



Constructeur de Robots
Pour les Loisirs Innovants et Pour l'Enseignement



Manuel Utilisateur du POB-Bot



Sommaire

1 PRESENTATION.....	5
2 MODULE POB-EYE	7
2.1 Description matérielle de la carte POB-EYE	8
2.2 Alimentation	9
2.3 Programmation et exécution	10
2.4 Port série (UART).....	11
2.5 Connecteur HE10.....	12
2.6 Le bus POB-EYE.....	14
2.7 Fonctionnement du bus POB-EYE depuis une carte d'extension	16
2.8 Connecteur capteur CMOS	22
2.9 Dimension du module.....	22
2.10 Connecter POB-EYE à votre ordinateur.....	23
3 POB-PROTO	25
3.1 Description des différents éléments de la carte.....	26
3.2 Exemple de configuration des ports :	36
3.3 Exemples de montages autour du POB-PROTO :	37
3.4 Informations mécaniques :	40
4 POB-LCD	42
4.1 Description matérielle.....	42
4.2 Connecter POB-LCD128 avec POB-EYE	43



Constructeur de Robots
Pour les Loisirs Innovants et Pour l'Enseignement

Manuel Utilisateur du POB-Bot

4.3	Dessiner sur le POB-LCD128	44
5	AIDE SUR LES FONCTIONS DE LA LIBRAIRIE LIBPOB.....	46
6	EXEMPLES D'APPLICATIONS	48



Constructeur de Robots
Pour les Loisirs Innovants et Pour l'Enseignement

Manuel Utilisateur du POB-Bot

Gestion du document

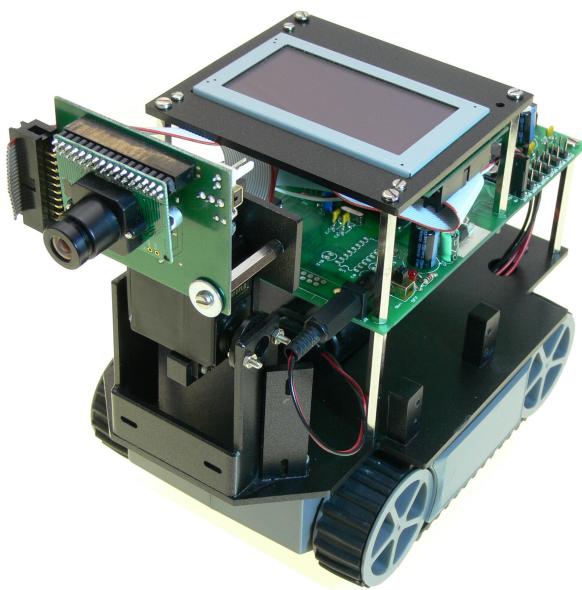
Nom de fichier	Manuel_PobBot_FR.doc
Date de création	10/10/2005
Auteur	Pierre Séguin
Modification	2.0 01.10.2008 chapitre PobTools retiré.

Contacter POB-Technology

Adresse	POB-TECHNOLOGY 11, avenue Albert Einstein 69 100 VILLEURBANNE, FRANCE
Adresse mail	contact@pob-technology.com
Téléphone	+33 (0)4 72 43 02 36
Fax	+33 (0)4 78 58 04 92

1 Présentation

Le **POB-Bot** est un robot mobile programmable. Il a été conçu pour vous permettre de construire votre propre robot



Le **POB-Bot** est composé d'un **POB-Eye**, la camera couleur intelligente ; d'une **POB-Proto**, la carte de commande des entrées et sorties ; et d'un **POB-LCD128**, l'afficheur graphique.

Le **POB-Eye** est le cœur de votre robot ! Programmable en C, Java, Basic en utilisant le **POB-Tools** approprié, il est aussi programmable avec **RISBEE**, le logiciel graphique POB de programmation du robot, accessible aux débutants.

Des bibliothèques de fonctions sont disponibles pour appréhender nativement la reconnaissance de formes, les déplacements, les servomoteurs sans effort de programmation.

Sur le bus Pob sont branchés les autres cartes robotiques, cependant un Bus I2C est disponible pour connecter de nouveaux composants.

La **POB-Proto**, est pilotée par le **POB-Eye**. Par programmation, vous agissez sur les moteurs, les servomoteurs. Vous lisez les valeurs de vos capteurs numériques et analogiques et vous gérez le joystick pour votre interface graphique.

Le **POB-LC128** est l'afficheur. Il vous permet de visualiser en temps réel ce qui provient de la camera, ou de développer une interface graphique.

Il est monté sur une base mobile à chenilles actionnées par deux moteurs à courant continu. Différents emplacements sont prévus pour y ajouter votre propre électronique.

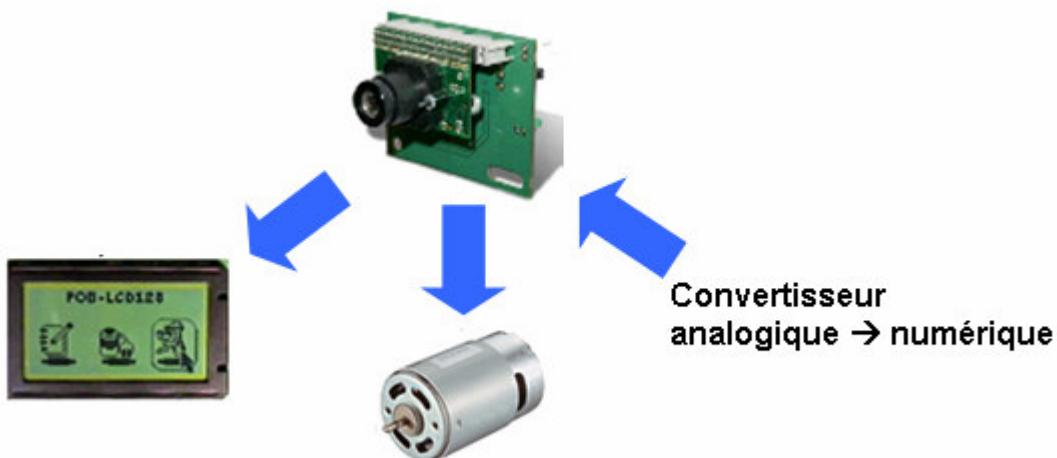


Module Camera Couleur

Programmable avec le Pob-Tools
(en C, Java, Basic) et avec RISBEE

2 Module POB-EYE

Le POB-EYE est un module camera permettant de faire de la reconnaissance de forme en temps réel. Grâce à ses 15 I/O, son port série et son bus I2C, POB-EYE est destiné à devenir le cœur de votre application. Il est livré avec une suite d'outils dédiés, permettant un développement rapide et simple.



Flash	128Koctets
Ram	64Koctets
Bus I2C	Oui
UART	Oui
Entrées/Sorties	15
Dimensions	64.28 x 49.50 mm
Taille de l'image capturée	88 x 120 pixels
Image couleur	Oui (1 octet par composante)
Taille de l'image en mémoire	32Koctets
Tension d'entrée	6V to 9V
Courant d'alimentation	2A maximum
Tension des signaux logiques	3.3V (5 Volt tolérant)
Courant des signaux logiques	10mA maximum

2.1 Description matérielle de la carte POB-EYE

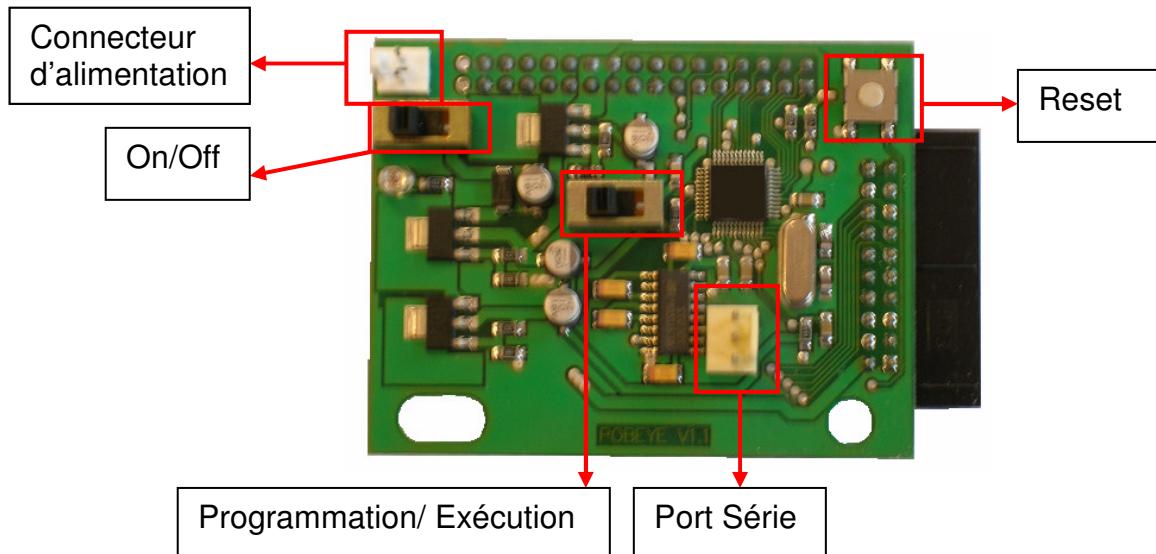


Figure 1 POB-EYE (face alimentation)

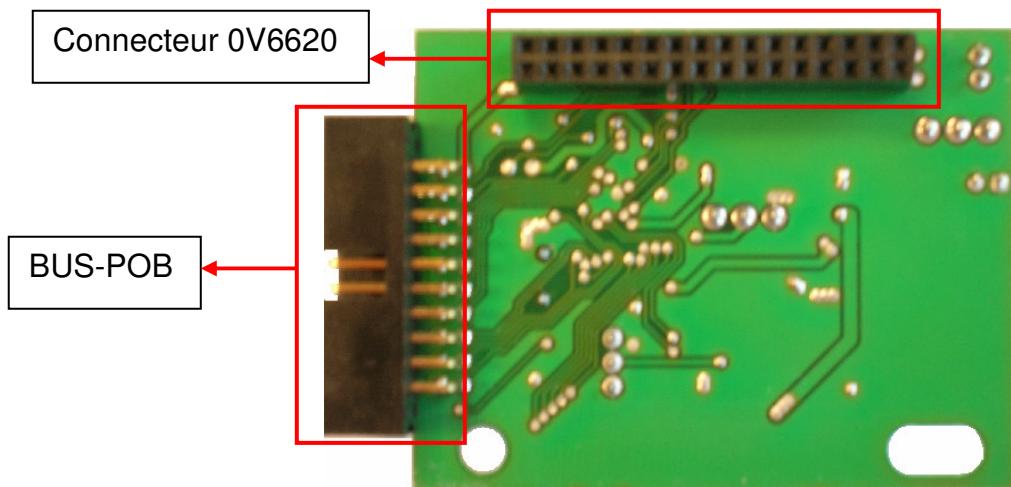


Figure 2 POB-EYE (face connecteur CMOS)

2.2 Alimentation

L'alimentation doit être comprise entre **6V et 12V**. Il existe deux façons d'alimenter le module :

- *Par le connecteur dédié*

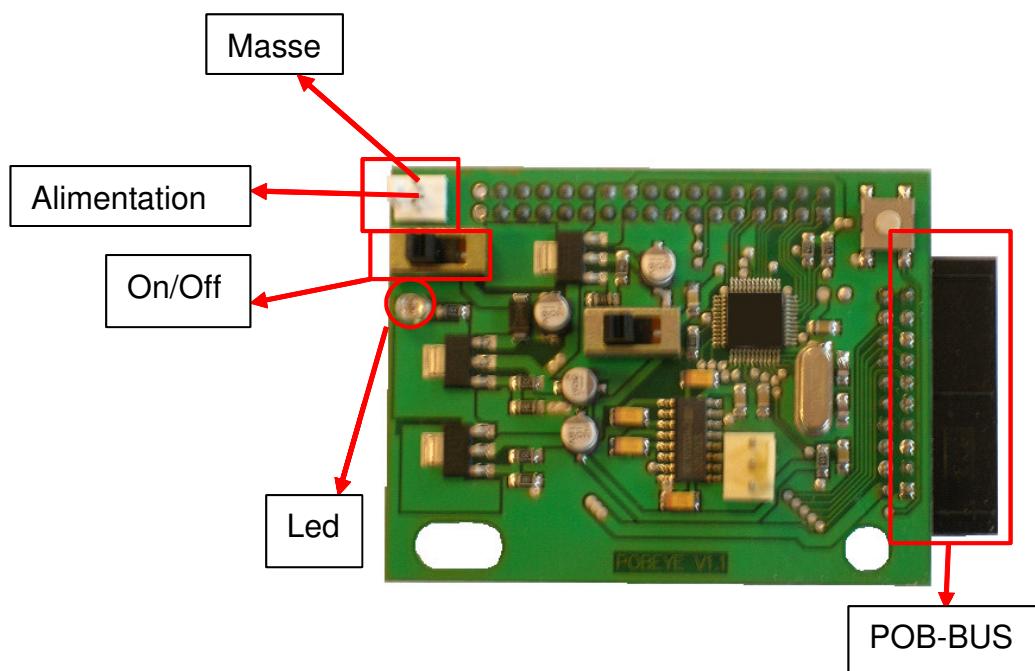


Figure 3 Alimentation du POB-EYE

La mise sous tension se fait par le câble d'alimentation et l'interrupteur « On/Off » à basculer en position « On ».

Le câble d'alimentation est fourni avec le module POB-EYE.

- *Par le connecteur HE10*

En utilisant les pins 1 (+Alimentation) et 20 (Masse GND) du POB-BUS (connecteur HE10), l'utilisateur peut alimenter le module POB-EYE par une carte fille branchée sur le connecteur HE10. L'interrupteur « On/Off » n'ayant aucun rôle dans cette configuration, l'utilisateur peut utiliser l'interrupteur de sa carte fille par exemple. Une fois le module POB-EYE sous tension, la led (témoin de mise sous tension) s'allume.

