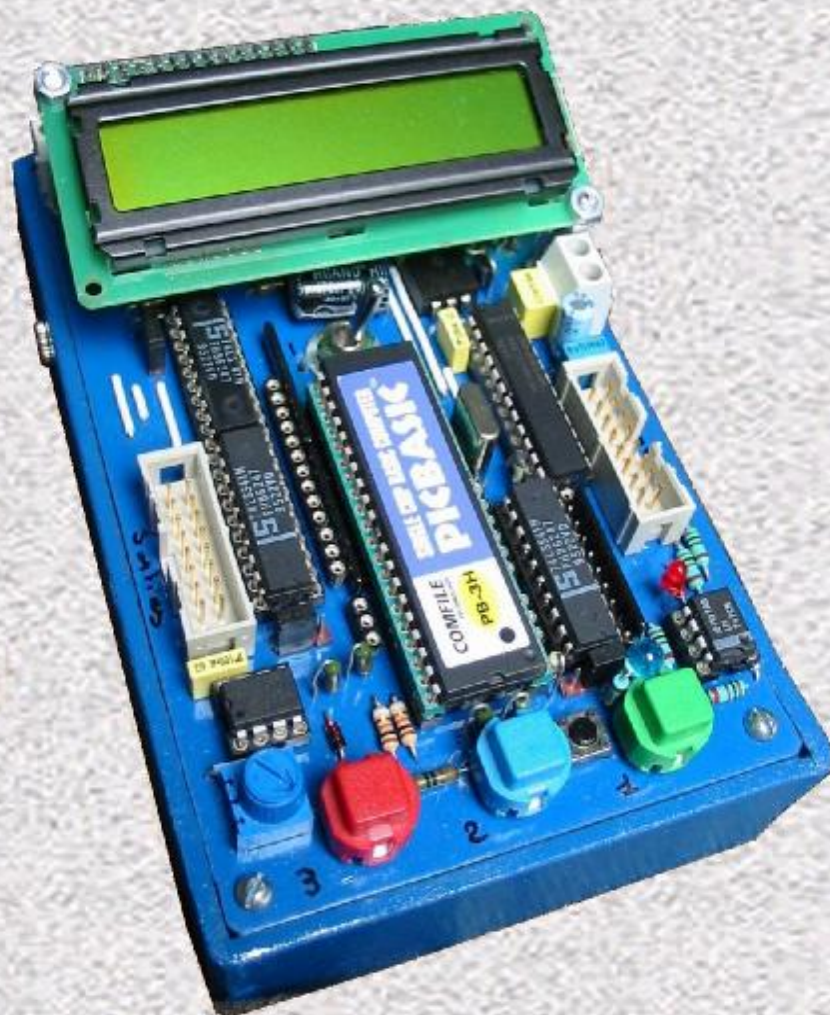


CARTE UNIVERSELLE
À PICBASIC



PRESENTATION

La carte de poche à PB3H est une carte microcontrôlée universelle pouvant réaliser aussi bien de la mesure embarquée que le pilotage d'un robot mobile. Elle est simple à réaliser et peu coûteuse, de plus elle se programme en Basic, langage à la portée de tous. Ses caractéristiques lui permettent de trouver sa place dans bon nombre de montage électroniques, de plus elle est un très bon outil d'apprentissage et d'expérimentation.

CARACTERISTIQUES :

- Microcontrôleur PicBasic type PB3H
- Platine simple face 80 x 135mm
- Ecran LCD 16x2 à liaison série
- 3 boutons de commande programmables
- Support pour EEPROM I2C (DIP8)
- Buzzer
- Prise de programmation par jack 3.5 (bootloader intégré au µC)
- Alimentation par accus 8.4v intégré avec prise de charge, indicateur de batterie faible par led (7.2v) et sortie 5v

- Prise SUB9 pour liaison RS232 avec PC
- Bus I2C

- 1 entrée analogique 12 bits sur potentiomètre
- 7 entrées analogiques +10v/-10v 12 bits par liaison série
- 1 entrée de comptage d'impulsion avec buffer jusqu'à (20 kHz)
- 8 entrées numériques avec buffers dont 7 reconfigurables en entrées analogiques 5v/10 bits par cavalier

- 2 sorties PWM à 1.22 kHz avec buffers
- 2 sortie sur relais 2RT (borniers)
- 9 sorties numériques (reconfigurables en entrées par cavaliers) avec buffers

