


Les fonctions imprimer, aperçu avant impression et exportation PDF ont été énormément améliorées dans PE6. Vous pouvez maintenant exporter / imprimer presque n'importe quel espace de l'écran, y compris le workspace et les panneaux.

Wizards

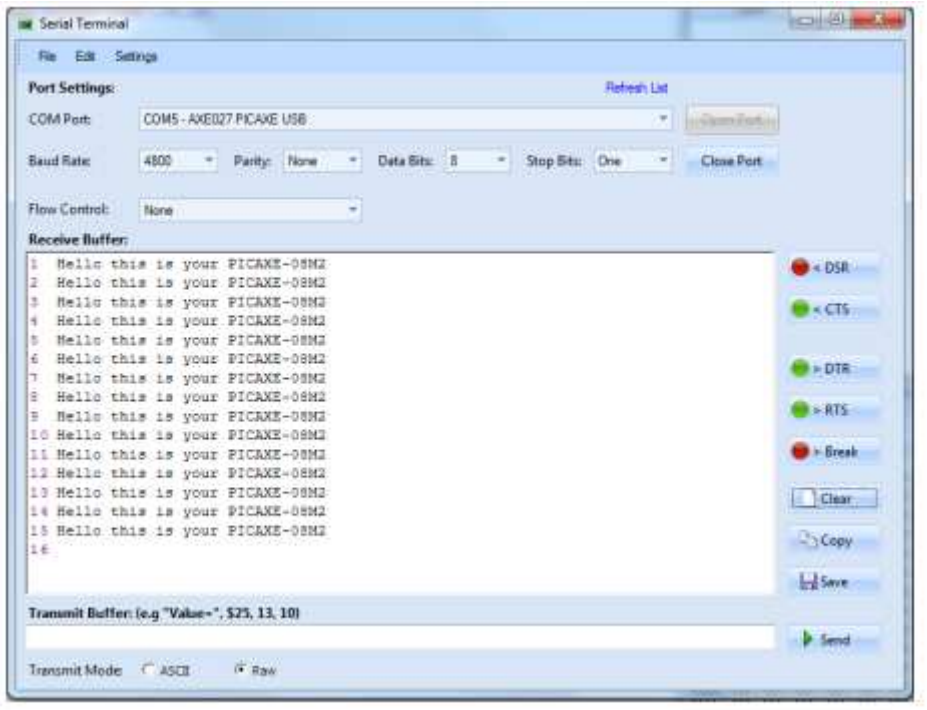
Les (code wizards) utilisent à présent un module de plug-in externe, ce qui permet à de nouveaux

(wizards) d'être ajoutés beaucoup plus facilement au moment et à l'endroit requis. Tous les (wizards) ont été réécris de zéro, et quelques uns tels que « pwmout » (wizard) ont maintenant des fonctionnalités supplémentaires et/ou améliorées.



Les nouveaux (wizards) tel que le troubleshooter de AXE027 sont maintenant aussi inclus.

Serial Terminal



Le Terminal Série a été significativement amélioré. Les nouvelles fonctionnalités clés incluent :

- Peut utiliser un port COM séparé du port programmation du PICAXE sur les machines à plusieurs ports.
- Tous les réglages de port comme « handshaking » et « stop » sont maintenant disponibles.
- Les signaux de contrôle tel que CTS, RTS, etc sont maintenant visibles
- Non-modal, peut donc être utilisé en même temps que l'éditeur principal.
- Les modes d'affichage Hexadécimal, décimal ou ASCII ont des codes-couleur.
- Affichage des caractères ASCII non-imprimés reçus.
- Les caractères de contrôle peuvent être optionnellement utilisés pour naviguer sur la boîte de réception.

La boîte de texte de sortie (envoyer) a également été remodelée. Dans PE5 la sortie était toujours en ASCII, ce qui pouvait porter à confusion en envoyant des nombres, par exemple est-ce que 23 est le nombre 23 ou les deux caractères ASCII « 2 » + « 3 » ?

