

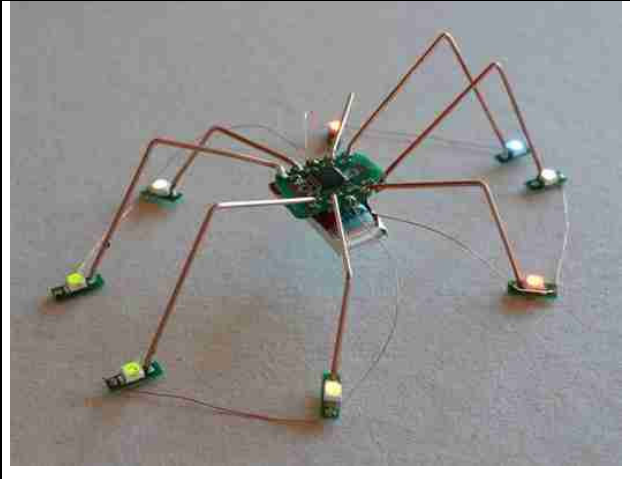
Vega - Environnement de programmation pour le circuit Vega ou Deneb

Introduction

L'utilisation de l'environnement de programmation est expliqué dans <http://www.bricobot.ch/docs/Abimo.pdf>
La doc sur le Cœur, BimokitLeds, Orion est aussi un bon moyen pour s'initier.
Voir

<http://www.bricobot.ch/docs/Smile.html>
On lira les pages 1 et 2 de ce document avant de continuer ici.

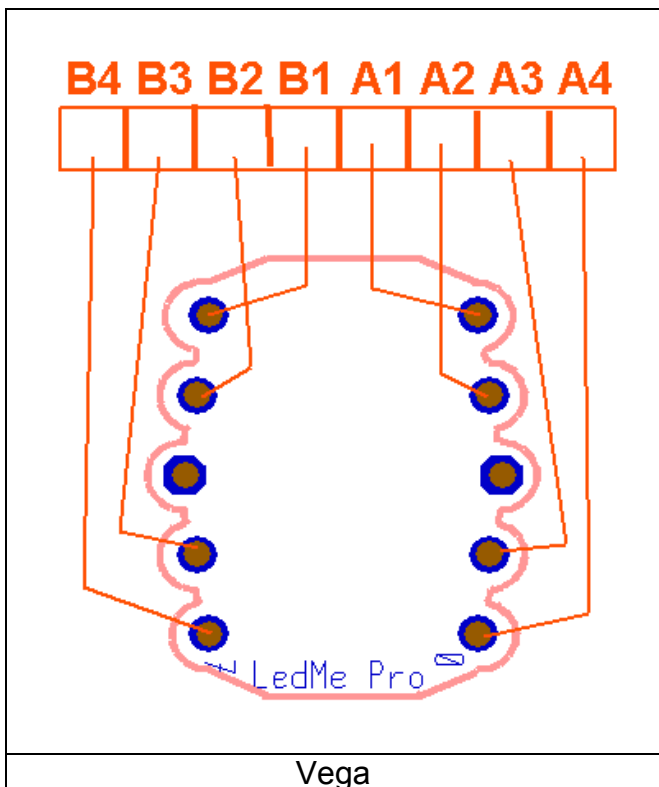
Le principe de programmation est le même que pour Orion et Coeur.



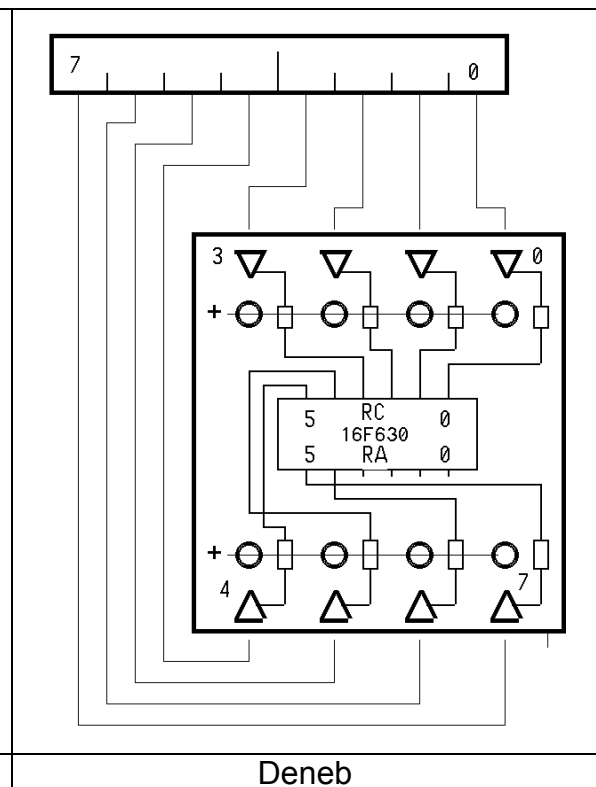
Le source des programmes Vega/Deneb se trouve sous