Attention : Si vous choisissez d'alimenter votre POB-EYE vis le POB-BUS, n'utilisez pas le connecteur d'alimentation du POB-EYE.

2.3 Programmation et exécution

- *Programmation du module*

Pour charger un programme dans la mémoire flash du POB-EYE, il faut positionner l'interrupteur de programmation/exécution en mode « **Programmation** » et mettre sous tension le module.

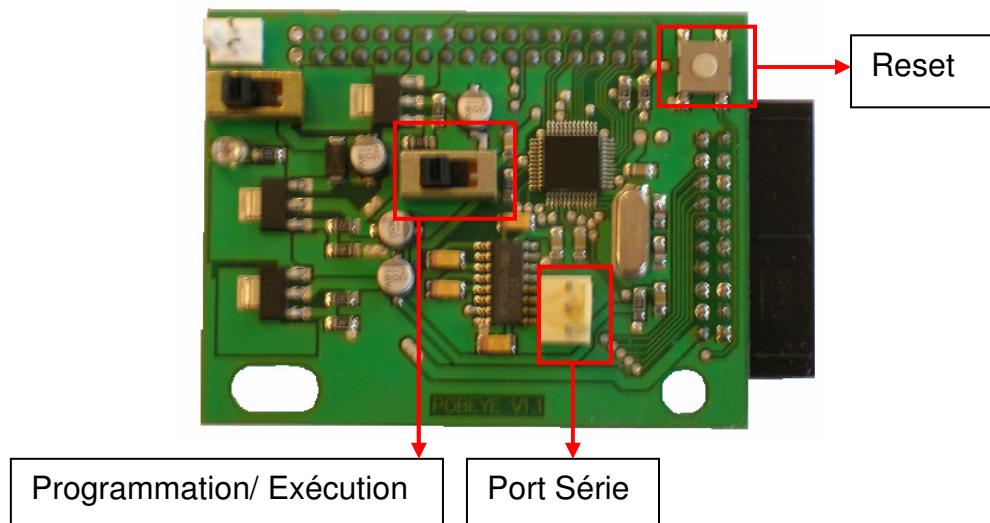


Figure 4 Programmation du POB-EYE

Remarque : si le module est déjà alimenté, il suffit de positionner l'interrupteur en mode « *Programmation* » et d'appuyer sur le bouton « *reset* ».

- *Exécution du programme*

Pour exécuter un programme, il faut positionner l'interrupteur de programmation/exécution en mode « **Exécution** » et allumer le module.

Remarque : si le module est déjà alimenté, il suffit de positionner l'interrupteur en mode « *Exécution* » et d'appuyer sur le bouton « *RESET* ». Attention : si votre système comporte d'autre carte fille autre que le POB-LCD128, il est nécessaire d'éteindre l'ensemble : certaines cartes filles ont besoin de redémarrer.

2.4 Port série (UART)

Le lien série permet de programmer le module POB-EYE et de communiquer avec l'extérieur lors de l'exécution du programme (c.f POB-Terminal).

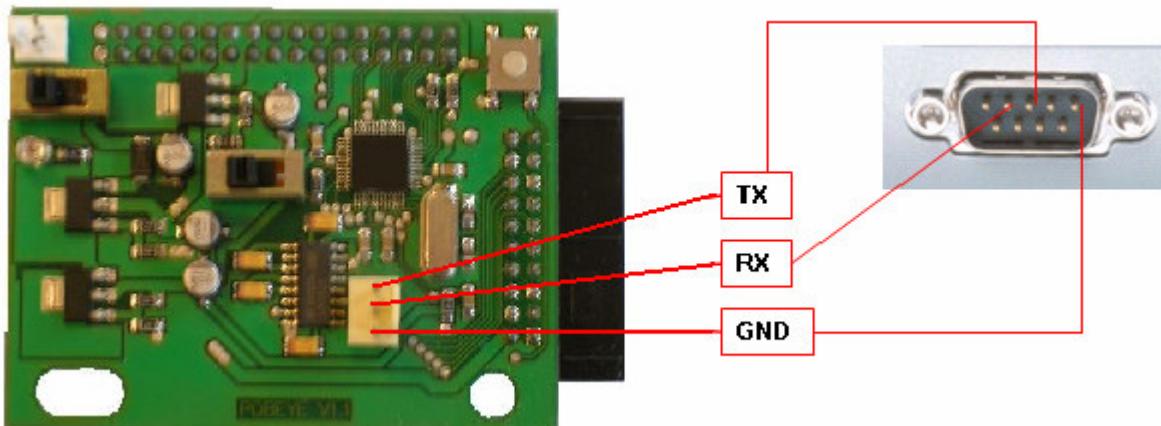


Figure 5 Port série

Remarque : Pour connecter le module POB-EYE à un ordinateur, un câble droit est nécessaire. Ce câble est fourni avec le module.



Attention : Pour convertir les différents niveaux de tension, un MAX232 est déjà présent sur le module POB-EYE. Les repères TX, RX et GND sont pris d'un point de vue PC.

2.5 Connecteur HE10

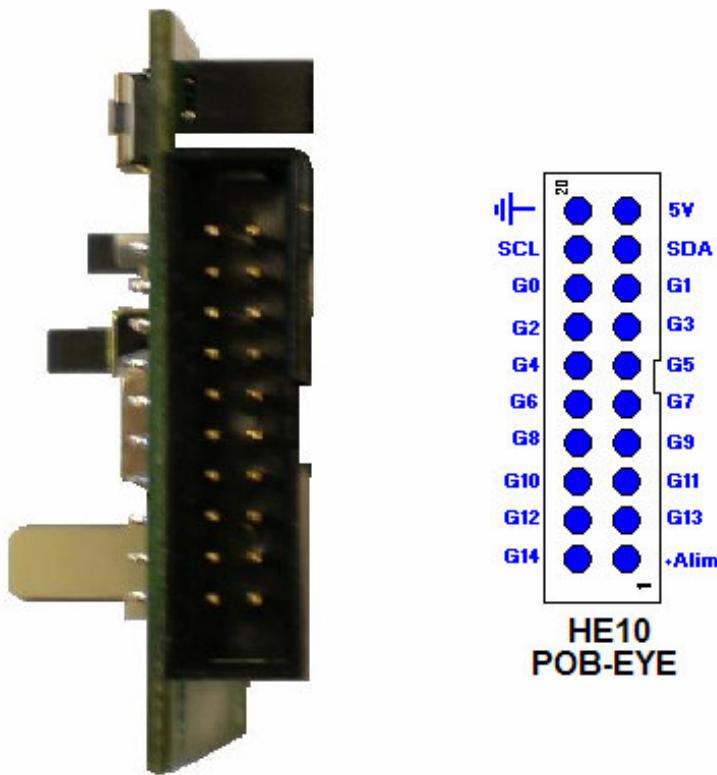


Figure 6 Bus I2C et entrées-sorties

Le connecteur HE10 permet à l'utilisateur de brancher ses propres cartes filles au module POB-EYE.

Ce connecteur donne accès au bus I2C (qui possède déjà ses résistances de pull-up), aux 15 entrées-sorties, au +5V, à la masse et à l'alimentation du POB-EYE.

Attention : Le module POB-EYE peut être directement alimenté par une carte fille qui lui fournirait le +Alim.

Une carte fille peut utiliser le +5V pour sa propre alimentation. Le courant maximum que peut fournir le POB-EYE est de 180 mA.

Attention : La tension de sortie des Entrées-sorties est de 3.3V. Malgré leur 5 Volt tolérant en entrée, elles ne supportent pas un courant de plus de 10 mA.

Les informations sur le connecteur HE10 sont dans le tableau suivant :

Broche	Type	Numéro sur le connecteur HE10
G0	Entrée/Sortie	16
G1	Entrée/Sortie	15
G2	Entrée/Sortie	14
G3	Entrée/Sortie	13
G4	Entrée/Sortie	12
G5	Entrée/Sortie	11
G6	Entrée/Sortie	10
G7	Entrée/Sortie	9
G8	Entrée/Sortie	8
G9	Entrée/Sortie	7
G10	Entrée/Sortie	6
G11	Entrée/Sortie	5
G12	Entrée/Sortie	4
G13	Entrée/Sortie	3
G14	Entrée/Sortie	2
SCL	Sortie	18
SDA	Entrée/Sortie	17
+5V	Sortie	19
+Alim		1
GND		20

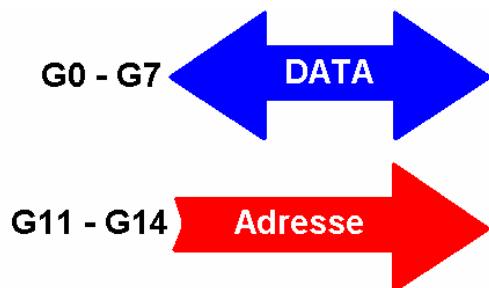
Figure 7 Connecteur HE10

2.6 Le bus POB-EYE

POB-Technology utilise les entrées-sorties sur le connecteur HE10 comme un bus d'adresses et de données.

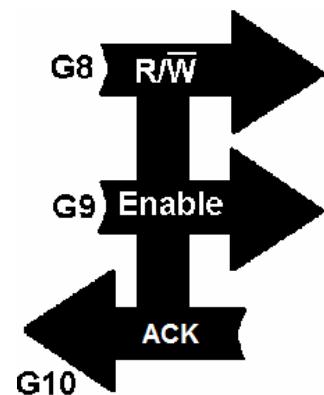
Les broches G0 à G7 sont utilisées pour les données. Les broches G11 à G14 sont utilisées pour le bus d'adresses. Ce système permet d'avoir 16 cartes sur le bus de POB-EYE.

Ce bus permet de connecter des cartes filles telles que des cartes de commande de servomoteurs ou de capteurs...



Ce bus est complété par 3 signaux de pilotage :

- Broche G8 : signal « R/W » indique une écriture ou une lecture sur le périphérique. Si G8 est à l'état bas, POB-EYE écrit des données vers la carte fille. Si G8 est à l'état haut, POB-EYE lit les données en provenance du périphérique.
- Broche G9 : signal « ENABLE » permet de valider les signaux du bus.
- Broche G10, signal « ACK » permet de connaître l'état du périphérique (le périphérique est-il prêt à recevoir des données).



Manuel Utilisateur du POB-Bot

Le tableau suivant récapitule le rôle de chacune des broches :

Broche	Nom sur le bus POB-EYE	Type	Numéro sur le connecteur HE10
G0	DATA 0	Entrée/Sortie	16
G1	DATA 1	Entrée/Sortie	15
G2	DATA 2	Entrée/Sortie	14
G3	DATA 3	Entrée/Sortie	13
G4	DATA 4	Entrée/Sortie	12
G5	DATA 5	Entrée/Sortie	11
G6	DATA 6	Entrée/Sortie	10
G7	DATA 7	Entrée/Sortie	9
G8	R/W	Sortie	8
G9	Enable	Sortie	7
G10	ACK	Entrée	6
G11	Adresse 0	Sortie	5
G12	Adresse 1	Sortie	4
G13	Adresse 2	Sortie	3
G14	Adresse 3	Sortie	2

Figure 8 Bus POB-EYE

Les différents périphériques connectés sur le bus POB-EYE disposent des adresses suivantes :

Adresse du périphérique	Nom du périphérique
0	POBLCD128
1	POBLCD128
2	POB-PROTO
3	réservee
4	réservee
5	réservee
6	réservee
7	réservee
8	réservee
9	réservee
10	réservee
11	réservee
12	Libre pour l'utilisateur
13	Libre pour l'utilisateur
14	Libre pour l'utilisateur
15	Libre pour l'utilisateur

Les adresses dites réservées sont celles que POB-TECHNOLOGY a choisies pour ses périphériques. Bien sur, si l'utilisateur le désire, il peut les utiliser pour ces propres périphériques.

2.7 Fonctionnement du bus POB-EYE depuis une carte d'extension

Pour expliquer le fonctionnement d'une carte fille sur le bus du POB-EYE, nous nous baserons sur le schéma électrique suivant :

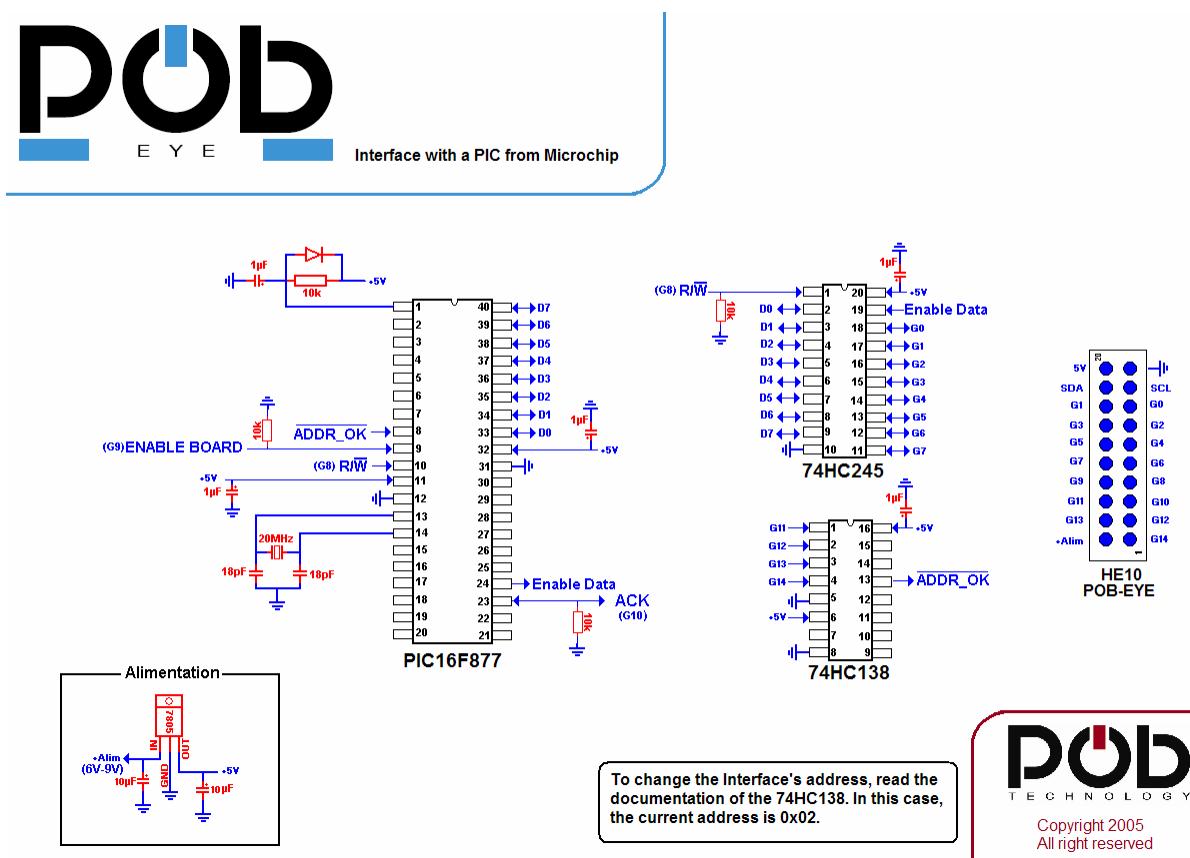


Figure 9 Schéma électrique de base d'une carte fille

Ce schéma permet de comprendre comment le module POB-EYE communique avec un microcontrôleur externe. Cet exemple vous donne une solide base pour fabriquer vos propres extensions.