Dans PE6 un nouveau « mode brut » permet à la boîte de dialogue de sortie d'être formatée exactement comme les commandes serout et serin de PICAXE, pour que les données brutes et séries puissent être plus simplement mélangées et envoyées ensemble. Par exemple :

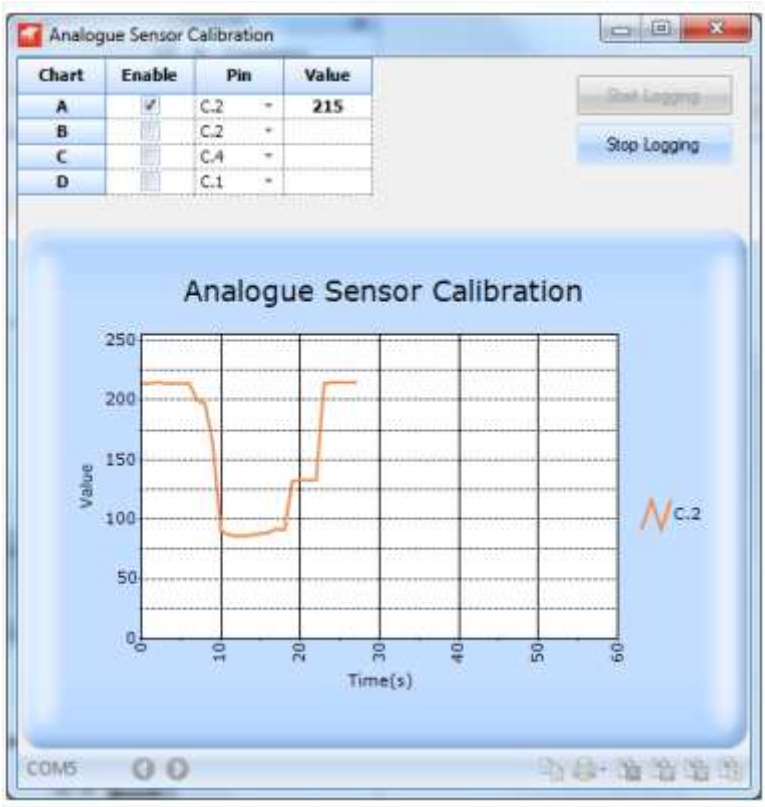
« hello », 13, 10

Notez que pour envoyer la série ASCII dans PE6 (ce qui veut dire pour se rapporter à l'ancien comportement de PE5), sélectionnez le mode « ASCII » à la place (ou utilisez simplement le mode brut avec le texte entouré de double guillemets).

Les entrées séries reçues peuvent être sauvegardées, imprimées, ou exportées en PDF.

Notez que les simulations à l'écran utilisent maintenant exactement la même fenêtre de Terminal, pour que le comportement des séries « simulées » soient plus proches de celles des séries « en vrai ».

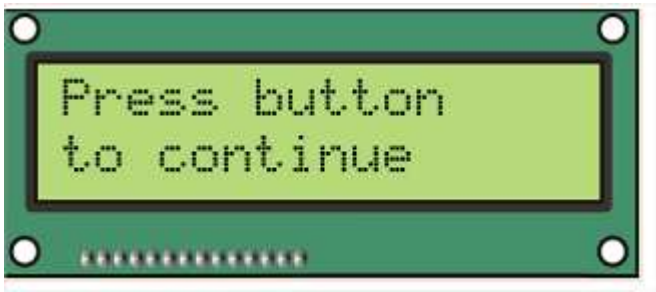
Calibrage de la sonde analogique



La fonction de calibration de la sonde a également été grandement améliorée et peut afficher un graphique en temps réel des valeurs analogiques tout en expérimentait avec la calibration. Le résultat peut être exporté en graphs, PDF ou feuille Excel.