www.bricobot.ch/docs/VegaProg.zip

Ces programmes s'exécutent dans l'environnement SmileNG avec un Pickkit2 pour programmer le microcontrôleur 16F630 du Deneb.



Vega



Deneb

Position des bits par rapport aux octets des instructions

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 instructions définit une action, par exemple 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 soit des constantes, soit des variables.

Dans les programmes du début, on utilise surtout des constantes ; on corrige leur valeur dans le programme, on recharge les nouvelles instructions dans le processeur, et on vérifie l'effet.

Une constante doit toujours être précédée du signe # (prononcé valeur). C'est un nombre, auquel on peut donner un nom (c'est souvent préférable) en utilisant le signe = (on parle d'assignation).

Les variables sont dans une zone mémoire ou il faut réserver sa place ; elles nécessitent plus de réflexion, on y reviendra dans la 2^e partie.

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).

Instruction de structures, identiques à Orion et Coeur4

	Exemples
AllerA Adresse C'est le Goto du Basic, le Jump d'autres langages	Boucle : ; <i>Des instructions que l'on veut répéter indéfiniment</i>
SiZeroAllerA Variable, Adresse Si la variable est à zéro, on saute à Adresse, autrement on continue.	AllerA Boucle S'utilise pour sortir d'une boucle ou on décale ou diminue une variable.
Stop Un programme se termine toujours par un AllerA ou par un Stop. Autrement, le processeur va exécuter des bits en mémoire, et on ne sait pas ce qu'il va faire	A noter que les instructions et les paramètres sont décalés et alignés pour faciliter la lecture. Minuscules et majuscules sont identifiées. Un commentaire est précédé d'un ; ou d'un \
Copie Source, Destination On prend la valeur de la source et on la met dans la variable destination (nécessairement une variable) La copie ne modifie pas la source.	Copie #7, Décompteur Copie la valeur 7 dans la variable Décompteur (on dit initialise cette variable)
Augmente Variable Diminue Variable DécaleADroite Variable DecaleAGauche Variable	Copie NbdeFois, Décompteur Copie une variable dans une autre Les variables doivent être déclarées Exemples dans la 2 ^e partie
Etiq : Repete Nombre de fois ; instructions répétées FinRepete Etiq	
Attente DuréeEnDixièmesdeSec	Attente #10 ; pour une seconde Attente VarDelai ; attente selon le contenu de la variable VarDelai. Si on a écrit avant Copie #10, VarDelai on a le même effet que la première instruction

Instructions spécifiques

Clignote Duree, NombreDeFois
Allume tout selon la durée, éteint pendant la même durée