Ce schéma et le code source des fonctions de communication se trouvent sur le Cd-rom livré avec le module POB-EYE.

Le code source contient uniquement les fonctions de communication entre le POB-EYE et la carte d'extension pour un microcontrôleur PIC16F877.

- *Ce schéma est composé de 3 circuits intégrés :*

74HC138

Ce circuit permet de décoder l'adresse posée sur le bus d'adressage. Lorsque l'adresse est valide, le signal /ADDR_OK est à l'état bas. A l'inverse, quand l'adresse n'est pas valide, /ADDR_OK est à l'état haut.

Dans cet exemple, le 74HC138 est configuré pour fonctionner avec l'adresse 0x02. Cette adresse est réservée à POB-TECHNOLOGY, pour plus d'information se référer à la figure 9.

74HC245

Ce circuit permet de faire circuler les données entre le POB-EYE et la carte fille. Il peut être validé ou non, et dans ce cas là, la totalité de ces broches (celles des données) se mettent en état de haute impédance. Le circuit est validé quand la pin 19 (/Enable) passe à l'état bas.

PIC16F877

C'est le cœur de la carte fille. Grâce à ses entrées analogiques et ses entrées -sorties, le PIC16F877 donne la possibilité de rajouter un nombre conséquent de fonctionnalités à votre système.

De plus, ce microcontrôleur est peu coûteux et l'outil de développement est gratuit.

- *Les signaux du bus POBEYE :*

Nom	Nom sur le 16F877	Sens	PIN	Description
/ADDR_OK	PORTE 0	Entrée	8	Signal indiquant que l'adresse est valide.
ENABLE_BOARD	PORTE 1	Entrée	9	Permet au POB-EYE de Valider la carte.
Read And Write	PORTE 2	Entrée	10	Indique si le POB-EYE lit ou écrit.
ACK	PORTC 4	Entrée/Sortie	23	Permet de dire au POB-EYE que la carte est prête.
ENABLE_DATA	PORTC 5	Sortie	24	Permet de valider les données présentes sur le 74H245.
	PORTB	Entrée/Sortie	33-40	Les données transitent par ce port.

▪ Fonctionnement du bus POB-EYE :

La logique du protocole de communication entre le POB-EYE est le PIC16F877 s'explique comme suit :

La carte reçoit une donnée**Attendre que :**

- l'adresse soit bonne (/ADDR_OK à 0).
- la carte soit validée (ENABLE_BOARD à 1).
- le POB-EYE soit en mode écriture (READ_AND_WRITE à 0).
- personne n'utilise ACK (ACK à 0).

Configurer :

- ACK en sortie
- le PORTB pour recevoir les données venant du POB-EYE

Mettre :

- ACK à 0
- ENABLE_DATA à 0

Enregistrer :

- Les données présentes sur le PORTB

Mettre :

- ENABLE_DATA à 1
- ACK à 1

Attendre que :

- la carte soit dévalidée (ENABLE_BOARD à 0) par le POB - EYE

Configurer :

- ACK à zéro. Ceci est une astuce pour gagner du temps : si ACK devient une entrée, la résistance de PULL-DOWN placée sur ACK fait croire au POB-EYE que ACK est bien passé à zéro.

Mettre :

- ACK à 0. Ne pas omettre cette étape : le matériel peut être détruit.

La carte envoie une donnée

Attendre que :

- l'adresse soit bonne (/ADDR_OK à 0).
- la carte soit validée (ENABLE_BOARD à 1).
- le POB-EYE soit en mode lecture (READ_AND_WRITE à 1).
- personne n'utilise ACK (ACK à 0).

Configurer :

- ACK en sortie
- le PORTB pour envoyer les données

Mettre :

- ACK à 0
- ENABLE_DATA à 0
- Les données à envoyer sur le PORTB
- ACK à 1

Attendre que :

- la carte soit dévalidée (ENABLE_BOARD à 0)

Mettre :

- ENABLE_DATA à 1

Configurer :

- ACK à zéro. Ceci est une astuce pour gagner du temps, si ACK devient une entrée, la résistance de PULL-DOWN placée sur ACK fait croire au POB-EYE que ACK est bien passé à zéro.

Mettre :

- ACK à 0. Ne pas omettre cette étape : le matériel peut être détruit.

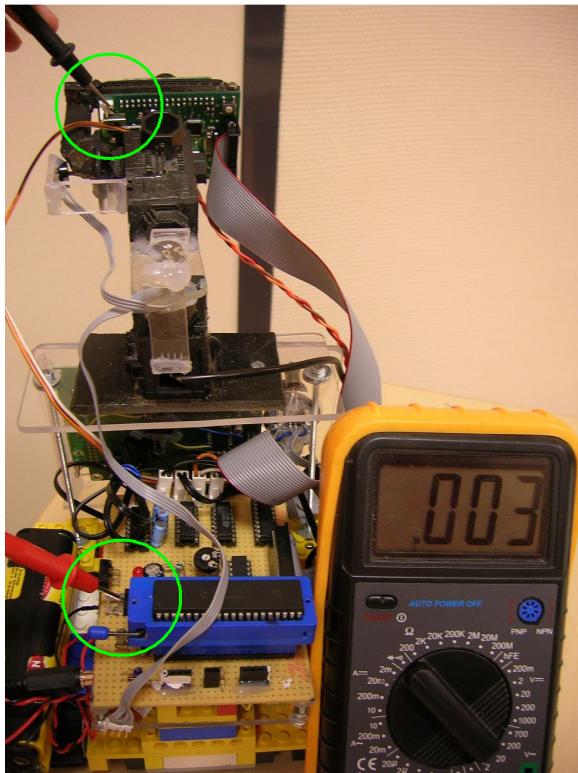
Configuration des différents signaux à l'initialisation de la carte fille

Nom	Sens	Etat du signal
/ADDR_OK	Entrée	
ACK	Entrée (devient une sortie uniquement lors de la communication)	
ENABLE_DATA	Sortie	Etat Haut
PORTB	Entrée	
ENABLE_BOARD	Entrée	
READ_AND_WRITE	Entrée	

Remarques :

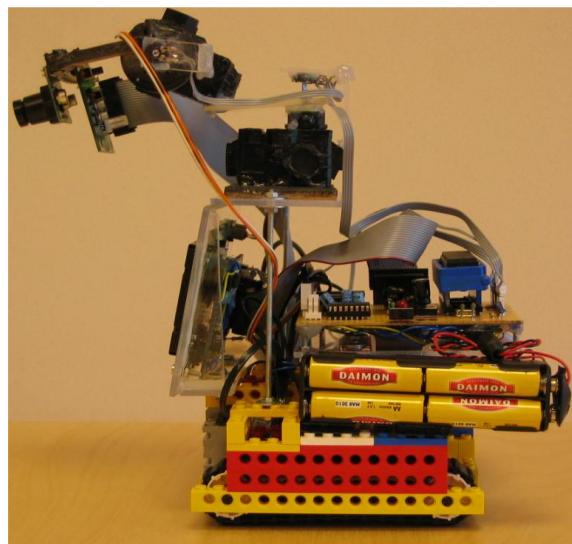
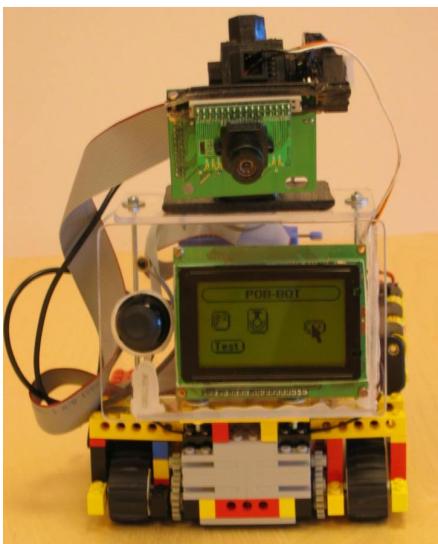
- Quand un nouveau programme est chargé dans le POB-EYE et des cartes filles sont ajoutées, pour des raisons de synchronisation, il est nécessaire d'éteindre puis de redémarrer l'ensemble du système.

- Lorsque vous ajoutez une carte fille à votre POB-EYE, **avant la mise sous tension**, vérifiez impérativement avec un multimètre en position « testeur de diode » si le câblage est parfaitement effectué. Pour cela, il vous suffit de placer un des contacts du multimètre sur la masse du POB-EYE, et l'autre sur une masse de votre montage. Si un contact franc apparaît, alors votre carte fille est correctement branchée dans votre circuit.



Manuel Utilisateur du POB-Bot

- *Exemple d'application du bus POB-EYE :*



Dans cet exemple d'application, le POB-EYE a 2 cartes filles :

- un POB-LCD128. Cet écran permet de dialoguer avec l'utilisateur via une interface homme machine.
- Une carte fille conçue sur la base du schéma électrique (figure 10).

Cette carte fille permet de :

- Commander 4 moteurs à courant continu (2 pour la tête et 2 pour la traction)
- Capturer la position de 4 potentiomètres (2 pour contrôler la tête et 2 pour le joystick)
- Déetecter l'appui d'un bouton poussoir.

Une vidéo de démonstration des capacités de ce robot est disponible sur notre site www.pob-technology.com.

2.8 Connecteur capteur CMOS

Ce connecteur permet de brancher le capteur CMOS OV6620 au module POB-EYE.

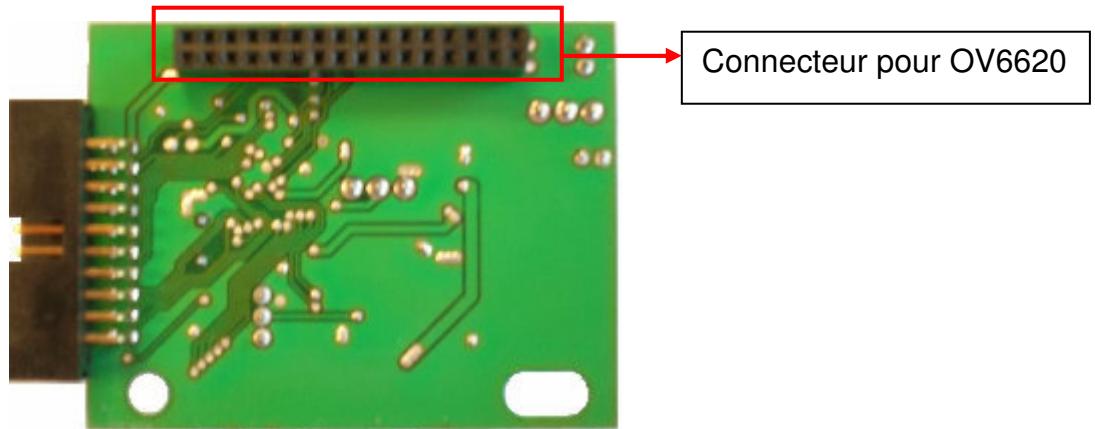


Figure 10 Connecteur CMOS

Attention : Le capteur CMOS se branche sur la gauche du connecteur.

Pour configurer les registres de l'OV6620, utilisez la fonction « *SendToCam* » de la librairie.