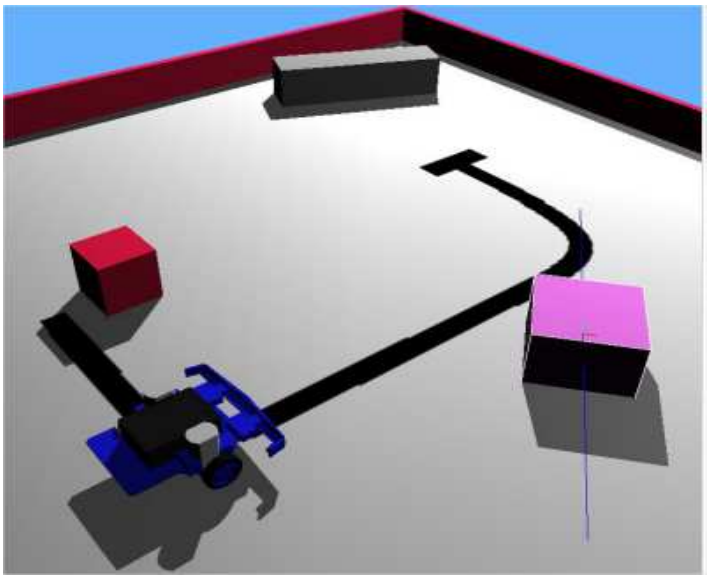
Section 3 : Moteur de Simulation

Le moteur de simulation à l'écran a été re-conçu pour le rendre plus efficace et pour qu'il soit plus proche des « vrais » appareils.



Les nouvelles fonctions-clés incluent :

- Un « diagramme » de simulation configurable pour différents schémas d'entrée/sortie
- La simulation de plusieurs fichiers de programmes (quand utilisé avec #INCLUDE)
- Un meilleur support pour de multiples entrée analogiques et de multiples entrées générales (pulsin, count, etc)
- Un panneau de contrôle des variables amélioré et des points de rupture conditionnés.
- La simulation en série utilise un terminal de série identique à un véritable appareil
- Amélioration de la simulation du son et des tonalités
- Simulations de LCD/OLED et contrôleurs infra rouges plus réalistes.
- « fast run » non-graphique pour mieux imiter un vrai appareil
- Support pour les puces PICAXE connectées, pour que les vraies entrées/sorties soient activées via une simulation à l'écran.
- Simulation de liens en temps réel à un circuit tiers et logiciel de modélisation en 3D, par exemple Webots pour une simulation en 3D d'un robot PICAXE mobile
- Les organigrammes utilisent exactement le même moteur de simulation que les fichiers BASIC.



Maquette Webots en 3D de BOT120

Section 4 : Le Pre-processeur PICAXE

Les buts principaux du nouveau pre-processeur PICAXE sont :

1)De permettre à des programmes BASIC d'être divisés en plusieurs fichiers
2)De permettre des macros (fonctions définies par l'utilisateur)
3)De permettre le remplacement de plusieurs séries de mots, par exemple myhighC(c) = high port C x

Le pre-processeur prend un ou plusieurs fichiers d'entrées, les combine, substitue et traite les macros, puis génère un seul fichier de sortie qui est ensuite passé au compilateur PCAXE standard. Au besoin les utilisateurs finaux peuvent voir le code de sortie du pre-processeur via file>Options>Diagnostics tab

Directives du pre-processeur

Les directives ont un symbole dièse/pound (#) comme premier caractère non-vide d'une ligne, et sont immédiatement suivis par le mot-clé directeur.

Il y a deux principaux types de directives, ceux utilisés par le pre-processeur et ceux utilisés par le compilateur. Le pre-processeur ignore simplement les directives du compilateur.

La liste ci-dessous est celle de toutes les directives de pre-processeur

#PICAXE	Définit le type de PICAXE. Il doit correspondre aux réglages de l'explorateur de Workspace
#INCLUDE	Inclut et fusionne un fichier spécifique, qui a en général une extension de fichier .basinc
#DEFINE	Définit le symbole d'une seule ligne
#MACRO / #ENDMACRO :	Définit un symbole ou une macro multiligne.
#UNDEF	Retire la définition d'un symbole ou d'une macro précédemment définie.
#IF	Traite le bloc suivant si le symbole / le calcul est vrai
#IFDEF	Traite le bloc suivant si le symbole est défini
#IFNDEF	Traite le bloc suivant si le symbole n'est pas défini
#ELSEIF	Traite le bloc suivant si le symbole / le calcul est vrai
#ELSEIFDEF	Traite le bloc suivant si le symbole est défini
#ELSE	Traite le bloc suivant si le bloc précédent n'a pas été traité
#ENDIF	Termine un bloc de condition
#REM	Commence le commentaire d'une section de code
#ENDREM	Finit le commentaire d'une section de code
#ERROR	Force une erreur à la position actuelle

Directives de compilateur

La liste suivante est celle de directives de compilateur. Ces directives sont simplement ignorées par le pre-processeur.

Directives de compilateur

#NO_DEBUG Ne met aucune commande de débuggage en sortie dans le programme.