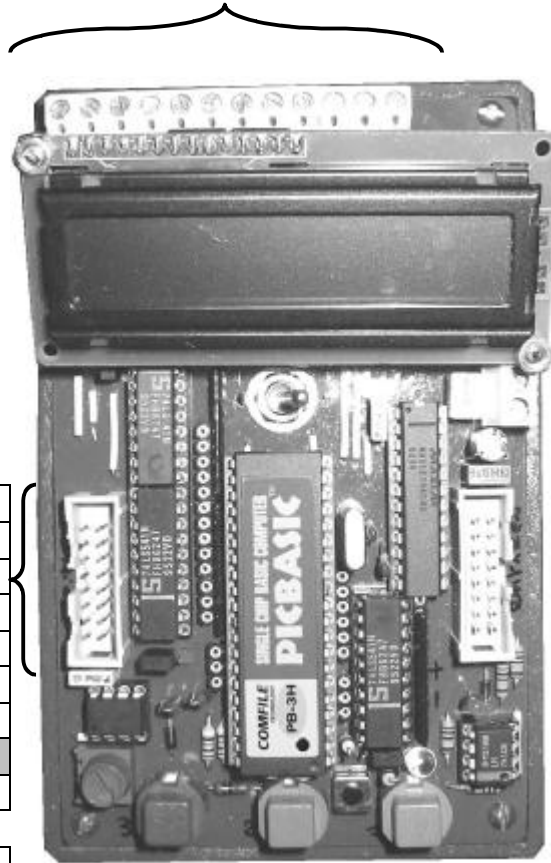
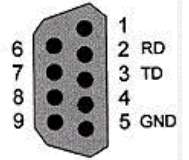
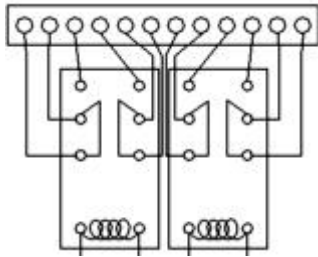
SOMMAIRE

Schéma fonctionnel	3
Assignation des broches de la carte	4
Le microcontrôleur	5
L'alimentation	6
Les trois touches programmable	7
Le buzzer	7
L'écran LCD 16 x 2	8
Le bus I2C	8
La mémoire EEPROM externe	9
Le CAN à liaison série MAX1270	10
La liaison RS232	13
Les entrées numériques	14
L'entrée de comptage d'impulsion	14
Les sorties numériques	15
Les sorties sur relais	15
Réalisation pratique (typon, implantation)	16
Exemple de programme	18
Listing du programme de test	19
Bibliographie	24

BROCHAGE DE LA CARTE

Relais

E2a R2a T2a E2b R2b T2b T1a R1a E1a T1b R1b E1b



Port RS232

Sorties

broches carte	
Scl	S1
Sda	S2
S7	S3
S8	S4
S9	S5
GND	S6
PWM 1	5V
PWM 2	

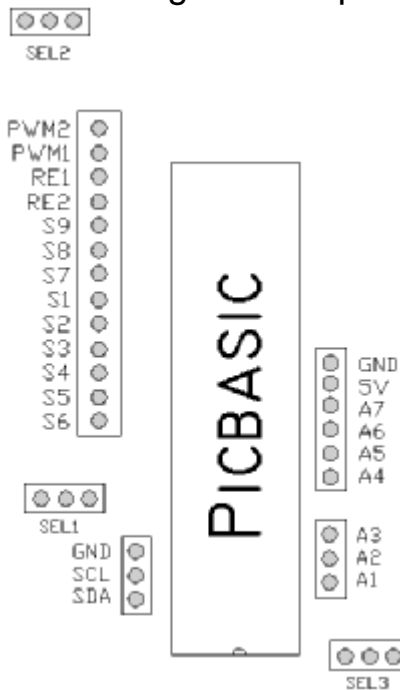
broches microcontrôleur	
IO25	IO11
IO26	IO20
IO14	IO21
IO13	IO22
IO12	IO23
GND	IO24
PWM 1/IO9	5V
PWM 2/IO10	

5V GND Entrées

broches carte	
A5	A6
A4	A7
A3	A2
A1	CLK
I5	I1
I6	I2
I7	I3
I8	I4

broches microcontrôleur	
A5	A6
A4	A7
A3	A2
A1	CLK
AIO7	AIO6
AIO5	AIO1
AIO4	AIO2
IO27	AIO3

Brochage des tulipes



LE MICROCONTROLEUR

Le microcontrôleur retenu pour cette carte est un PicBasic PB3H de la société coréenne Comfile Technologie disponible en France chez Lextronic. Ce μC est en fait un PIC 16F877 pré chargé d'un bootloader et d'un interpréteur Basic. J'ai choisit ce μC pour sa très grande facilité d'utilisation, la puissance de son langage et parce qu'un PIC 16F877 lui est parfaitement substituable, si on souhaite utiliser la carte avec un autre langage (assembleur, C).