2.9 Dimension du module

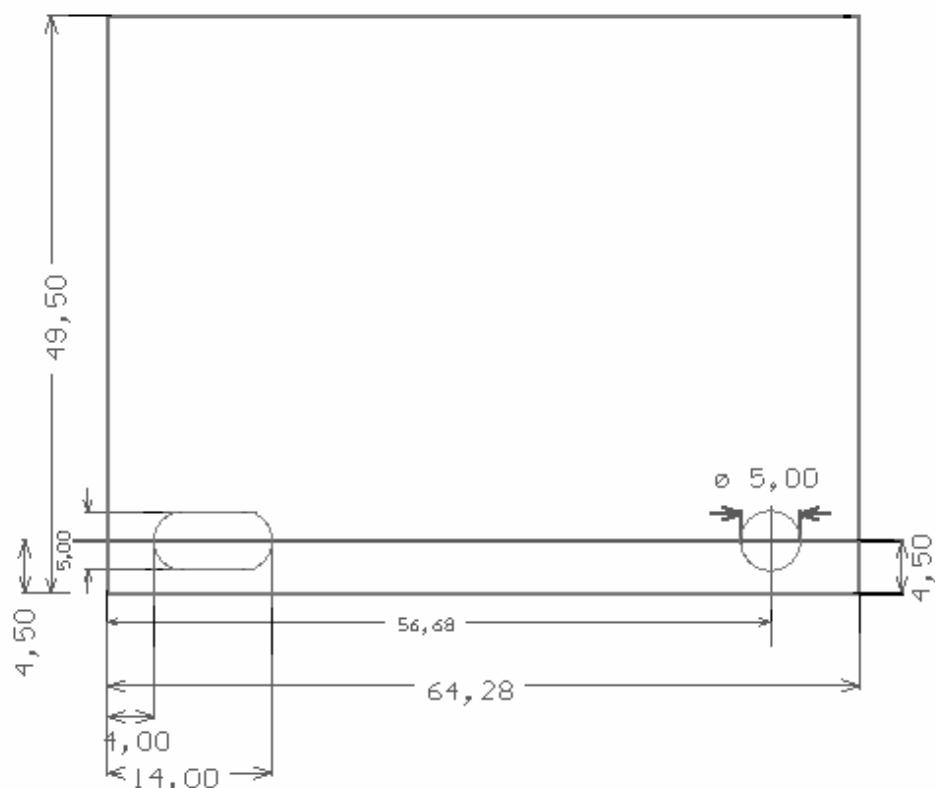
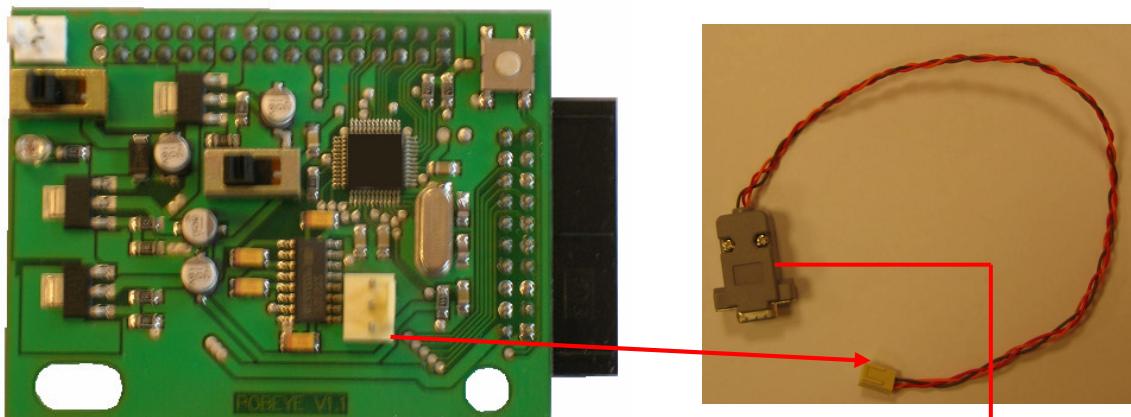


Figure 11 Dimension du module POB-EYE

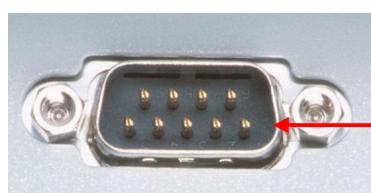
2.10 Connecter POB-EYE à votre ordinateur

- *Port série*

Il suffit de connecter le câble fourni avec POB-EYE :

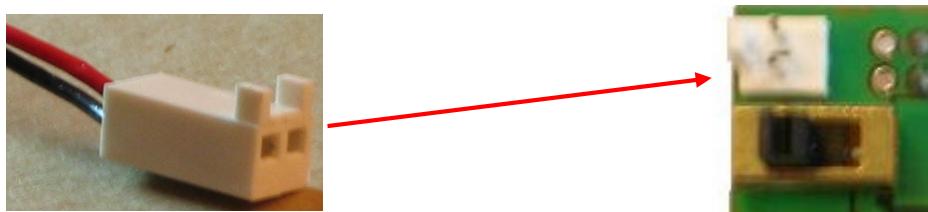


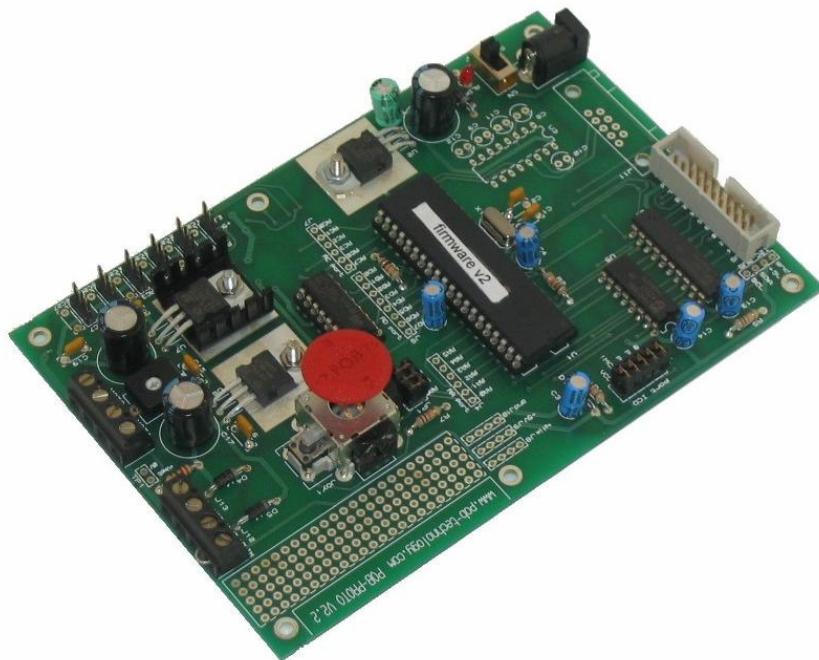
Et de relier l'extrémité du câble à un port série de votre ordinateur.



- *Alimentation du POB-EYE*

Il faut connecter le câble d'alimentation au module POB-EYE.





Carte de Commande

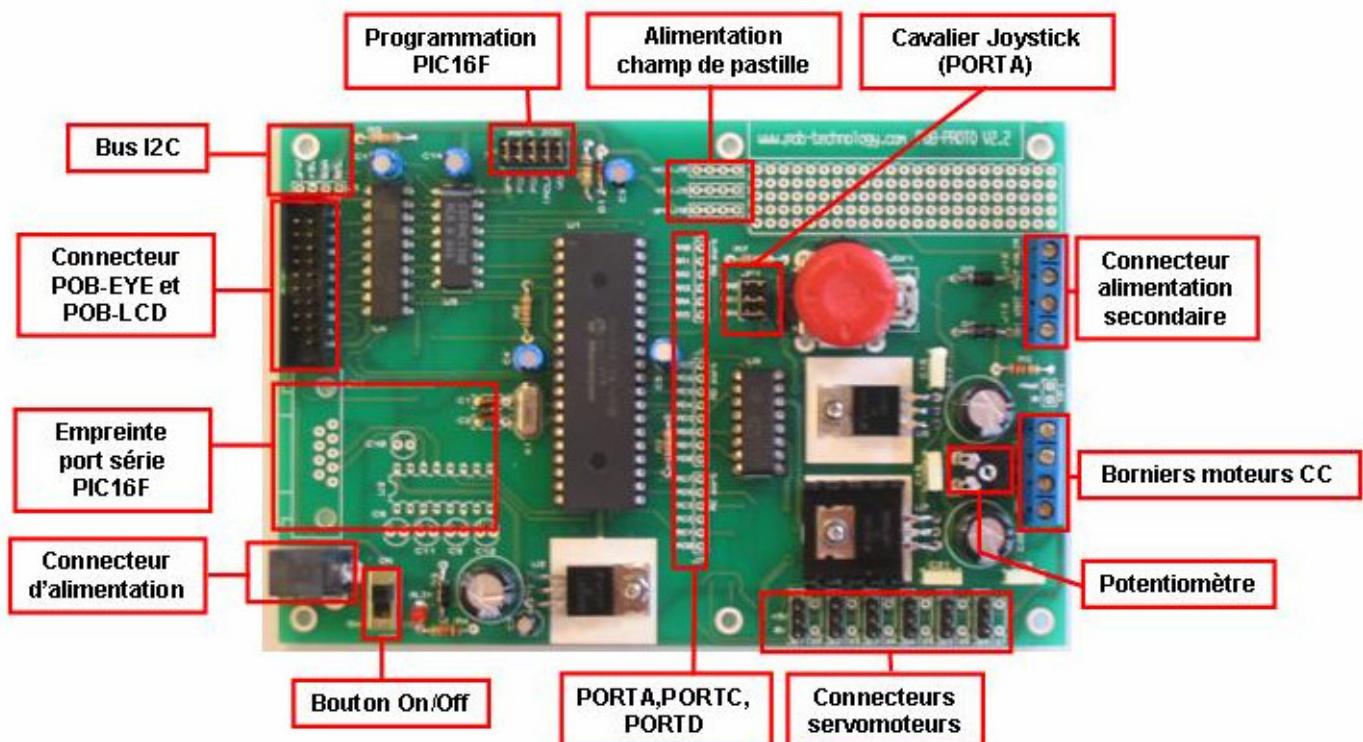
Commandez vos capteurs/actionneurs
branchés sur les Entrées / Sorties numériques et analogiques
Bus Pob et I2C

3 POB-PROTO

POB-PROTO est un périphérique destiné au POB-EYE. Cette carte permet de construire rapidement un robot fonctionnel. Elle est équipée de :

- 6 connecteurs pour commander des servomoteurs (type « *Futaba* »)
- 1 joystick analogique et son bouton poussoir.
- un double pont en H permettant de contrôler 2 moteurs à courant continu.

Schéma général de la carte POB-PROTO :



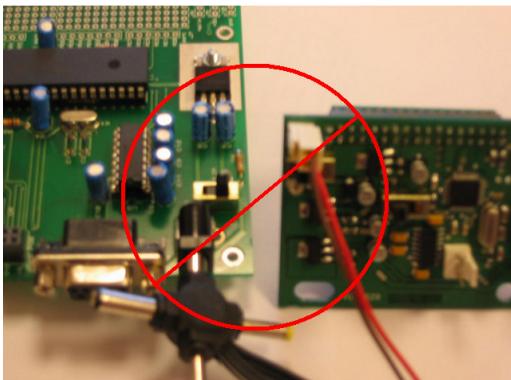
3.1 Description des différents éléments de la carte

- *Alimentation de la carte :*

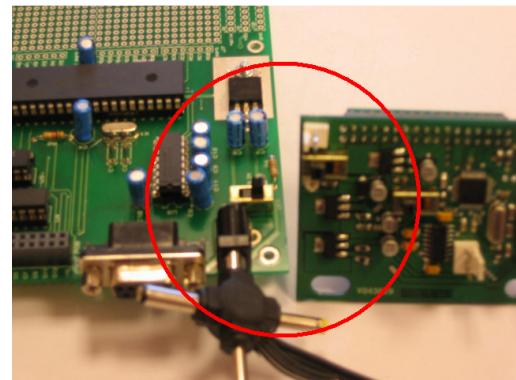


Elle se fait par le connecteur d'alimentation. Veuillez à bien respecter la polarité. Ce connecteur permet d'alimenter l'ensemble du système via la broche 2 du bus POB.

ATTENTION : pour éviter toute destruction de matériel il est obligatoire de ne pas connecter le POB-EYE par son connecteur d'alimentation si vous utiliser une alimentation sur la carte POB-PROTO.



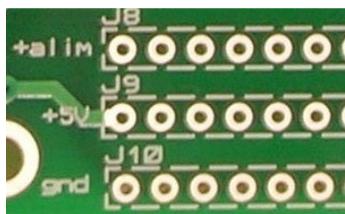
Ne pas alimenter POB-PROTO et POB EYE avec deux alimentations séparées



Alimentation effectuée uniquement du côté POB-PROTO



Sur la carte POB-PROTO se trouve un interrupteur permettant de contrôler l'alimentation générale du système.

▪ *Alimentation pour le champ de pastilles :*

La carte possède près de son champ de pastille 3 lignes servant à l'alimentation des composants supplémentaires.

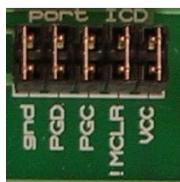
▪ *Connexion du POB-EYE et POB-PROTO :*

Le connecteur HE10 permet de relier la carte POB-PROTO au bus POB. La connexion se fait grâce au câble fourni. Le câble sert à relier les modules les uns aux autres via le POB-BUS.



Attention : Si vous décidez de fabriquer votre propre câble, vous devez respecter les règles suivantes : les brises doivent être serrées avec les détrompeur orienté tous dans le même sens. Dans le cas contraire, un branchement avec un tel câble endommagerait définitivement votre matériel.

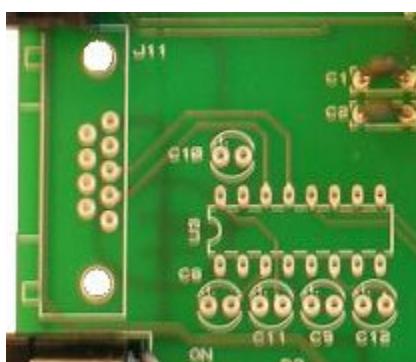
- *Programmation du PIC16F877 :*



POB-PROTO est muni d'un connecteur permettant la programmation du microcontrôleur par l'ICD2 de Microchip. L'utilisateur doit enlever les 5 cavaliers, puis placer son câble de programmation sur les broches du bas (GND, PGD, PGC, MCLEAR, VCC).

Une fois le PIC programmé, il faut remettre les 5 cavaliers. Pour effectuer la programmation, votre ICD2 doit être équipé de sa propre alimentation.

- *Le port Série :*



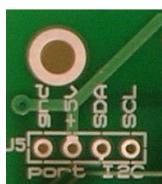
La carte POB-PROTO est équipée des emplacements nécessaires à l'intégration des différents composants pour le port série du PIC16F877.



Le module possède l'empreinte pour un connecteur DB9 femelle.