#NO_END N'ajoute pas la commande END automatique à la fin du fichier

#TITLE Met un titre pour l'outil de téléchargement à distance

#REVISION Met un numéro de version

Directives de compilateur / module de téléchargement

#SLOT Définit le slot cible du programme

#COM Sélectionne le port COM pour le téléchargement (passe au dessus de l'explorateur de workspace)

#NO_DATA Ne télécharge pas les données EEPROM

#NO_TABLE Ne télécharge pas les données de TABLE

#TERMINAL Ouvre le terminal à un taux de bauds spécifié.

#FREQ Règle la freq (fréquence?) pour les anciennes parties obsolètes

Directives de présentation

#REGION / #ENDREGION Défini le déroulement d'un bloc de région

Devenus obsolètes dans PE6

#GOSUBS - Sélectionnez à présent le compilateur spécial via File>Options>Compilateurs

#SIMTASK - Sélectionnez à présent Simulation dans l'Explorateur de Workspace

#SIMSPEED - Maintenant, utilisez simplement le curseur de vitesse de simulation en bas à droit de la barre de statut.

Substitutions aux séries variables du pre-processeur

Les séries de pre-processeur prédéfinies suivantes se substituent comme suit durant le pré-traitement. L'heure et la date sont basée sur l'horloge du système.

ppp_date « YYYY-MM-DD »
ppp_date_uk « DD-MM-YYYY »
ppp_date_us « MM-DD-YYYY »
ppp_datetime « YYYY-MM-DD HH:MM:SS »
ppp_time « HH:MM:SS »
ppp_filename Le nom de fichier raccourci du fichier principal d'entrée BASIC
ppp_filepath Le nom de fichier en entier et le cheminement du fichier principal d'entrée BASIC

Exemple d'utilisation :

sertxd (« Filename : », ppp_filename)
sertxd (« Downloaded : », ppp_datetime)

#INCLUDE « filename.basinc »

La directive #INCLUDE ouvre et traite un fichier, puis continue à traiter le fichier original. Un fichier ouvert via la directive #INCLUDE peut lui-même contenir d'autres directives #INCLUDE. Il n'y a aucune limite aux nombre de fichiers qui peuvent être imbriqués.

Le nom de fichier peut être inclus entre de doubles guillemets et peut utiliser un cheminement absolu ou relatif. Si le cheminement est relatif, le pre-processeur essayera de localiser le fichier dans ces fichiers dans cet ordre :

1) Le dossier du fichier principal
2) Chaque fichier identifié par la variable Windows %Path%Environment

Les fichiers inclus sont normalement nommés avec une extension .basinc, mais peuvent également être des fichiers .bas. Il n'y a pas de différence entre le format de ces fichiers à part leur extension.

Les avantages d'utiliser l'extension .basinc pour inclure des fichiers sont :

1) PE6 désactive la syntaxe Check/Program, les boutons sur ce type de fichier, pour que vous n'essayiez pas de compiler accidentellement un fichier inclus par lui-même.
2) PE6 sauvegarde automatiquement n'importe quel fichier .basinc du programme principal ouvert. Ceci permet d'assurer que tout changement fait à l'écran aux fichiers inclus soient correctement inclus dans la compilation.
3) Les autres utilisateurs de PICAXE reconnaîtront le fichier comme un fichier inclus.

Les fichiers inclus doivent être sauvegardés en ASCII ou UTF-8 Unicode avec une fin de ligne en type CR ou CR-LF (pas de LF seul)

Tandis que le compilateur traite « de haut en bas » sur le fichier de sortie du pre-processeur, il est nécessaire de faire attention à « sauter » le fichier #INCLUDE (si nécessaire) quand il est inclus en haut d'un programme.

```
Reset_here :   Goto Init              ; jump over include file

#INCLUDE    « sphero.basinc »

Init :                                ; start program here
```

#DEFINE and #MACRO

#DEFINE est utilisé pour définir les symboles de directive qui seront ensuite utilisés dans le pré-traitement. A ne pas confondre avec les types de symboles « symbol xxx = yyy » qui sont utilisés par le compilateur dans le programme principal. Le pre-processeur ne peut ni utiliser, ni reconnaître les symboles définis « symbol xxx = »

#MACRO / #ENDMACRO (ou #ENDM) est simplement la version multiligne de #DEFINE

Exemples d'utilisation :