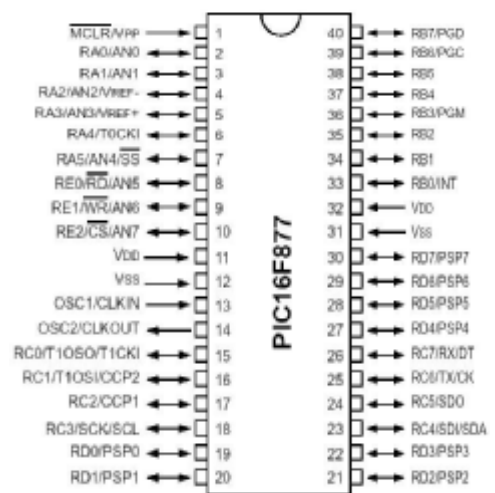
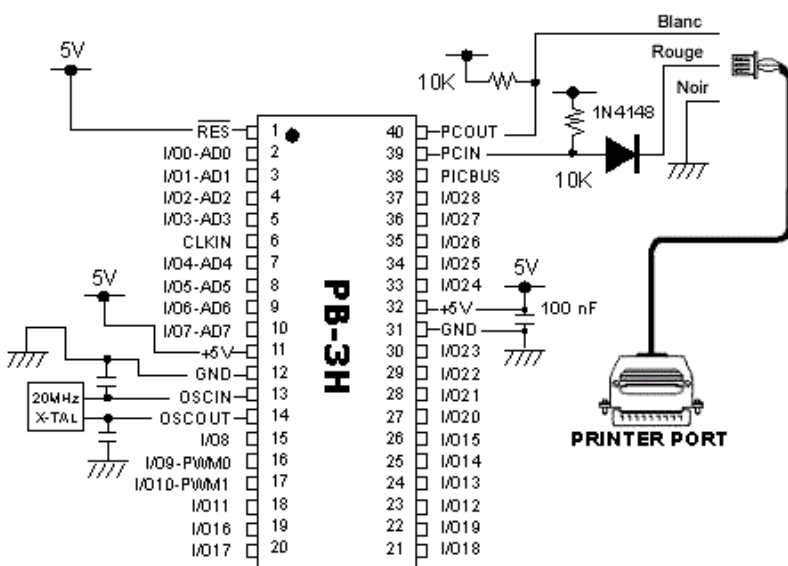
Par rapport à un 16F877 vierge le PB3H possède certaine restriction sur l'usage de ses broches. Ainsi les broches 39 et 40 sont réservées à la programmation grâce au bootloader et seront câblés comme le préconise le fabriquant et reliées à une prise jack 3.5mm stéréo plus solide et fiable à l'usage que le connecteur prévu d'origine. Ce montage évitera l'usage d'un programmeur et autorisera la programmation in-situ.

La broche 38 est quand à elle réservée à l'usage d'un écran LCD à liaison série. Comfile commercialise des interfaces série permettant d'utiliser n'importe quel écran alphanumérique standard. C'est ce choix qui à été fait et un écran 16 x 2 est donc relié à la sortie PICBUS via une telle interface.

La broche 6 est réservée pour faire du comptage d'impulsions (≤ 20 kHz). Elle sera reliée au connecteur des entrées via un buffer (74HCT541). Notons qu'au repos elle est tirée à la masse par une résistance.

Enfin la broche 1 est réservée au reset et sera donc tirée au 5v par une résistance de 10k et pourvue d'une touche de reset la reliant à la masse.

Toutes les autres broche du μC sont libres d'usage et sont conforme à celle du 16F877 d'origine.



(pour information)

L'assignation des broches est résumée dans le tableau suivant :

Broche	Désignation	Bloc	Fonction				
32	+5V		Alimentation Reset Quartz	19	I/O 16	2 (ST)	E/S
1	RES			20	I/O 17	2 (ST)	E/S
31	GND			21	I/O 18	2 (ST)	E/S
13/14	OSCIN/OSCOUT			22	I/O 19	2 (ST)	E/S
2	I/O 0-AD0	0 (TTL)	E/S ou CAN	27	I/O 20	2 (ST)	E/S
3	I/O 1-AD1	0 (TTL)	E/S ou CAN	28	I/O 21	2 (ST)	E/S
4	I/O 2-AD2	0 (TTL)	E/S ou CAN	29	I/O 22	2 (ST)	E/S
5	I/O 3-AD3	0 (TTL)	E/S ou CAN	30	I/O 23	2 (ST)	E/S
7	I/O 4-AD4	0 (TTL)	E/S ou CAN	33	I/O 24	2 (ST)	E/S
8	I/O 4-AD5	0 (TTL)	E/S ou CAN	34	I/O 25	2 (ST)	E/S
9	I/O 4-AD6	0 (TTL)	E/S ou CAN	35	I/O 26	2 (ST)	E/S
10	I/O 4-AD7	0 (TTL)	E/S ou CAN	36	I/O 27	2 (ST)	E/S
15	I/O 8	1 (ST)	E/S	37	I/O 28	2 (ST)	E/S
16	I/O 9-PWM0	1 (ST)	E/S ou PWM	6	CLKIN	(ST)	Entrée de comptage Cde afficheur série
17	I/O 10-PWM1	1 (ST)	E/S ou PWM	26	PICBUS		
18	I/O 11	1 (ST)	E/S	39/41	PCIN/PCOUT		Utilisés pour la programmation
23	I/O 12	1 (ST)	E/S				
24	I/O 13	1 (ST)	E/S				
25	I/O 14	1 (ST)	E/S				
26	I/O 15	1 (ST)	E/S				