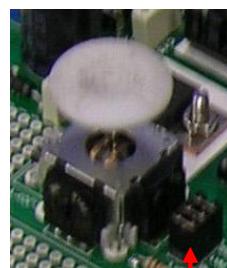


Pour convertir le niveau électrique des signaux série, POB-PROTO possède l'empreinte pour un circuit intégré de type MAX232 et des condensateurs.

Manuel Utilisateur du POB-Bot**■ Le bus I2C du POB-EYE**

En haut à droite de la carte, une empreinte permet d'utiliser le bus I2C du module POB-EYE. Les signaux SDA et SCL, GND et +5V sont présents.

Pour information, les résistances de PULL-UP sont déjà présentes.

■ Le Joystick

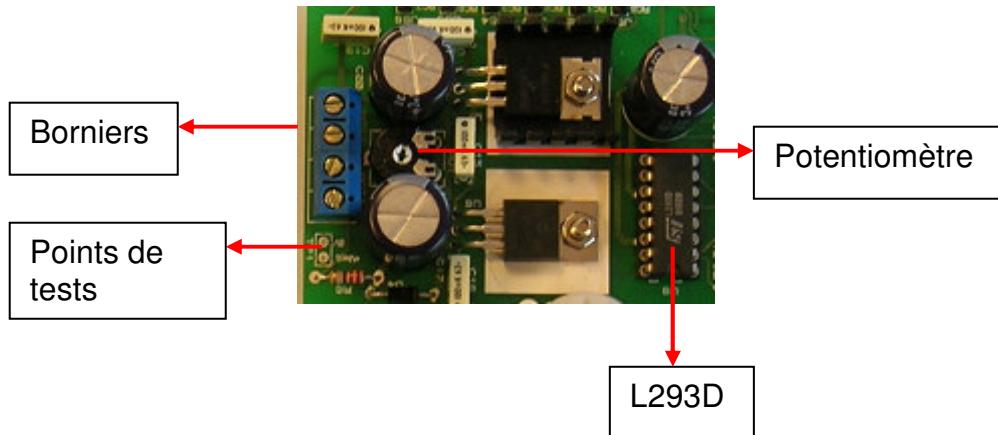
Cavaliers de raccordement

Le Joystick permet par exemple de commander un curseur sur l'interface graphique. Il fonctionne grâce à 2 potentiomètre reliés au PORTA (RA0 et RA1). Pour faire office de bouton de sélection, un bouton poussoir (rélié à RA4) est situé à la verticale du joystick.

Si l'utilisateur décide d'utiliser le PORTA, il peut enlever les cavaliers de raccordement.

En utilisant le joystick, l'utilisateur sera obligé d'utiliser RA3, RA4, RA5 en entrée analogiques (cf configuration du PORTA)

- *Le Pont en H*



Ce pont en H est relié au PORTD (RD0, RD1, RD2, RD3). Il permet de contrôler 2 moteurs à courant continu. Ces moteurs se connectent aux borniers bleus. Pour régler la tension aux bornes des moteurs, utilisez le potentiomètre.

Afin de mesurer cette tension, vous pouvez appliquer vos sondes de voltmètre sur les 2 points de tests.

Attention : le courant débiter par le pont en H ne doit pas excéder les 600mA.

Les broches du PORTD étant configurables indépendamment les unes des autres, les broches RD4, RD5, RD6, RD7 restent disponibles pour un usage général.

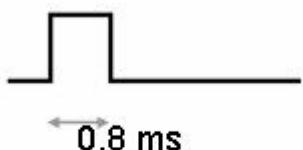
Attention : Si vous ne désirez pas utiliser le pont en H, il suffit d'enlever le L293D de son support.

■ Les connecteurs pour servomoteurs :

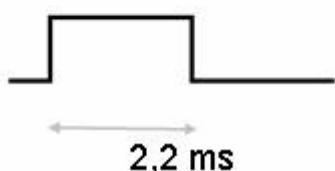
Ces connecteurs sont reliés respectivement aux broches RC0, RC1, RC2, RC3, RC6, RC7 du PORTC.

N'étant relié à rien de particulier, vous pourrez utiliser chacune de ces broches à votre guise (en l'absence de servomoteurs).

La valeur de consigne pour piloter les servomoteurs est comprise entre 0 et 255.



0,8 ms représente la valeur 0.



2,2 ms représente la valeur 255.

Attention : Si l'ensemble de vos servomoteurs consomme trop de courant, la température du régulateur de tension peut être élevée malgré la présence du radiateur.

Nous vous conseillons de ne jamais appliquer quoi que ce soit sur le régulateur (notamment les doigts), la chaleur dégagée pouvant l'endommagé.

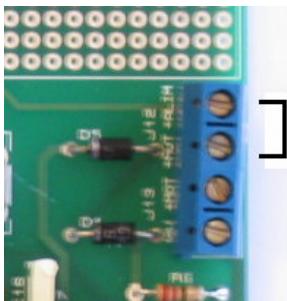
- *Connecteur pour alimentation secondaire des servomoteurs :*



L'utilisation des moteurs à courant continu et des servomoteurs peut provoquer des perturbations.

Ces perturbations peuvent par exemple provoquer un redémarrage intempestif de votre système.

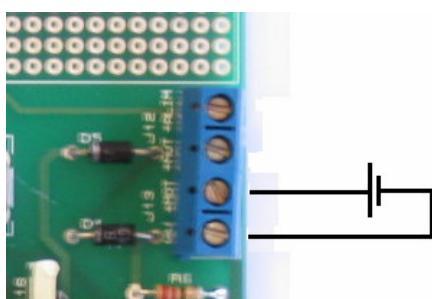
Pour palier à ce problème, la carte POB-PROTO possède 2 modes de fonctionnement :



Alimentation commune :

Si vous désirez utiliser l'alimentation déjà présente sur la carte, vous devez brancher un cavalier comme sur la photo de gauche.

L'alimentation commune est le branchement par défaut.



Alimentation secondaire :

En revanche, si vous désirez utiliser une alimentation externe, et ainsi limiter toutes sources de perturbations, vous devez :

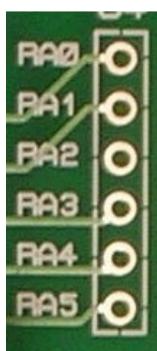
- débrancher le cavalier de l'alimentation commune.
- connecter votre alimentation externe comme sur l'illustration ci-jointe.

Attention : Prenez bien garde à respecter ces schémas. Un non respect de ces derniers pourrait endommager votre matériel en créant un court-circuit.

▪ ***PORTA, PORTC et PORTD :***

Par défaut les ports sont tous équipés de matériels (connecteur, ponts en H...) permettant l'utilisation rapide de la carte. Si vous le désirez, vous pouvez vous désolidariser du matériel (cavalier, enlever le composant) et utiliser chacun des ports comme bon vous semble.

PORTA



Le PORTA peut fonctionner en 2 modes :

- Entrée analogique, sauf RA4 qui sera un entrée digital prenant les valeurs 0 ou 255.
- Entrée-sortie digitale, chacune des broches peut-être configurées séparément.

Si vous configurez le PORTA en entrées analogiques, l'appel à la fonction *GetPortAnalog* avec comme paramètre RA4 ne vous renverrez que 0 ou 255. Ceci est dû au fait que RA4 n'est pas une entrée analogique, mais seulement digitale.

Configurer en sortie, RA4 fonctionne en drain ouvert, il vous faut donc rajouté si nécessaire une résistance en RA4 et le +5V.

Fonctions pour utiliser le PORTA :

UInt8 GetPortA (void)

Retourne la valeur du PORTA.

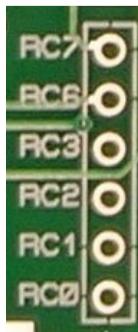
void SetPortA (UInt8 data)

Configure la valeur du PORTA.

UInt8 GetPortAnalog (UInt8 RAx)

Retourne la valeur analogique de la broche RAx.

PORTC



Chacune des broches du PORTC peut-être configurée séparément.

L'utilisateur peut utiliser RC0, RC1, RC2, RC3, RC6 et RC7 pour commander 6 servomoteurs.

Remarque : Si l'utilisateur décide de reprogrammer le PIC16F877, il pourra en rajoutant un MAX232, utiliser le port série présent sur le PORTC.

Fonctions pour utiliser le PORTC :

UInt8 GetPortC (void)

Retourne la valeur du PORTC.

void SetPortC (UInt8 data)

Configure la valeur du PORTC.

void SetServoMotor(UInt8 RCx, UInt8 Pos)

Règle la position du servomoteur connecté à RCx.

void SwitchOffAllServo(void)

Arrête l'asservissement de tous les servomoteurs, ceci permettant une économie d'énergie.

void SwitchOnAllServo(void)

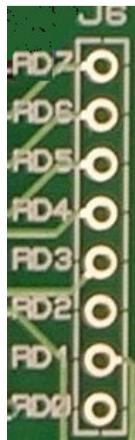
Réactive l'asservissement de tous les servomoteurs.

void SwitchOffOneServo(UInt8 Servo)

Arrête l'asservissement d'un servomoteur en particulier.

void SwitchOnOneServo(UInt8 Servo)

Réactive l'asservissement d'un servomoteur en particulier.

PORTD :

PORTD est uniquement un port d'entrée sortie, chacune des broches peut-être configurée séparément.

Fonctions pour utiliser le PORTD :**UInt8 GetPortD (void)**

Renvoie la valeur du PORTD

void SetPortD (UInt8 data)

Configure la valeur des pins du PORTD réglées en sortie.

Void SendToSapien(UInt8 order)

Envoie un ordre au Robotsapiens.

3.2 Exemple de configuration des ports :

Le programme PIC16F877 livré avec le POB-PROTO permet d'accéder aux différents ports disponibles.

```
PobProto Proto;  
Proto.porta=ALL_PORTA_AS_ANA;  
Proto.portc=RC7_AS_DI | RC6_AS_DI | RC3_AS_DI | RC2_AS_DI | RC1_AS_DI | RC0_AS_SERVO;  
Proto.portd=RD7_AS_DI | RD6_AS_DI | RD5_AS_DI | RD4_AS_DI | RD3_AS_DO | RD2_AS_DO | RD1_AS_DO | RD0_AS_DO;  
SetPobProto(&Proto);
```

Dans l'exemple ci-dessus nous configurons le PORTA en entrée analogique. Sur le PORTC, RC7, RC6, RC3 RC2, RC1 sont configurées en entrées, RC0 est quant à lui configurer en commande de servomoteur.

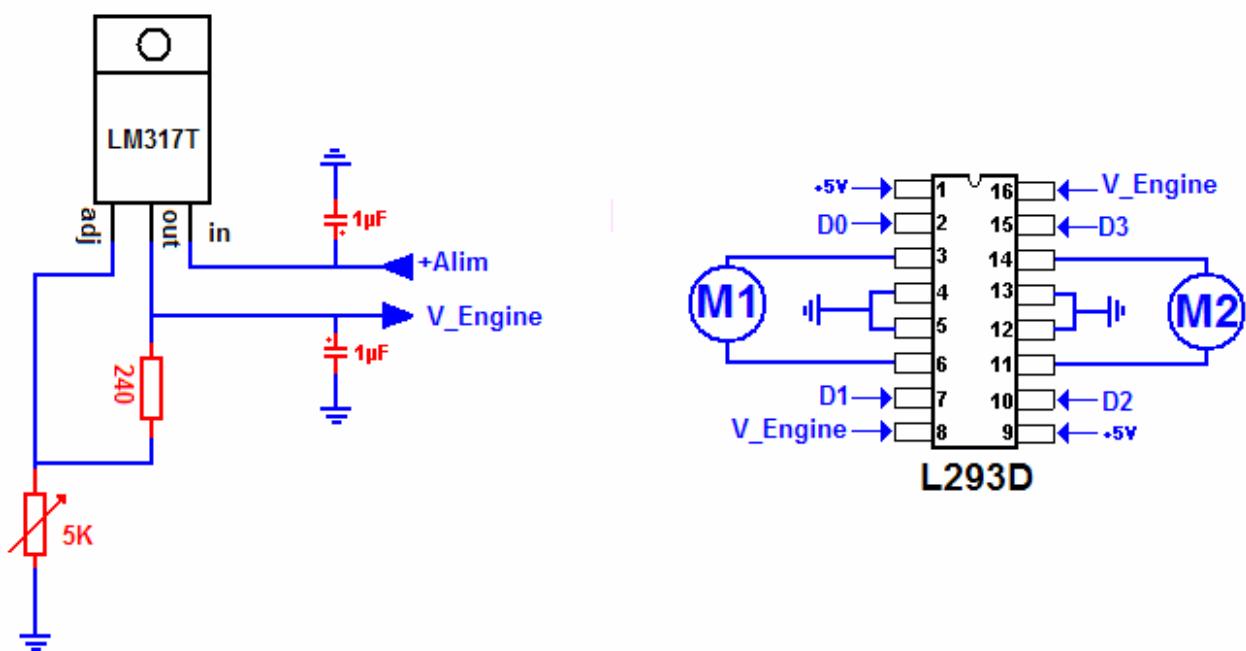
Concernant le PORTD, RD7, RD6, RD5, RD4 sont configurer en entrées digitales, RD3, RD2, RD1, RD0 sont réglées comme des sorties digitales.

3.3 Exemples de montages autour du POB-PROTO :

- *Commande d'un bloc moteur :*

Généralement, un robot utilise un système à deux moteurs à courant continu couplés à une roue dite « folles » pour se déplacer. La commande des moteurs nécessite un étage de puissance.

L'électronique à rajouter au POB-PROTO pour commander des moteurs à courant continu est montrée dans le schéma suivant.



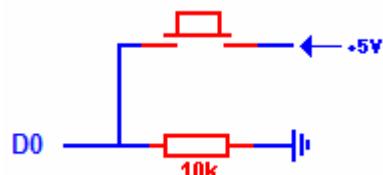
Ce schéma est basé sur un L293D (double pont en H) dédié à la commande de moteurs. Chaque moteur est commandé par 2 broches du microcontrôleur.