1) La définition simple d'un symbole à utiliser avec #IFDEF / #IFNDEF

```
#DEFINE SPHERO
```

2) Substitution de série

```
#DEFINE MAGIC_NUMBER    « 0006664A3BF8 »
#DEFINE SpheroWakeUp    Gosub Sphero-Initialise
#DEFINE SetBackLedOn    b0 = 255 : Gosub SenBackLED
#DEFINE SetBackLedOff   b0 = 0 : Gosub SendBackLED
```

3) Substitution de valeurs numériques

```
#DEFINE PROGRAM_VERSION     22
#DEFINE PORT_VALUE          %10101010
#DEFINE FALSE                   0
#DEFINE TRUE                1
```

4) Substitution d'une autre macro

```
#DEFINE SetRgbColorOff              SetRgbColor (0, 0, 0)
#DEFINE SetRgbColorRed              SetRgbColor (255, 0, 0)
#DEFINE SetRgbColorGreen        SetRgbColor (0, 255, 0)
#DEFINE SetRgbColorBlue             SetRgbColor (0, 0, 255)
```

5) Passer des paramètres qui seront ensuite utilisés avec la substitution

```
#MACRO  SetRgbColor (R, G, B)
        b0 = R
        b1 = G
        b2 = B
        Gosub SendRGB
#ENDMACRO

#DEFINE SetHeadingSpeed(H)      w0 = H : Gosub SendHeading

#MACRO SetHeadingSPeed(H, S)
        w1 = H
        b0 = S
        Gosub SendHeadingSpeed
#ENDMACRO
```

Les paramètres peuvent être définis en utilisant n'importe quel mot commençant par une lettre et qui n'est pas déjà un mot-clé BASIC.

Traitement conditionnel.

Le traitement peut être conditionné par l'utilisation des directives #IFDED / #IFNDEF / #IF

```
#IFDEF symbol   Traite si le symbole a été défini.
#IFNDEF symbol          Traite si le symbole n 'a pas été défini
```

#IF symbol	Traite si le symbole a une valeur vraie (pas 0) ou numérique
#IF symbol>constant	Traite si le calcul de la constante numérique est vraie. Les opérateurs supportées sont =, <>,>, >=, <,<=

Plusieurs tests peuvent être utilisés en utilisant les directives #ELSEIF ou #ELSEIFDEF, et un seul #ELSE optionnel peut aussi être utilisé avec le #ENDIF, par exemple :

#IF symbol > 10

#ELSEIF symbol > 5

#ELSE

#ENDIF

ou en utilisant des symboles définis.