(extrait de la documentation)

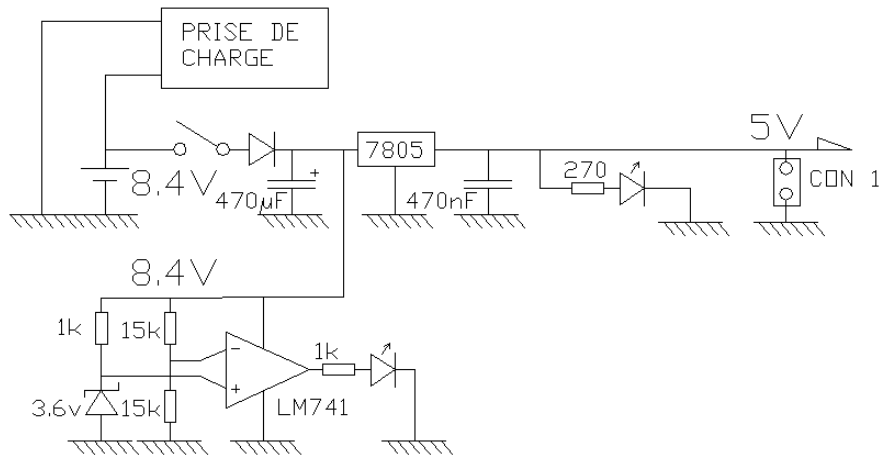
La programmation se fera grâce au logiciel fournis avec le composant et grâce au câble également fournis qui se connecte au port parallèle du PC. Pour l'édition des programmes et leur programmation dans le microcontrôleur consultez l'excellente documentation française réalisée par l'importateur : la société Lextronic (documentation disponible sur leur site internet.) Pour information le prix de ce microcontrôleur est de 44€ (prix début 2003). Le logiciel de programmation dispose également d'une fonction intéressante puisqu'on peut faire tourner le programme « pas à pas » dans le μ C en étant relié au PC et ainsi suivre à l'écran l'évolution des variables. Le débogage est ainsi très facile.

ALIMENTATION

L'alimentation de la carte sera assurée par un accumulateur rechargeable NiCad 8.4v 500 mAh disposant d'une prise de charge en face avant du boîtier. Ce choix est motivé par la grande disponibilité de ce type d'accus dans le commerce et son faible coût. La charge de l'accus se fera carte éteinte, par le biais d'un petit chargeur constitué d'un bloc secteur très ordinaire et d'un montage limiteur de courant à 50 mA, la charge dure 14 heure et procure à la carte une autonomie de plus de 10 heures puisque sa consommation est de 45 mA (sans périphériques).

La tension de 8.4v est régulée à 5v par un régulateur trois pattes type 7805 qui dispose bien des deux volts de marge de tension nécessaire à son bon fonctionnement. Cette tension est découplée par des condensateurs de diverse capacité (470 μ F et 470nF) qui assurerons une protection contre les parasites et pourvoions aux appels de courant éventuels. Une led bleue est montée en sortie de régulateur et fera office de témoin lumineux de mise sous tension. Enfin cette tension régulée sera disponible sur un bornier à vis afin de pouvoir alimenter les montage extérieur ou bien de permettre l'alimentation de la carte avec une source extérieur.

Notons enfin que le 5v est également présent sur le connecteur HE10 des sorties. (voir assignation des broches p 4)



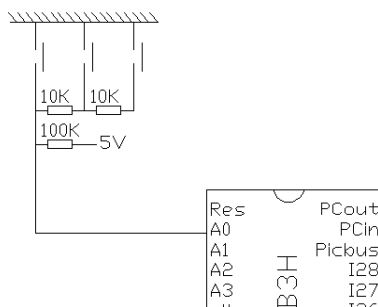
Un dispositif de surveillance de la tension d'alimentation est également présent pour prévenir l'utilisateur quand l'accus est vide. Ce dispositif est un classique AOP monté en comparateur et veillant à ce que la tension de l'accus ne chute pas en dessous d'une tension de référence fixées via une diode zener. Si tel est le cas l'AOP bascule et alimente une led rouge pour prévenir l'utilisateur que la batterie est vide. La tension de basculement est de 7.2v soit 1.03v par élément de l'accus.