Dans cet exemple, les broches D0, D1, D2, D3 du PORTD servent à commander les moteurs. Dans l'application côté POB-EYE, il faudra les configurer en sorties.

Le LM317T est un régulateur de tension ajustable. L'utilisateur pourra à l'aide de la résistance ajustable, obtenir la tension idéale pour ses moteurs.

■ *Gestion d'un bouton poussoir :*

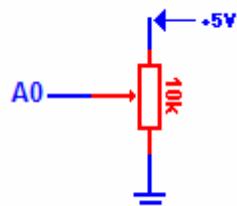
Si l'utilisateur décide de créer sa propre Interface Homme Machine, il aura sûrement besoin d'un système mécanique de navigation. La solution la plus simple est d'utiliser des boutons poussoir. Dans le schéma suivant, vous trouverez le montage de base pour relier un bouton poussoir au POB-PROTO.



L'utilisateur doit configurer la broche D0 en entrée et lire le PORTD avec les masques adéquats pour récupérer la valeur de la broche D0.

■ *Acquisition d'une valeur analogique :*

Le schéma suivant montre comment relier une résistance ajustable à une entrée du PORTA.

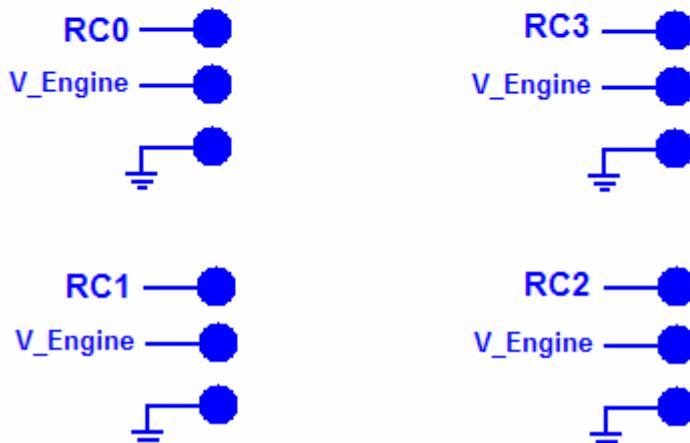
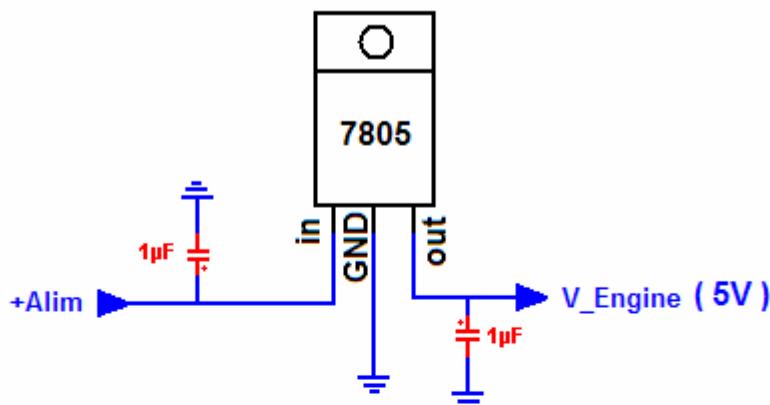


Attention : sur le PIC16F877, sur le PORTA, seules RA0 RA1 RA2 RA3 et RA5 sont des entrées analogiques. RA4 est une entrée digitale, qui renverra 0 ou 255 lors de l'appel de la fonction *GetPortAnalog*.

▪ *Commande de servomoteurs :*

Uniquement les broches RC0, RC1, RC2, RC3, RC6 et RC7 du PORTC peuvent être utilisés pour contrôler des servomoteurs.

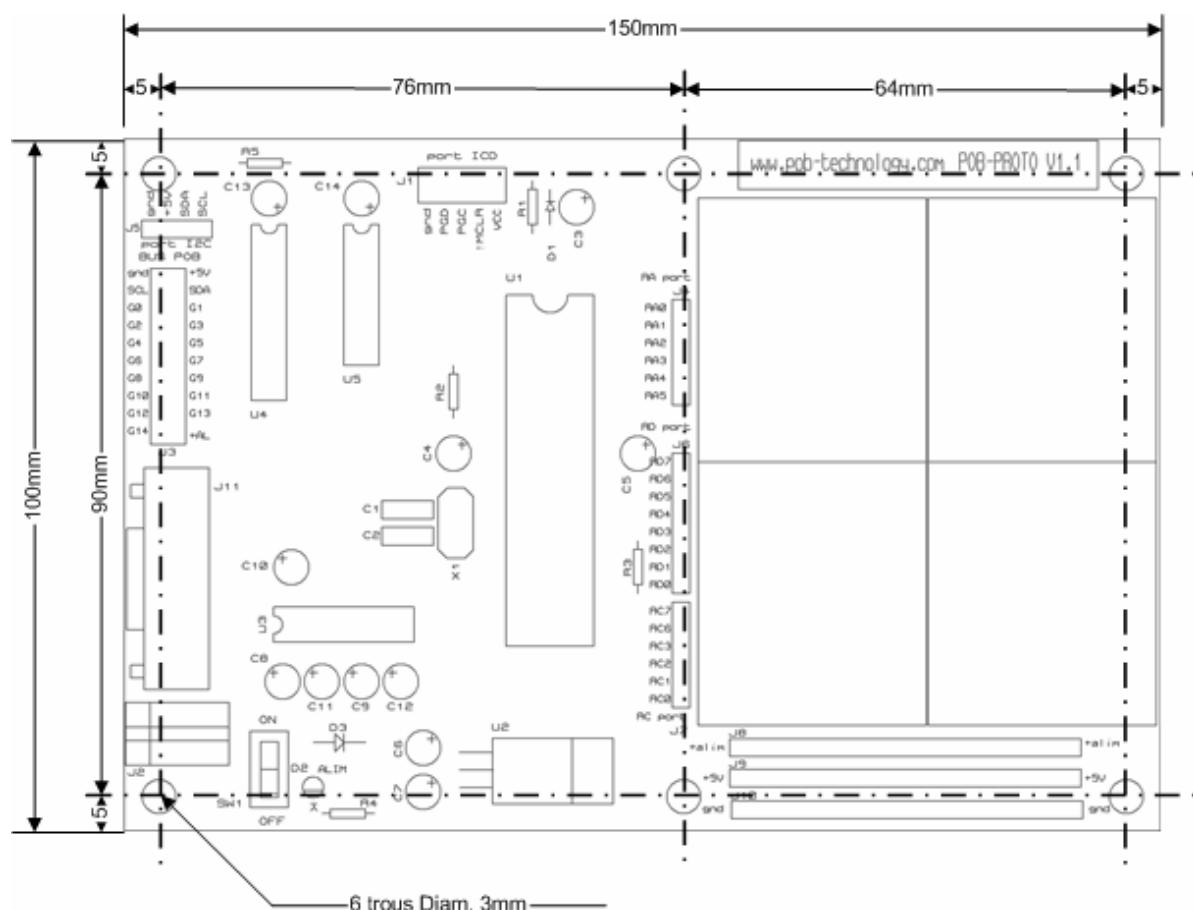
Le schéma suivant montre comment relier des servomoteurs à la carte POB-PROTO :



Attention : Pour éviter que les appels de courant des servomoteurs perturbent le PIC16F877, il est fortement conseillé de dédier un régulateur de tension à l'alimentation des servomoteurs.

Remarque : L'utilisateur peut configurer RC0, RC1, RC2 et RC3 comme des commandes de servomoteurs et configurer librement RC6 et RC7 comme entrées ou sorties digital.

3.4 Informations mécaniques :





Module d’Affichage Graphique

Visualisation en temps réel de l’image du Pob-Eye
Développement d’une interface graphique

4 POB-LCD

POB-LCD128 est une carte d'extension pour le module POB-EYE. POB-LCD128 permet au module POB-EYE d'afficher toutes sortes de graphiques.

Ces deux modules sont étroitement liés : POB-LCD128 permet d'afficher en temps réel l'image capturée par la caméra du POB-EYE et de gagner du temps dans le développement de votre application.

Pour faciliter le dessin sur l'écran LCD, POB-Technology a mis au point un logiciel permettant d'intégrer facilement des images dans une application. Cet outil se nomme « *POB-Bitmap* » et est décrit dans le chapitre consacré au POB-TOOLS.

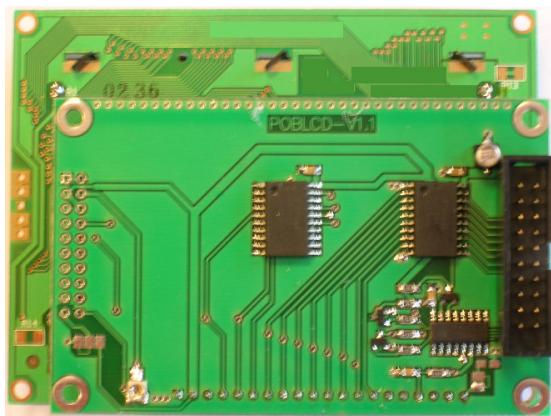
4.1 Description matérielle

Techniquement, le POB-LCD128 est composée de deux parties :

- un écran LCD de type matriciel de 128 par 64 pixels.

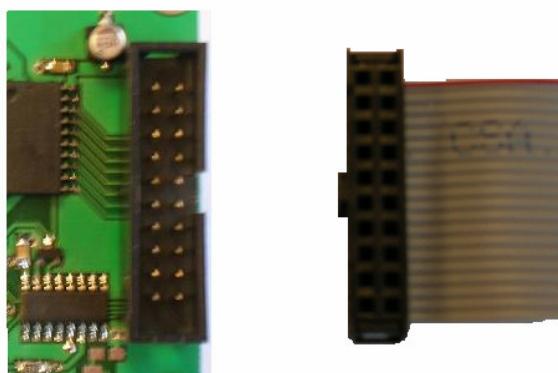


- une interface électronique.

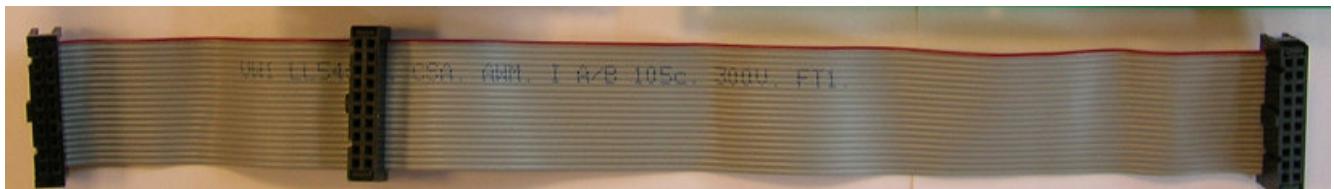


L'interface électronique permet d'utiliser POB-LCD128 sur le bus POB-EYE. Il prend les adresses 0 et 1 du bus POB-EYE.

4.2 Connecter POB-LCD128 avec POB-EYE



Il suffit de relier le câble (livré avec le POB-EYE) avec POB-EYE et POB-LCD128.



Attention : Si vous décidez de fabriquer votre propre câble, vous devez respecter les règles suivantes : les brises doivent être serties avec les détrompeur orienté tous dans le même sens. Dans le cas contraire, un branchement avec un tel câble endommagerait définitivement votre matériel.

4.3 Dessiner sur le POB-LCD128

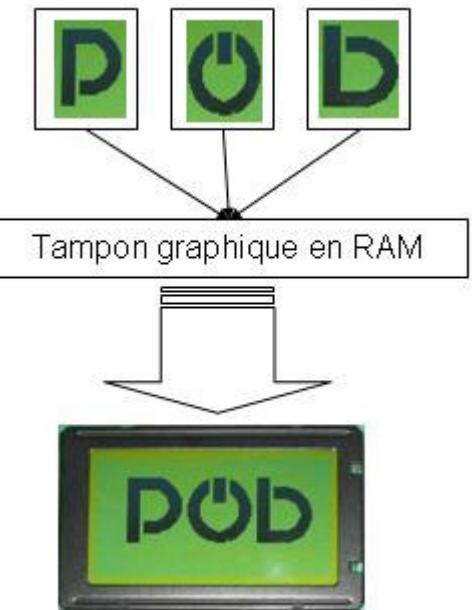
La mise en œuvre du POB-LCD se fait grâce à la librairie fournie avec le module.

- *Fonctionnement du tampon graphique*

Les capacités du POB-EYE permettent d'utiliser un tampon graphique en RAM. Toutes les opérations de dessin manipulent ce tampon, puis une fois fini, le tampon est transféré vers la mémoire du POB-LCD.

Cela permet un traitement efficace de l'affichage et d'éviter la visualisation de pixels se dessinant un par un sur l'écran. Ce délai viendrait en autre des allés retours de données entre le POB-EYE, et la mémoire interne de l'écran LCD.

Avec cette méthode de tampon, les opérations se font directement dans la RAM du POB-EYE. Une fois finie, le buffer est transféré de la RAM du POB-EYE à celle de l'écran LCD.



- *A propos des pixels*

Un écran LCD noir et blanc utilise un bit pour colorer un pixel. Avec un octet, on va donc gérer 8 pixels sur l'écran. Au minimum, le buffer graphique utilise donc 8129 bits (l'écran du POB-LCD est d'une taille de 128 pixels par 64 pixels) en mémoire sur le module POB-EYE.

$$128 \times 64 \text{ pixels} = 8129 \text{ bits soit environ 1Koctets}$$

Le problème de ce tampon graphique est d'avoir un rapport « 1 bit = 1 pixel ». Les opérations sur les bits sont coûteuses pour un processeur. En effet pour changer la valeur d'un bit en mémoire il faut obligatoirement lui appliquer un masque. Ce dernier étant différent si l'on désire le mettre à 1 ou à 0. Ce type d'opération est reproduit 8129 fois, la manipulation bit à bit d'un buffer prend donc un temps non négligeable.