TourneG Durée, NombreDeFois
Allume une led après l'autre en tournant

TourneD Durée, NombreDeFois
Idem dans l'autre sens

ToutAllume Duree
ToutEteint Duree

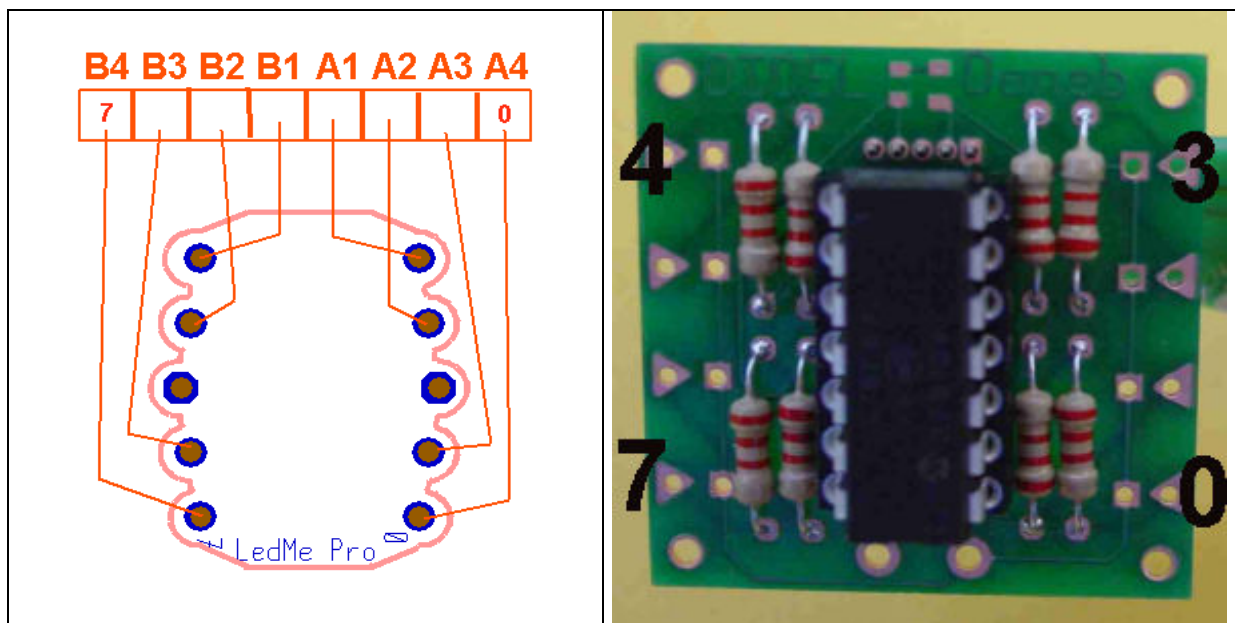
Chenillard nbBits, durée, nbde fois
Allume nbits l'un après l'autre et les fait tourner jusqu'à disparition

Cligno #1,#10
Clignote rapidement 10 fois

Tou: Repete #4
TourneG #4,#1
Attend #2
TourneG #4,#1
Attend #2
FinRepete

Comment remplacer Cligno #2,#4
en utilisant ces deux instructions ?

Chenillard #4,#5,#2



Motif Constante ou variable 8 bits

Motif #2'00111100

Prépare pour allumer ou varier B2 B1 A1 A2

Allume Duree
Allume le motif préparé pendant la Durée

Motif #2'00111100
Allume #6

Varie DureeEvolution, NbdeFois
Augmente et diminue l'intensité du motif préparé NbdeFois, selon DuréeEvolution

Motif #2'10000001
Varie #3,#5
Varie les LEDs 7 et 0 5 fois de suite

AdresseTable: **DebutTable**

Une table liste des valeurs succesives que vont prendre les LEDs. Il faut commencer par une étiquette et l'instruction DebutTable..

Les tables sont toutes (avec des étiquettes différentes) entre VegaDef et VegaProg

Leds suite de huit 0 et 1 correspondant aux LEDs

TaRa : DebutTable

Leds 01100110

Leds 00100100

Leds 00011000

Leds 00011110

Leds 01111000

AppelleTable

Adresstable,LongueurDeLaTable,Durée,NombreRepet

AppelleTable Tara,#5#3,#2

Répète 2 fois le balayage de la table Tara avec la vitesse 3

Structure générale des programmes

\prog :Nom.asm et brève explication

.Ins VegaDef.asi

; mettre les variables ici

.Ins Vegalnit.asi

; mettre les tables ici:

 ; mettre les tables ici:

.Ins VegaProg.asi

 ;mettre le programme ici

Nom "N o m"

.End

Exemple : programme demo du 21.8.08

\prog:VegaPap.asm demo du 21.8.08

.Ins VegaDef.asi

; mettre les variables ici

.Ins Vegalnit.asi

; mettre les tables ici:

; mettre les tables ici:

TaPapi:

 DebutTable

 Leds 00011000

 Leds 01000010

 Leds 00010000

 Leds 01000100

 Leds 00001000

 Leds 10000010

 Leds 00010000

 Leds 00100010

 Leds 0

(suite colonne droite)

.Ins VegaProg.asi

Boucle:

 ClignoPaire1 #1,#2

 Varie #2'00011000,#3,#6

 ClignoPaire1 #1,#2

 Descend #3,#1

 Monte #3,#1

 ToutEteint #3

 Allume A1,#2

 Allume B1,#2

 Cligno #3,#5

 ToutAllume #5

 ToutEteint #5

 Varie 11111111,#8,#1

 AppelleTable TaPapi,#1,#4

 AllerA Boucle

Nom "N o m"

Utilisation des variables

Les variables permettent de faire des boucles dans lesquelles des paramètres changent.