LES TROIS TOUCHES PROGRAMMABLES

La carte dispose de trois touches qui sont reliées à une entrées analogique du µC (A0) en réalisant un pont diviseur de tension. Cette disposition est astucieuse car elle permet de relier ces trois touches à une unique entrée du µC. Ces touches seront les bienvenues pour interagir avec le programme et seront facilement intégrables dans celui-ci puisque cette possibilités à été prévue par le fabricant du µC et qu'une instruction Basic dédiée est prévue :

```
dim touche as byte
touche = adkeyin(port)
```

Cette instruction assure le traitement du signal reçu et renvoie une variable indiquant la touche pressée. Elle peut gérer jusqu'a 10 touches par entrées analogique mais j'ai jugé que trois suffisaient à l'usage prévu.

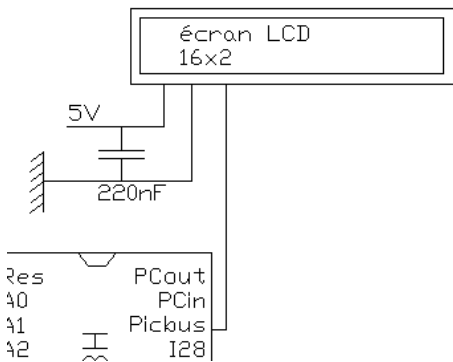


LE BUZZER

Un petit résonateur piezzo est relié à la broche I/O28 et sera très appréciable pour agrémenter les programmes de confirmations sonores. De plus une instruction Basic est prévue pour utiliser ce périphérique et programmer très facilement de petite "phrases musicales" donnant plus de vie à ce montage.

```
play 37, « ... »
```

L'ECRAN LCD



Un écran LCD alphanumérique de 2 lignes de 16 caractères est relié à la carte via une interface série (vendue par le fabricant). Cet écran est connecté au μ C sur son entrée spécialisée « picbus » par un seul fil. Un ensemble d'instruction spécialisée permet de le piloter très facilement.

La société Comfile Technologie, qui produit le Picbasic propose une gamme complète d'écran jusqu'à 4 x 20 substituable à celui-ci.

Voici un exemple d'utilisation de l'instruction `print` qui permet de gérer cet écran, pour plus de détail sur la programmation consultez le manuel du microcontrôleur.

```
set picbus high      'configure la liaison série à 19200bps
lcdinit             'initialisation
locate 0.0
print "  carte à PB3H  "
```

Il est également possible de piloter l'écran directement avec les instructions de liaison série pour diriger plus en détail le fonctionnement de l'afficheur et ainsi de réaliser des animations visuelles telle que : bargraphe, texte défilant etc.

Voici un exemple de programme de texte défilant :

```
const byte message=("http://alex.narbonne.free.fr")
10  for decal = 0 to 43          'boucle de décalage de message
    for X = 0 to 15            'boucle de position d'écran
        symbol=x+decal        'chargement du terme n+1
        lettre = message( symbol ) 'chargement du symbole
        busout &ha1;x;&h01;&ha2;lettre;&H00 'envoi à l'écran du
                                   symbole L à la position X
    next X
    play buz, "F9"             'au suivant
    delay 50                   'tempo
    play buz, "B9"
    next decal                 'au suivant !
goto 10
```

Le programme de test réalisé pour cette carte contient également un exemple de bargraphe.

LE BUS I2C

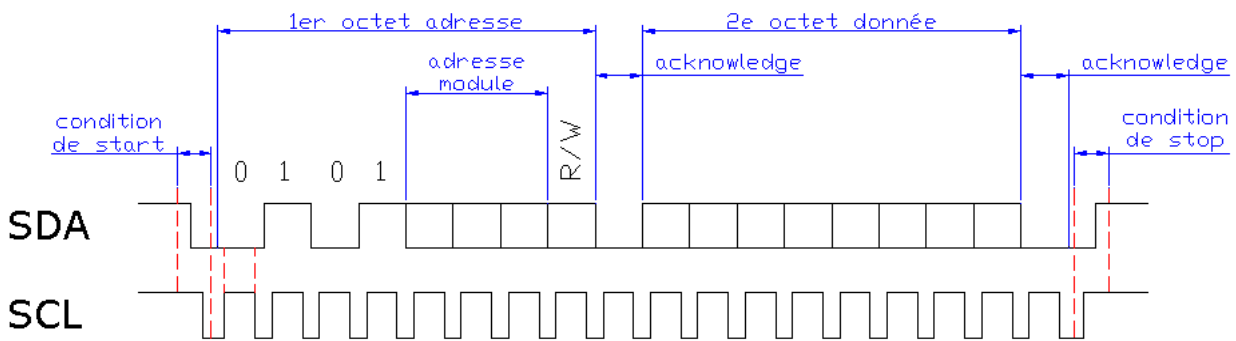
Je vais présenter brièvement le principe d'un tel bus, pour plus de détails consultez la bibliographie.