L'avantage du rapport « 1 bit = 1 pixel » est de consommer le strict minimum d'espace mémoire. Une autre manière plus rapide de dessiner à l'écran est de considérer qu'un octet colore un pixel. Ainsi nous n'avons plus de décalage et autres masques pour allumer ou éteindre un pixel. Le désavantage de cette solution est de coûter 8 fois plus de place en mémoire que la solution précédente.

128 par 64 pixels = 8129 pixels
1 bit par pixel, le buffer est égal 1K octets.
1 octet par pixel, le buffer est égal à 8K octets.

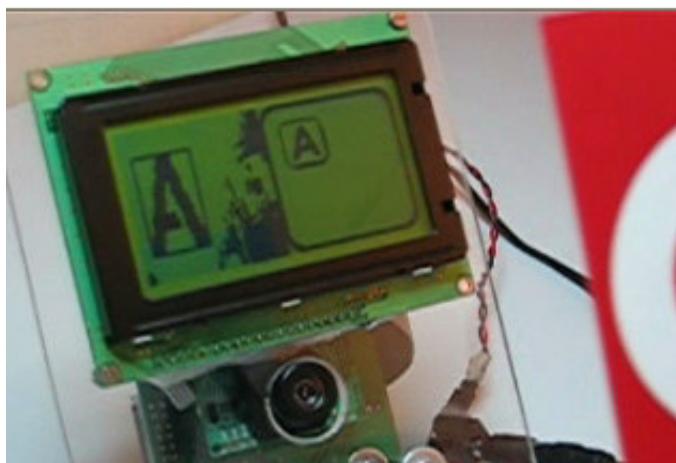
Conclusion :

- Utiliser **1 octet** par pixel permet d'accélérer le traitement des fonctions graphiques, mais prend de la place en mémoire.
- Utiliser **1 bit** par pixel permet d'économiser la mémoire du POB-EYE, mais peut prendre un certain temps.

Pour offrir à l'utilisateur un contrôle total de l'affichage sur le POB-LCD128, la librairie fournie avec le module est capable de gérer un tampon de 1 bit par pixel ou un tampon de 1 octet par pixel.

■ *Partage d'écran*

La librairie fournit est capable d'afficher sur l'écran complet, ou de le partager en deux écrans de 64 par 64 pixels. Si dessous, l'application affiche les images vues par la caméra sur la partie gauche de l'écran et affiche une interface utilisateur sur la partie droite.



Pour comprendre comment utiliser les fonctions de la librairie graphiques, vous pouvez vous reporter au code source exemple4 fourni avec le POB-EYE.

5 Aide sur les fonctions de la librairie LibPOB

Pour obtenir l'aide sur la librairie fournit avec le POB-EYE, cliquez sur le bouton [POBLIB's Help](#) dans l'outil POB-COMPILER.

Votre explorateur Internet se lancera pour afficher la page d'aide.

[Main Page](#) | [Data Structures](#) | [File List](#) | [Data Fields](#) | [Globals](#)

POB Library Documentation

1.0

Generated by  1.4.1

This is a documentation from  All rights reserved.

A partir de cette page, vous avez accès à toutes les définitions des fonctions et autres structures utilisées dans la librairie LibPOB :

Functions

```
void SetIOWay (UInt32 Value)
UInt16 GetInput (void)
void SetOutput (UInt32 Value)
void ClrOutput (UInt32 Value)
void WriteByte (UInt16 Addr, UInt8 Data)
UInt8 ReadByte (UInt16 Addr)
```

Detailed Description

I/O Functions.

Function to manage the I/O and bus for POB-EYE.

Author:

Pierre SEGUIN. POB-Technology

Function Documentation

void ClrOutput (UInt32 Value)

Clear the output.

Parameters:

Value : use 1 to clear an output, 0 has no effect on the output.



Exemples d'applications simples

Reconnaître une forme et l'afficher dans POB-Terminal
Afficher des images sur le POB-LCD

Affichage en temps réel des images sur le POB-LCD

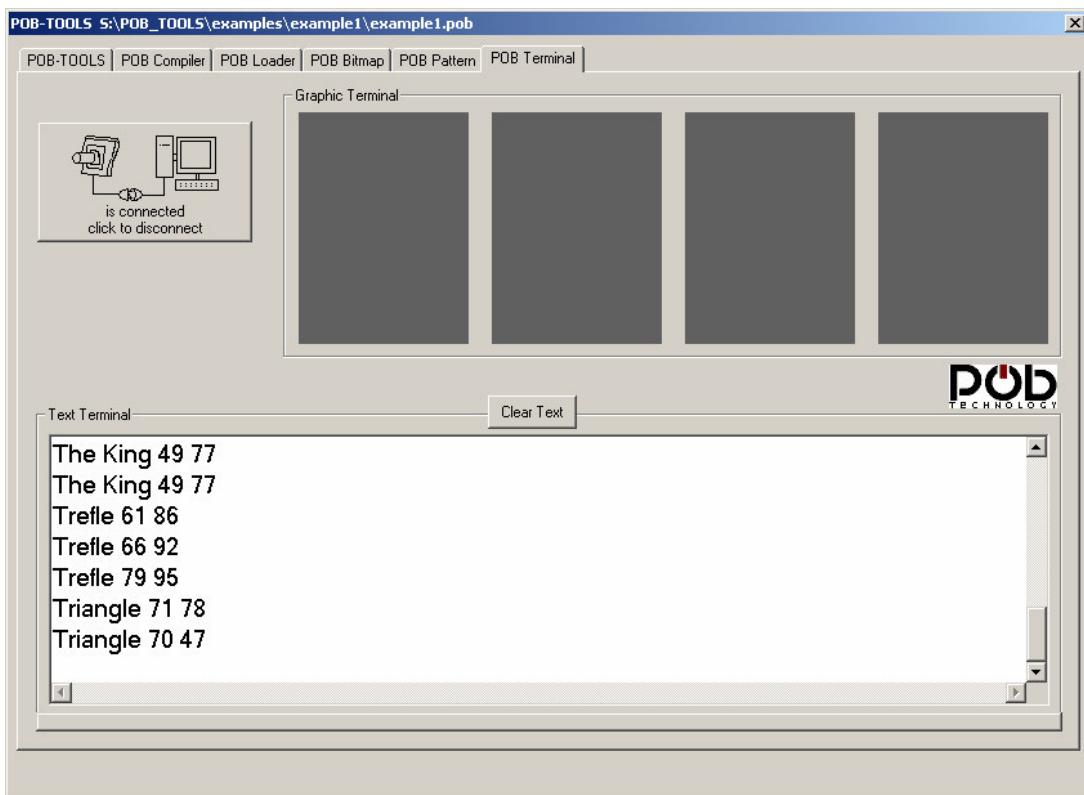
Rendez-vous sur le blog **DEVZone** de Pob pour
d'autres exemples et documents

www.pob-technology.com/blog

6 Exemples d'applications

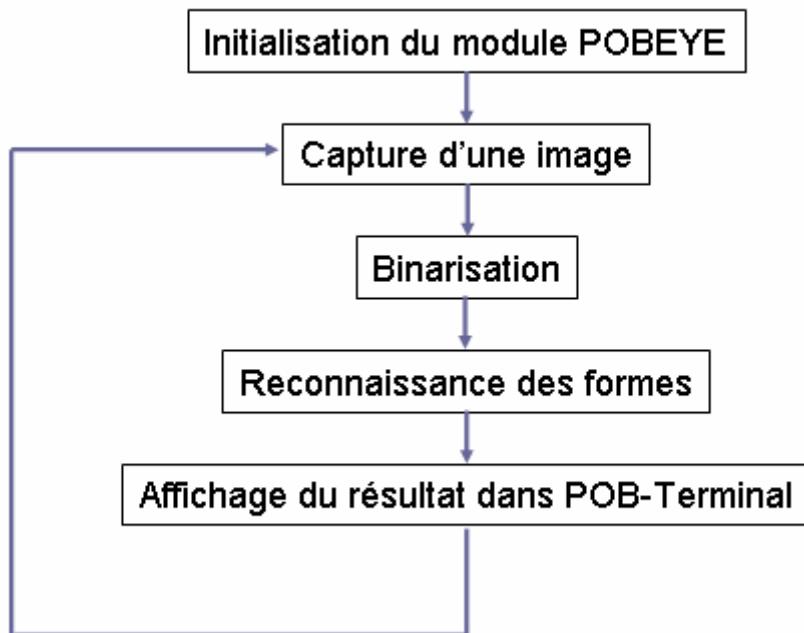
- *Reconnaitre une forme et l'afficher dans POB-Terminal*

Cet exemple se trouve dans le dossier « *example1* ». Pour ouvrir le projet dans POB-TOOLS, sélectionnez le fichier « *example1.pob* ».



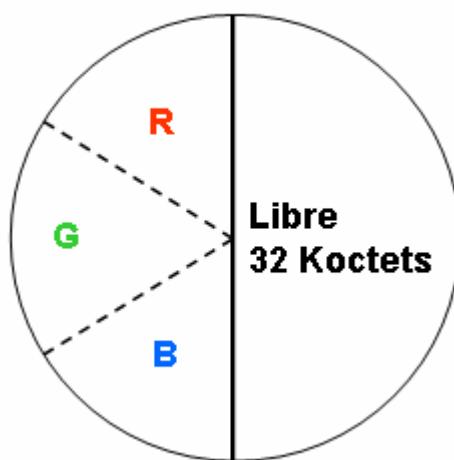
Le programme consiste à capturer une image, à identifier les formes présente dans l'image et à afficher le nom de la forme et les coordonnées dans le POB-Terminal. Ce programme est codé en langage C et utilise les fonctions du POB-TOOLS et du POB-EYE.

L'algorithme du programme :



Remarque :

Un espace de 32Koctets est alloué pour stocker les composantes RGB (88 pixels par 120 pixels = 10560 octets par composante). Lors des appels de fonction pour manipuler une image vidéo, les opérations se déroulent sur cet espace.



Code source du programme :

Un seul fichier est à inclure pour utiliser toutes les fonctions du POB-EYE (caméra, UART, debug sur POB-Terminal)

```
#include <pop-eye.h> /* En tête à placer pour tout les programmes utilisant le POB-EYE */
```

Le deuxième fichier à inclure est le dictionnaire de forme. Ce fichier se crée à l'aide de l'outil POB-Tools dans l'onglet POB-Pattern.

```
#include "pattern.h" /*Inclusion du dictionnaire de forme */
```

Début du programme :

```
int main (void)
{
    UInt8 i=0;

    UInt8 Nb_Identify=0; /* Variable correspondant au nombre de formes */
    Form ListOfForm[MAX_OF_FORM]; /* Tableau où seront stockés les formes reconnues */

    RGBFrame FrameFromCam; /* structure des composantes RGB de l'image */
```

Le module POB-EYE est initialisé par la fonction « **InitPOBEYE** ». Cette fonction initialise la caméra, l'UART et le bus I2C. Il faut obligatoirement appeler cette fonction pour un fonctionnement optimal correct du module.

```
InitPOB-EYE(); /* Pour utiliser le module POB-EYE, il faut l'initialiser par la fonction
InitPOB-EYE() */
```

Pour utiliser les fonctions de la caméra, une structure particulière doit être mise en place. Cette structure se nomme « *RGBFrame* » et contient l'adresse des composantes RGB d'une image. Pour initialiser cette structure, il faut appeler la fonction « **GetPointerOnRGBFrame** ». L'adresse de l'espace mémoire permettant de stocker les 3 composantes RGB ne changeant pas, cette fonction peut-être appelée qu'une seule fois.

```
GetPointerOnRGBFrame(&FrameFromCam); /*demande l'adresse des composantes
RGB et place le résultat dans FrameFromCam */
```

Après les initialisations, le programme peut se lancer :

```
while(1) /* Boucle principale du programme */
{
```

Première action à réaliser : capturer une image. La fonction « **GrabFrameRGB** » permet d'acquérir une image de la caméra. Cette image est stockée dans la zone pointée par la structure « *FrameFromCam* ».

GrabFrameRGB(); / on capture une image. La zone mémoire des composantes RGB est mise à jour */*

Pour reconnaître une forme, il faut binariser l'image couleur. Les trois composantes RGB seront identiques et vont être codé par 0 pour du blanc et 1 pour du noir après l'opération de binarisation.

BinaryRGBFrame(&FrameFromCam); / on binarise les composantes RGB */*

La fonction « **IdentifyForm** » permet d'identifier des formes dans une image. Elle prend pour paramètres : un pointeur vers l'image binarisé, un tableau de forme vide (« *ListOfForm* ») et le dictionnaire de forme. La fonction remplit le tableau de forme et renvoie le nombre de formes reconnues. A noter que le dictionnaire de forme « *Pattern* » est construit à partir de POB-PATTERN.

Nb_Identify=IdentifyForm(&FrameFromCam,ListOfForm,Pattern); / Identification des formes : la fonction renvoie le nombre de forme reconnue */*

Pour terminer, les informations du tableau « *ListOfForm* » sont extraites puis affichées sur le POB-Terminal. La fonction « **PrintTextOnPobTerminal** » permet d'afficher une ligne de texte dans le POB-Terminal.

