

Lb8 - Environnement de programmation pour le BimoKitLed

Introduction

Le Bimo peut embarquer des diodes, directement soudées sur le circuit imprimé, ou reliées par des longs fils. Voir <http://www.bricobot.ch/kits/BimoKitLeds.pdf>

Cette documentation est transposée de celle du Coeur. On pourrait imaginer de combiner la documentation Abimo et Lb8 pour avoir des effets lumineux en fonction des déplacement. Cela peut se faire en assembleur ou dans un autre langage.

L'utilisation de l'environnement de programmation est expliqué dans

<http://www.bricobot.ch/docs/InstallPic.pdf>

Pour celui qui est familiarisé, il faut copier le CD dans un dossier appelé par exemple Coeur, et établir des raccourcis à SmileNG.exe dans le sous-dossier SmileNG, et à Pickit2.exe dans le sous-dossier Pickit2.

Les fichiers se trouvent dans le dossier Lb8.

Ne pas modifier les fichiers avec l'extension .asi.

Pour programmer, le plus simple est d'avoir ajouté le connecteur de programmation sur le module BimoKitLed .

Un mauvais contact ou une inversion peut mettre le processeur dans un mode qui nécessite ensuite une procédure de récupération (voir

<http://www.bricobot.ch/docs/RecupPic.pdf>)



Instructions

Commençons par quelques définitions, qu'il faudra relire après avoir vu les exemples. En première lecture, c'est un peu abstrait.

Les instructions qui ont été définies permettent d'écrire et modifier des programmes qui agissent sur les LEDs. Une instruction définit une action, par exemple faire clignoter toutes les Leds. Mais il faut dire à quelle vitesse on veut qu'elles clignent, et combien de fois. ce sont les paramètres de l'instruction, qui sont ici des constantes.

Une constante doit toujours être précédée du signe # (prononcé valeur). La valeur maximale est 255. On peut modifier la valeur d'une constante, et c'est une partie du jeu que l'on va faire, mais il faut chaque fois re-traduire le programme, recharger les nouvelles instructions dans le processeur, et vérifier l'effet en exécutant le programme.

Les instructions sont alignées dans la mémoire programme. On peut donner un nom à une position mémoire programme (une étiquette, comme pour marquer des casiers superposés). On doit le faire si on veut sauter des instructions pour continuer à un endroit précis. Une étiquette est suivie d'un : (deux-points). Elle doit avoir un nom nouveau qui, comme pour tous les noms que vous devrez inventer, ne commence pas par un chiffre (et n'est pas déjà utilisé par le programme de traduction).

Structure générale des programmes

Pour que nos instructions s'exécutent correctement, il faut dire plusieurs choses au programme de traduction, dans un certain ordre. Des instructions sont insérées par l'ordre .Ins Les signes \ et ; caractérisent le début d'un commentaire. La fin de la ligne est ignorée. On peut mettre des lignes vides et, pour aligner, on utilise de préférence la touche « tab » (à gauche, marquée par une flèche).

```
\prog :NomDuFichier.asm et brève explication
.Ins LbDef.asi
.Ins LbInit.asi
; éventuellement mettre ici le nom du fichier, codé
.Ins LbProg.asi
; mettre le programme ici
; le programme doit se terminer par un AllerA, autrement le processeur
; va exécuter des mots binaires dans la mémoire qui feront on ne sait quoi !
.Ins LbFin.asi
.End
```

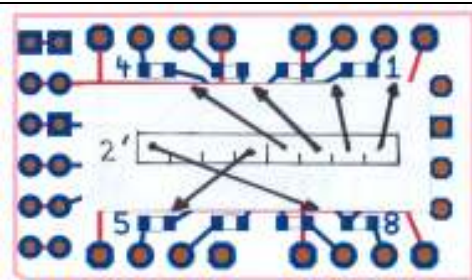
Exemple : programme demo du 21.8.08

<pre>\prog:Lh1.asm demo .Ins LbDef.asi .Ins LbInit.asi .16 "L","b","1"," "," " (suite colonne droite)</pre>	<pre>.Ins LbProg.asi Boucle : Motif #2'11001001 Allume #20 Motif #0 Allume #5 Motif #2'11111111 Clignote #4,#6 AllerA BouCle .Ins LbFin.asi .End</pre>
-------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------