Le bus I2C a été développé par Philips dans les années 80 pour servir de bus universel entre composants électroniques. Il s'agit d'un bus bifilaire (Serial DAta et Serial CLock plus la masse) sur lequel est réalisé un dialogue sériel synchrone entre un maître (le composant qui prend le contrôle de la ligne) et un ou plusieurs esclaves (le ou les composant qui écoute le message).

Le protocole de communication est le suivant. Le maître prend le contrôle du bus en réalisant une "condition de start" c'est à dire qu'il passe la ligne SDA **et ensuite** SCL à l'état bas. Des lors les autres composants sont attentifs au bus et attendrons la "condition de stop" pour prendre le contrôle du bus s'il en ont besoin.

Une fois le contrôle pris, le maître envoie sur la ligne un premier octet qui est l'adresse du composant I2C à qui il s'adresse. Les quatre premiers bits sont fixés par le fabricant pour désigner le type de composant (0101 pour les mémoires par exemple), les trois bits restants sont laissés libres à l'utilisateur pour désigner son composant (on peut donc mettre 8 composants du même type sur un même bus I2C). Le huitième bit sert à préciser si on désire lire le composant ou y écrire.

Une fois cet octet envoyé le maître laisse la ligne SDA à l'état haut et l'esclave doit la forcer à l'état bas pour confirmer sa bonne réception, c'est l'acquiescement (acknowledge). Ensuite les octets suivants (donnée) peuvent être transmis, chacun devant être suivi d'un acquiescement. Quand le message est terminé le maître réalise la condition de stop en passant SCL **et ensuite** SDA à l'état haut, le bus est libre.



Cette carte est pourvue d'un bus I2C disponible sur le connecteur HE 10 des sorties et sur trois connecteurs « tulipes ». La ligne est réalisée par les broches I25 et I26 du µC. Une instruction Basic dédiée est prévue et permet d'utiliser facilement ce type de bus.

```
shiftout port1, port2, param, bit
shiftin port1, port2, param, bit
```

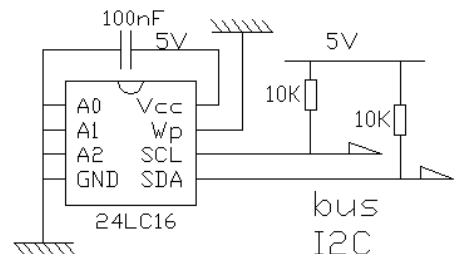
Le programme de test présent en fin de document comporte des routines de lecture /écriture qui pourront vous servir dans vos propres programmes.

L'intérêt d'un tel bus est simple à comprendre tant ses possibilités sont nombreuses. En effet on trouve toute sorte de composants compatibles I2C, des capteurs, des modules entrées/sorties (PCF8574), des carte de pilotage de servos (voir bibliographie).

Des interfaces pour PC existe aussi si on souhaite intégrer un PC au bus.

LA MEMOIRE EEPROM EXTERNE

La carte dispose d'une EEPROM externe à liaison série 24LC16 d'une capacité de 2k octets. Elle est câblée sur le bus I2C et est montée sur un support tulipe afin de pouvoir être changée facilement par un autre modèle. Son adresse est 000. pour plus de détails concernant le protocole de communication avec les eeprom voir en bibliographie. Cette mémoire permettra de stocker des mesures, des données etc..



Les routines suivantes pourront être utilisées pour piloter ce composant.

```

écriture:
    gosub start
    shiftout scl,sda,2,&b10100000
    shiftout scl,sda,2,adresse
    shiftout scl,sda,2,donnee
    gosub stop
    return

lecture:
    gosub start
    shiftout scl,sda,2,&b10100000
    shiftout scl,sda,2,adresse
    gosub start
    shiftout scl,sda,2,&b10100001
    donnee=shiftin(scl,sda,1)
    gosub stop
    return

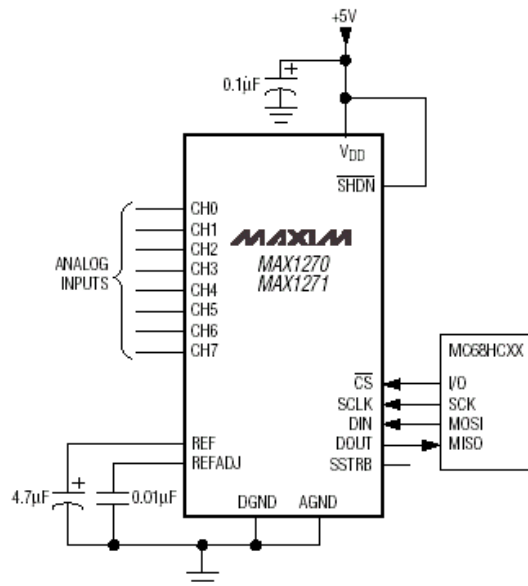
start:
    out scl,1
    out sda,1
    out sda,0
    out scl,0
    return

stop:
    out sda,0
    out scl,1
    out sda,1
    return
    
```