#IFDEF symbol1

#ELSEIFDEF symbol2

#ELSEIFNDEF symbol3

#ELSE

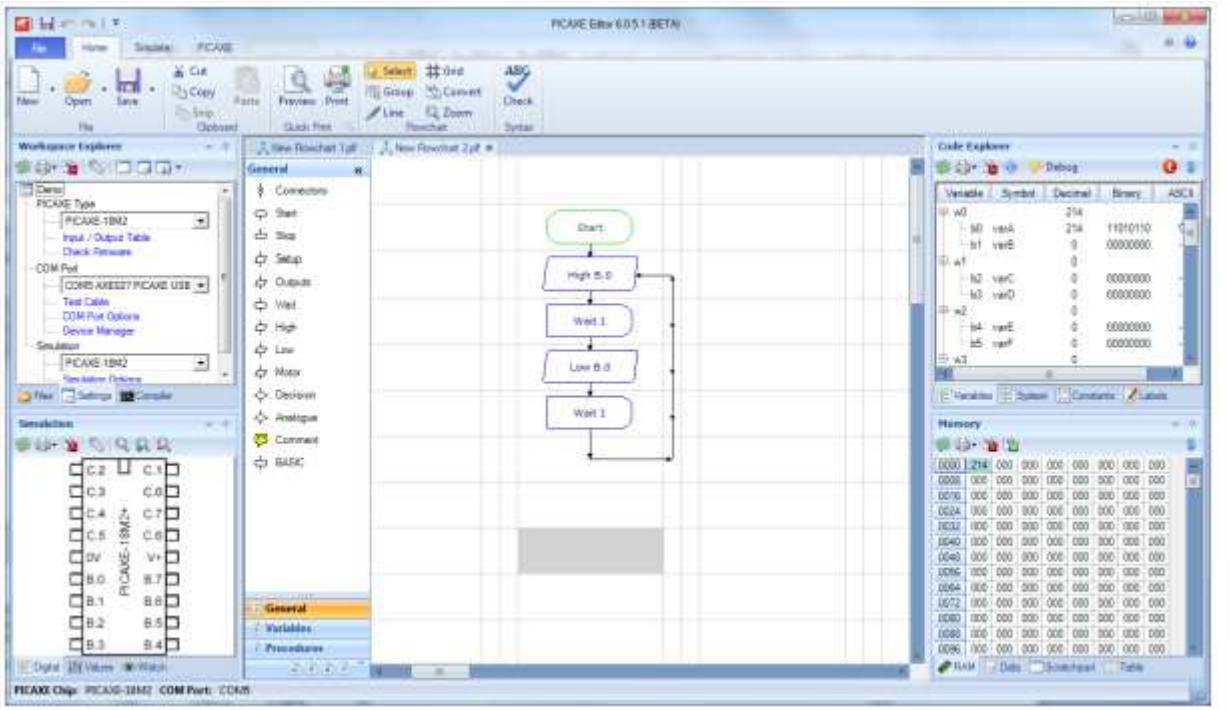
#ENDIF

Erreurs forcées

#ERROR « Ce code est incomplet ! »

#ERROR force le pré-traitement à s'arrêter et sort le message d'erreur

Section 6 : Organigrammes dans PE6



PE5 avait un module d'organigramme obsolète qui a été remplacé pendant plusieurs années par le populaire logiciel éducatif populaire d'organigrammes « Logicator for PIC micros ». Cependant, comme Logicator était auparavant une application complètement séparée, elle requérait aussi une maintenance et un développement séparé chronophage.

Donc, Logicator a été complètement fusionné dans PE6, pour que l'application seule puisse utiliser les deux méthodes de développement de code PICAXE (programmation en BASIC et organigramme). Il n'y a pas de supplément à payer pour PE6 donc les programmations en organigramme et en BASIC sont complètement gratuits à utiliser.

Cette amélioration permet aussi l'utilisation de toutes les fonctionnalités de PICAXE, en particulier les pattes i/o configurables sur les parties M2 et X2, des variables supplémentaires et beaucoup plus de canaux d'entrée ADC sont maintenant disponibles. L'utilisateur peut maintenant utiliser la nouvelle cellule « réglages » de l'organigramme pour définir quelles pattes sont des entrées et lesquelles sont des sorties sur toutes les parties applicables du PICAXE.

La boite à outil de la cellule « organigramme » est à présent complètement configurable, les utilisateurs peuvent donc montrer ou cacher les différentes cellules de commande selon l'age ou l'expérience des étudiants utilisant l’application. Les utilisateurs peuvent également créer, s'ils le désirent, leurs cellules d'organigrammes propres et entièrement neuves (par exemple pour un type particulier de sonde) puis les ajouter à la boîte à outils. Cette nouvelle cellule sera également entièrement simulée.

Enfin, l'interface utilisateur a été complètement rafraîchie avec des lignes plus faciles à dessiner, à effacer et des fonctions de sélection de groupe. Les utilisateurs peuvent également sélectionner le schéma de couleurs utilisé avec l'organigramme.

Sommaire des nouvelles fonctions de l'organigramme :

- Configure les pattes d'entrée et de sortie sur tous les types de PICAXE supportés
- Moteur de simulation plus précis qui supporte toutes les cellules de commande, y compris les cellules génériques de basic
- Dessin et suppression de ligne amélioré
- Sélection de groupe amélioré pour couper et coller
- Plusieurs organigrammes peuvent être ouverts simultanément
- Terminal de série amélioré, débuggage et réglage des sondes de calibrage analogique
- Supporte jusqu'à 8 tâches en parallèle
- Nouveaux points de rupture et fonctions de surveillance des variables
- Schéma de couleur sélectionnable par l'utilisateur
- Cellule de boite à outil configurables par l'utilisateur (cache/affiche différentes commandes de cellule)
- L'utilisateur peut ajouter de toutes nouvelles cellules de commande « définies par l'utilisateur »
- Toutes les cellules de commande sont traduites pour toutes les langues
- Les simulations sont liées à des circuits tiers et modèles en 3D