Pour assembler et programmer, relire la 2^e partie de InstallPic.pdf
<http://www.bricobot.ch/docs/InstallPic.pdf>

Instructions spécifiques

Avec l'instruction **Motif**, on prépare ce qui sera affiché. Il faut décrire les 8 bits dans l'ordre, et noter deux nombres binaires que le processeur comprendra et enverra dans l'électronique qui commande les diodes lumineuses.
 Le processeur ne connaît que les mots binaires de 8 bits. Il faut donc préparer deux mots consécutifs pour commander les 16 diodes.
 La notation #2' devant un nombre (formé de 0 et de 1) dit au programme que c'est une constante écrite en binaire. On peut se simplifier la vie parfois et écrire 0 au lieu de 2'00000000.



<p>Allume #Duree Exemple : Motif #2'11001001 Allume #20 ; env 2 sec</p>	<p>Allume le motif pendant la durée</p>	
<p>Clignote #Duree,#NombreDeFois Exemple : Motif #2'11110000 Clignote #1,#10</p>	<p>Allume le motif pendant la durée, éteint pendant la même durée, répète NombreDeFois</p>	

Question :

Comment remplacer **Clignote #2,#4** en utilisant uniquement l'instruction **Allume**?
Pour créer un nouveau programme, le plus simple est de reprendre un programme existant et de tout de suite le renommer (onglet Fichier – Enregistrer sous)

Varie #DuréeEvolution, #NbdeFois Exemple Varie #1,#20 ; Pulse 20 fois le plus rapidement possible ; (env 4 battement/seconde)	Augmente et diminue l'intensité du motif préparé, NbdeFois, selon DuréeEvolution
-----------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------

Faites un programme avec uniquement cette instruction. Quelle valeur pour un battement proche de votre cœur ?

Avec une valeur de 1, cela ressemble avec un Allume-tout Allume-rien rapide. Vous essayez de comparer ? Si vous n'êtes pas encore à l'aise avec l'écriture des programmes, chargez le programme Lh3.asm.

Avec ce programme, la période est la même, mais on peut voir une différence : si on agite le cœur rapidement dans l'obscurité : l'instruction **Varie** pulse pour faire une variation progressive, ce que ne fait pas **Allume**.

Les instructions suivantes n'utilisent pas le Motif.

TourneG Durée,NombreDeFois Allume une led après l'autre en tournant	Tou: Repete #4 TourneG #4,#1
TourneD Durée,NombreDeFois Idem dans l'autre sens	Attend #2 TourneG #4,#1 Attend #2 FinRepete

Chenillard nbBits,durée,nbde fois Allume nbBits Leds une après l'autre et les fait circuler jusqu'à disparition.	Chenillard #20,#1,#1 Remplit en 16 impulsions, attend 4 impulsions et vide.
----------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------

Instruction de structures, identiques à Abimo

Attente DuréeEnDixièmesdeSec Ex : Attente #10 ; pour une seconde	Les attentes ne sont pas calibrées précisément.
----------------------------------------------------------------------------	-------------------------------------------------

AllerA Adresse Ex : Boucle : ; Des instructions que l'on veut répéter définiment AllerA Boucle	C'est le Goto du Basic, le Jump d'autres langages Une étiquette
----------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------

Stop A mettre à la fin d'in programme si on ne veut pas rester dans une boucle	C'est l'équivalent de Ici : AllerA Ici
------------------------------------------------------------------------------------------	-------------------------------------------

Etiquette : Repete Nombre de fois ; instructions répétées FinRepete Etiquette	
----------------------------------------------------------------------------------------------------------	--

A noter qu'un paramètre égal à zéro est interprété par le processeur comme identique à 256. A noter aussi que chaque instruction a des paramètres par défaut. Si on écrit Motif tout seul, le programme voit Motif #2'11111111

A vous de jouer. Bien du plaisir à inventer vos programmes.