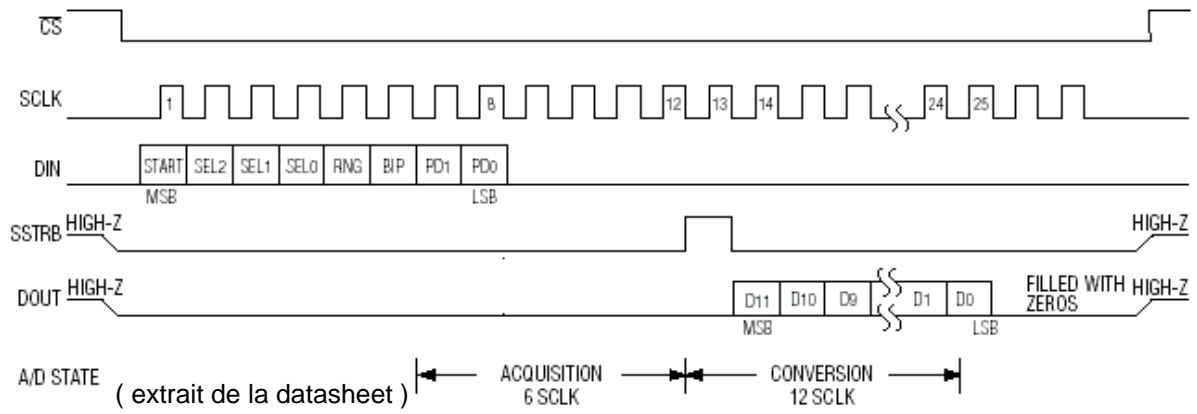
LE MODULE D'AQUISITION ANALOGIQUE MAX1270

Les conversions analogique/numériques sont réalisées par un composant spécifique, le MAX1270. Ce composant est un octuple CAN 12 bits à liaison série (type SPI). Les mesures peuvent se faire sur 4 échelles différentes. 0/5v 0/10v $\pm 5v$ $\pm 10v$. De plus il est protégé des surtensions jusqu'à $\pm 16.5v$.

Il est relié au μC via quatre lignes SCLK / \overline{CS} / Din / Dout (Sstrb est inutilisé) et se commande selon protocole illustré sur le chronogramme ci-contre.



- s'assurer que les lignes Sclk, Din, Dout sont à l'état bas (elles sont à l'état haut au repos)
- passer \overline{CS} à l'état bas pour activer le composant.
- générer une clock sur SCLK
- envoyer sur la ligne Din un octet de commande pour préciser au MAX1270 l'entrée à convertir et l'amplitude de cette mesure
- lire le résultat sur la ligne Dout
- couper la clock
- passer \overline{CS} à l'état haut pour désactiver le composant.



Le résultat de la mesure est ensuite émis sur la broche Dout sous la forme d'un mot de 12 bits.

L'octet de commande émis sur Din déclenche la mesure et permet de la paramétrer. Par ailleurs il permet de contrôler le CAN en choisissant son mode de fonctionnement. Les tableaux ci-dessous décrivent en détail la composition de cet octet.

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
start	sel 2	sel 1	sel 0	range	bip	pd 1	pd 0

bit	nom	fonction
7	start	1 logique indiquant le début de l'octet
6,5,4	sel 2,1,0	3 bits de sélection de l'entrée à convertir (table 1)
3	range	bit de sélection de l'échelle de mesure (table 2)
2	bip	bit de sélection de la polarité de la mesure (table 2)
1	pd 1,0	2 bits de sélection du mode de fonctionnement (table 3)