Cette fonction a le même format que la fonction « *printf* » de la bibliothèque standard du langage C.

```

/* On parcourt le tableau ListOfForm jusqu'à atteindre Nb_Identify (le nombre de forme reconnues dans l'image par IdentifyForm) */
for( i=0 ; i < Nb_Identify ; i++ )
{
    /* Pour chaque forme, on récupérer son id */
    switch (ListOfForm[i].id)
    {
        /* et on affiche le résultat sur POB-Terminal en fonction de l'id de la forme */
        case IDP_0_CROSS:
            PrintTextOnPobTerminal("Cross %d %d",ListOfForm[i].x,ListOfForm[i].y);
            break;

        case IDP_1_BIGA:
            PrintTextOnPobTerminal("A Big A %d %d",ListOfForm[i].x,ListOfForm[i].y);
            break;

        case IDP_2_KING:
            PrintTextOnPobTerminal("The King %d %d",ListOfForm[i].x,ListOfForm[i].y);
            break;

        case IDP_3_TOWER:

```



Manuel Utilisateur du POB-Bot

```
PrintTextOnPobTerminal("Tower %d %d",ListOfForm[i].x,ListOfForm[i].y);
break;

case IDP_4_TREFLE:
PrintTextOnPobTerminal("Trefle %d %d",ListOfForm[i].x,ListOfForm[i].y);
break;

case IDP_5_TRIANGLE:
PrintTextOnPobTerminal("Triangle %d %d",ListOfForm[i].x,ListOfForm[i].y);
break;

case IDP_6_CIRCLE:
PrintTextOnPobTerminal("Circle %d %d",ListOfForm[i].x,ListOfForm[i].y);
break;

default:
break;
}
}
return 0;
}
```

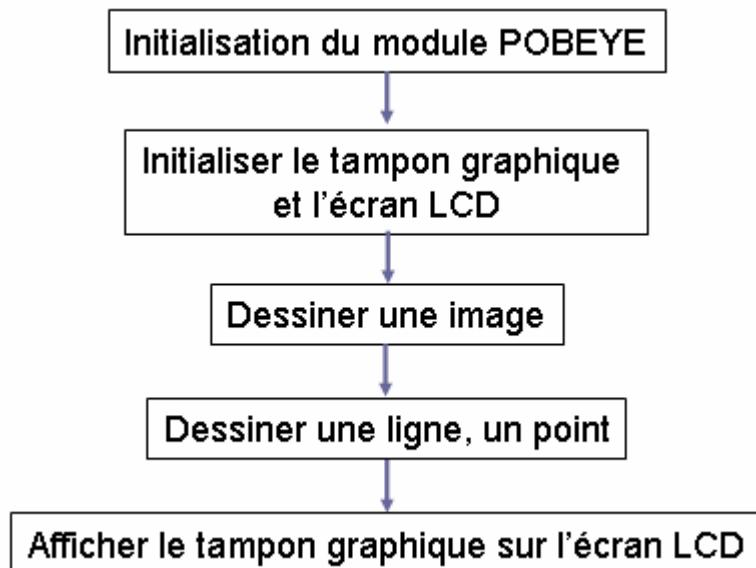
Manuel Utilisateur du POB-Bot**▪ Afficher des images dans POB-LCD**

Cet exemple se trouve dans le dossier « *example2* ». Pour ouvrir le projet dans POB-TOOLS, sélectionnez le fichier « *example2.pob* ».

Le programme affiche des images sur l'écran LCD et manipule les fonctions de dessin (tracé une ligne, affiché un point).



L'algorithme du programme :



Code source du programme :

Pour utiliser les fonctions du module POB-EYE et du module LCD, il faut inclure le fichier suivant :

```
#include <pob-eye.h> /* Fichier à inclure pour tout les programmes utilisant le POB-EYE */
```

Le programme affiche des ressources graphiques créée par POB-TOOLS. Il faut inclure le fichier des images :

```
#include "Bitmap.h" /* Inclusion des ressources graphiques produits par POB-BITMAP */
```

```
int main (void)
{
```

Pour accélérer l'affichage sur l'écran LCD, un tampon graphique est utilisé : toutes les opération d'affichage vont être réaliser sur ce tampon, ensuite le tampon sera transférer sur l'écran LCD.

Il est donc nécessaire de déclarer un tableau « *LCD_Buffer* » pour stocker les pixels. Pour l'application, un tableau de 128 par 64 utilisant 1 bit par pixel est utilisé. **Attention** il est impératif d'indiquer l'unité utilisée : dans ce cas des bits. Il faut utiliser un des deux symboles : *BITS* (pour un bit par pixel) ou *BYTES* (pour un octet par pixel).

```
UInt8 LCD_Buffer [LCD_WIDTH*LCD_HEIGHT*BITS]; /* Tableau de stockage des pixels de 128 par 64 pixels, 1 bit par pixel */
```

En plus du tampon, une structure est nécessaire pour stocker les informations sur la taille

Manuel Utilisateur du POB-Bot

de l'écran.

GraphicBuffer ScreenBuffer ; / structure du tampon graphique */*

Pour un fonctionnement correct du module POB-EYE et de l'écran LCD, il faut les initialiser :

InitPOB-EYE(); / Initialisation du module POB-EYE */
InitLCD(); /* Initialisation de l'écran LCD */*

Dernière initialisation, le tampon graphique doit être mis en place. La fonction « **InitGraphicBuffer** » initialise le tampon graphique à partir de la taille d'écran, du nombre de bit par pixel et l'emplacement du tableau de stockage (« *LCD_Buffer* »).

*InitGraphicBuffer(&ScreenBuffer, LCD_WIDTH,LCD_HEIGHT,**ONE_BIT**,LCD_Buffer); /* la structure ScreenBuffer est initialisée avec les valeurs de l'écran et l'emplacement du tampon */*

Pour le nombre de bit par pixel, il faut utiliser un des deux symboles : *ONE_BIT* (pour un bit par pixel) ou *EIGHT_BITS* (pour 1 octet par pixel).

Remarque : Pour déclarer le buffer graphique, il faut utiliser *BITS* ou *BYTES*. Pour la fonction *InitGraphicBuffer*, il faut utiliser *ONE_BIT* ou *EIGHT_BITS*.

ClearGraphicBuffer(&ScreenBuffer); / Avant de travailler sur le tampon, il est nécessaire de l'effacer */*

L'affichage d'image s'effectue par la fonction « **DrawBitmap** ». Cette fonction prend pour paramètres les coordonnées X Y, le numéro de l'image, le tableau des images et un pointeur vers la structure du tampon graphique.

A noter que le numéro et le tableau « *Bitmap* » sont produits par l'outil POB-Tools.

DrawBitmap(30,10,IDB_1,Bitmap,&ScreenBuffer); / image 1 est dessiné */
DrawBitmap(45,10,IDB_2,Bitmap,&ScreenBuffer); /* image 2 est dessiné */
DrawBitmap(65,10,IDB_3,Bitmap,&ScreenBuffer); /* image 3 est dessiné */*

L'affichage de ligne se fait par la fonction « **DrawLine** ». Elle prend pour paramètres : les coordonnées de départ et d'arrivée et le pointeur sur le tampon graphique.

DrawLine(10,10,25,30,&ScreenBuffer);

L'affichage de point se fait par la fonction « **PlotAPoint** ».



Constructeur de Robots
Pour les Loisirs Innovants et Pour l'Enseignement

Manuel Utilisateur du POB-Bot

PlotAPoint(20,10,&ScreenBuffer);

Enfin, on affiche le tampon graphique sur l'écran LCD. La fonction « **DrawLCD** » prend pour paramètre le pointeur sur le tampon graphique.

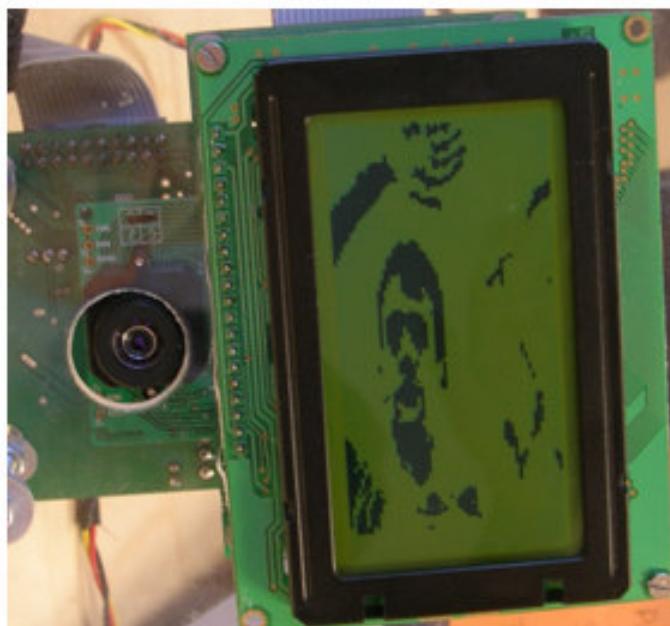
DrawLCD(&ScreenBuffer); / Affichage sur l'écran */*

return 0;

}

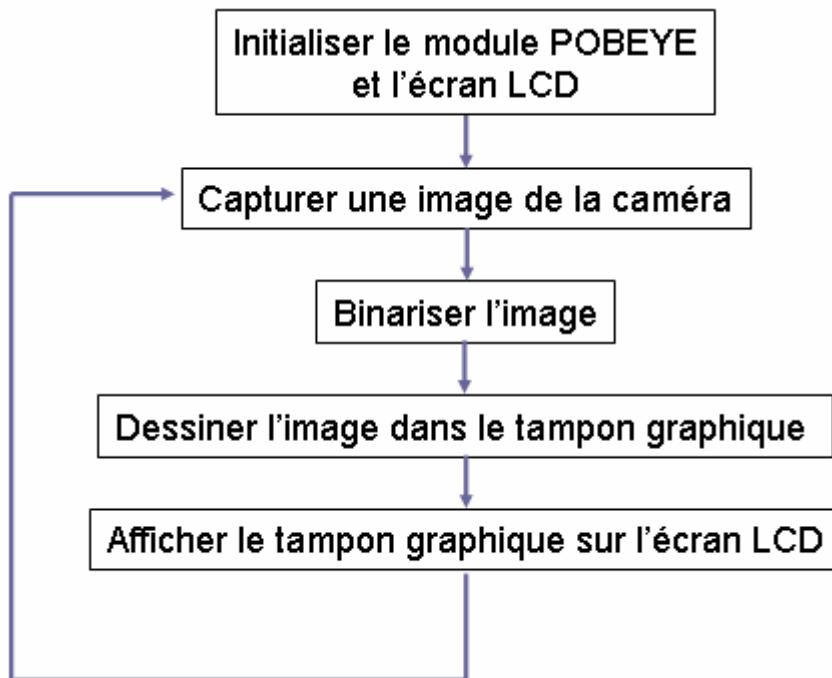
▪ *Affichage en temps réel des images sur le POB-LCD*

Le programme utilise le module POB-EYE et l'écran LCD pour afficher directement sur l'écran LCD l'image de la caméra.



Cet exemple se trouve dans le dossier « *example3* ». Pour ouvrir le projet dans POB-TOOLS, sélectionnez le fichier « *example3.pob* ».

L'algorithme du programme :



Code source du programme :

Pour utiliser les fonctions du module POB-EYE et de POB-LCD128, il faut inclure le fichier suivant :

```
#include <pob-eye.h> /* En tête à placer pour tout les programmes utilisant le POB-EYE */
```

Début du programme :

```
int main (void)
{
    Int16 i=0,j=0,k=0;
```

Pour afficher sur l'écran LCD, deux variables sont à mettre en place : un tableau pour stocker les pixels à afficher et une structure pour gérer l'affichage du tampon graphique.

Dans cette application, pour obtenir un affichage fluide, nous devons travailler sur un buffer graphique d'un octet par pixel (c.f chapitre sur le POB-LCD128).

```
UInt8 LCD_Buffer [LCD_WIDTH*LCD_HEIGHT*BYTES]; /* tableau de 128 par 64 pixels,  
1 octets par pixel pour stocker les pixels */
```

```
GraphicBuffer LCD_Screen; /* Tampon graphique */
```

```
RGBFrame FrameFromCam; /* Composante RGB de la camera */
```

```
InitPOB-EYE(); /* Initialisation du module POB-EYE */  
InitLCD(); /* Initialisation POB-LCD */
```

Le tampon graphique doit être initialisé avec les dimensions de l'écran et le tableau de stockage des pixels.

```
/* Initialisation tampon graphique : 128 par 64 pixel, 1 octet par pixel, LCD_Buffer */  
InitGraphicBuffer(&LCD_Screen,LCD_WIDTH,LCD_HEIGHT,EIGHT_BITS,LCD_Buffer);
```

```
ClearGraphicBuffer(&LCD_Screen); /* Mise à 0 du tampon graphique */
```

Enfin, l'adresse des composantes RGB de la caméra doit être initialisé.

```
GetPointerOnRGBFrame(&FrameFromCam); /* Récupère l'adresse des composantes  
RGB */
```

Boucle principale du programme :

```
while (1)  
{
```

On récupère une image de la camera :

```
GrabFrameRGB(); /* capture une image de la camera */
```

L'écran LCD étant noir et blanc, afficher l'image capturée, nous devons la binariser. Il ne faut pas oublier que, après la binarisation, les 3 composantes RGB deviennent l'image binarisée. C'est-à-dire que l'on obtient 3 fois l'image binarisée.

```
BinaryRGBFrame(&FrameFromCam); /* Binarise les composantes RGB */
```

Pour afficher l'image de la caméra binarisé sur le POB-LCD, on remplit le tableau de pixel avec une composante de la caméra.

A noter que l'image est binaire (blanc/noir) : les composantes sont donc toutes égales. Ici, nous prenons la composante rouge pour remplir le tableau mais on obtient le même affichage avec les autres composantes.

```
for (k=0,i=64;i;i--)
{
    for (j=0;j<120;j++,k++)
    {
        LCD_Screen.buffer[k] = FrameFromCam.red[i+(j*88)];
    }
    k+=8;
}
```

Finalement, on affiche sur l'écran LCD le résultat : c'est l'image capturé par la caméra.

```
DrawLCD(&LCD_Screen);
}

return 0;
}
```



Constructeur de Robots
Pour les Loisirs Innovants et Pour l'Enseignement

Manuel Utilisateur du POB-Bot

Contacter POB-Technology

POB-TECHNOLOGY
11, avenue Albert Einstein
69 100 VILLEURBANNE
FRANCE

Adresse web : www.pob-technology.com

Adresse mail : contact@pob-technology.com

Téléphone : +33 (0)4 72 43 02 36
Fax : +33 (0) 4 72 66 18 99