L'ordre Cligno par exemple a un paramètre qui est la durée du clignotement. Cette durée peut être fixe, ou variable

Si on écrit Cligno #2, la durée est constante, environ 0.2 secondes

Si on écrit Cligno Dur1, on ne peut pas dire quelle est la durée en lisant

l'instruction. La valeur de la durée est dans la position mémoire Dur1, ce sont les instructions précédentes qui ont préparé la valeur dans Dur1.

Si on écrit

 Copie #2,Dur1

Cligno Dur1
alors on connaît la durée du clignotement.
L'avantage des variable, c'est qu'elles peuvent varier! Ecrivons

```
Copie #2,Dur1
Rr1: Repete #20
      Cligno Dur1
      Diminue Dur1
      FinRepete Rr1
```

Le clignotement va se répéter 20 fois avec une durée qui augmente.
Ceci permet des effets plus spectaculaires en écrivant moins d'instructions.
Les variables doivent être déclarées, pour que le traducteur leur réserve de la place en mémoire. On doit donc inventer un nom, aussi explicite que possible.
A l'emplacement prévu pour les variables, on écrira

```
Var Dur1
```

Pour faire un chenillard qui va dans l'autre sens, on décale une variable qui contient le motif dans DATA (variable déjà réservée, pas besoin de la déclarer).

Motif #2'11000000 (copie dans Data)	(suite)
Deca: DecaleADroite Data	AllerA Deca
Allume #4	
SiZeroAllerA Data,Fin	Fin: ...

L'art d'un programme bien écrit, que l'on relit facilement après quelques mois, tient beaucoup dans le choix des noms donnés aux variables et aux étiquettes. Dans les long programme, un commentaire doit annoncer un groupe d'instruction.

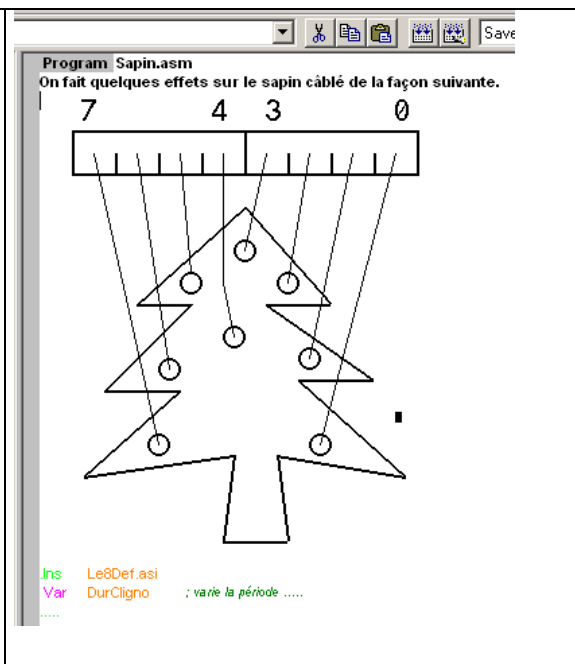
A noter que dans les programmes SmileNG, on peut insérer des images. Le fichier doit être un .bmp et on écrit au début d'une ligne

\image:Mondessin.bmp (pas d'espaces autour du :)

La touche F8 fait apparaître ce texte ou le dessin

Exemple

```
\prog:VegaSapin.asm
;On fait quelques effets sur le sapin câblé de la façon suivante.
\image:Sapin.bmp
.Ins VegaDef.asi
Var DurCligno ; varie la période
.Ins Vegalnit.asi
; mettre les tables ici ; pas de tables
.Ins VegaProg.asi
Boucle:
.....
AllerA Boucle
Nom "S a p i n"
```



Résumé des ordres

AllerA Adresse	Cligno Duree,NombreDeFois
SiZeroAllerA Variable, Adresse	TourneG Durée,NombreDeFois
Stop	

Copie Source, Destination Augmente Destination Diminue Destination DécaleADroite Destination DecaleAGauche Destination Etiquette : Repete Nombre de fois ; instructions répétées FinRepete Etiquette Attente DuréeEnDixièmesdeSec	TourneD Durée, NombreDeFois ToutAllume Duree ToutEteint Duree Motif Constante ou variable 8 bits Allume Duree Varie DureeEvolution, NbdeFois
AdresseTable: DebutTable Leds suite de huit 0 et 1 AppelleTable Adresstable, LongueurDeLaTable, Durée, NombreRepet	