sel 2	sel 1	sel 0	entrée
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

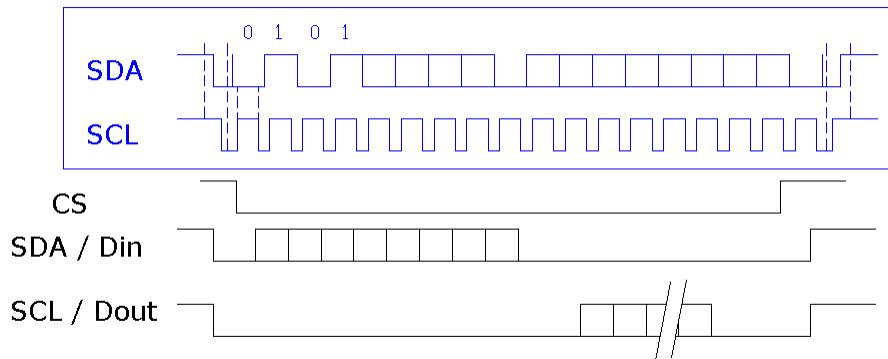
range	bip	échelle
0	0	0 à 5 V
1	0	0 à 10 V
0	1	± 5 V
1	1	± 10 V

pd 1	pd 0	fonction
0	0	clock interne / fonctionnement normal
0	1	clock externe / fonctionnement normal (cas de notre montage)
1	0	mode stand by
1	1	mode full power down

Le câblage de ce composant est particulier dans ce montage. En effet il est monté de façon peu conventionnelle puisqu'il partage des lignes de données avec le bus I2C bien qu'il fonctionne selon un protocole différent. En fait en utilisation normale, jamais ce composant ne sera utilisé simultanément avec le bus I2C (il faudrait 2 maîtres). En conséquence et afin de mobiliser le moins de broches possible la ligne Din est commune avec la ligne Sda du bus I2C et la ligne Dout commune avec la ligne Scl du bus I2C. Ce partage ne pose pas de problème particulier comme l'illustre le diagramme ci-contre. (Le fonctionnement de l'I2C est rappelé en encadré.)

Quand le bus I2C est actif le MAX1270 est rendu inactif par l'état haut de son entrée \overline{CS} , toutes ses broches de liaison série sont en haute impédance il ne perturbe donc pas le bus I2C et est transparent à l'utilisateur.

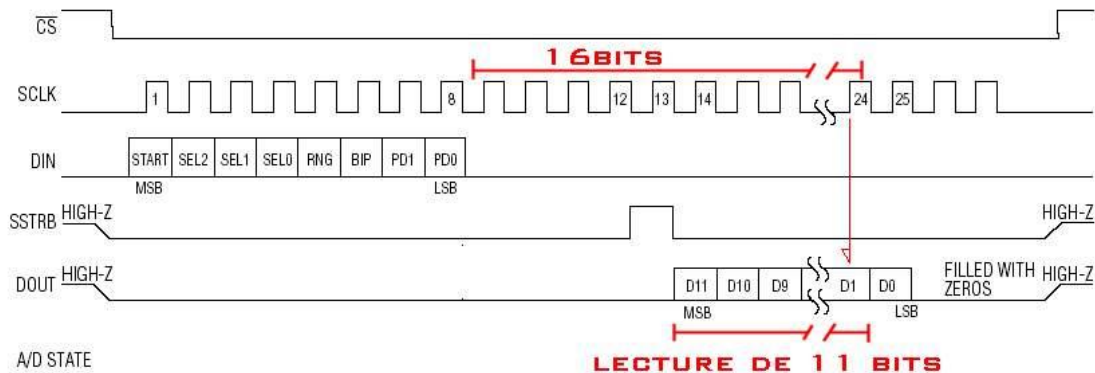
Quand c'est le MAX1270 qui travaille le chronogramme montre bien que la ligne Din ne change jamais d'état pendant que Dout est à l'état haut, dès lors les conditions de start (et de stop) ne sont jamais réunies et les composant I2C restent inactifs. Combien même un composant I2C penserait percevoir une condition de start il ne recevrait aucun message puisque quand Din (Sda) transmet un octet Dout (Scl) est au repos et inversement. De fait, les essais avec cette configuration ont confirmé ce raisonnement et aucun problème n'a été rencontré à ce jour.



Voici un exemple de routine de dialogue avec ce composant.

```
max: out csb,0
      shiftout scl,din,1,octet de commande
      resultat=shiftin(scl,dout,1,16)
      out csb,1
      return
```

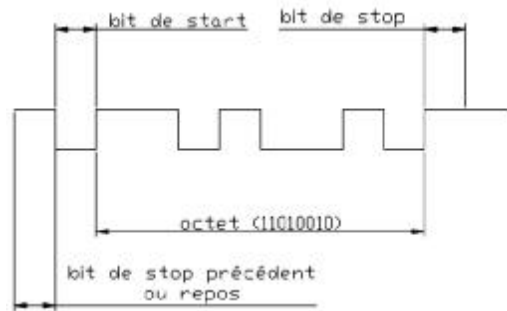
L'étude du chronogramme ci-dessous nous montre que la lecture du résultat qui se fait sur 16 bits ne permet de lire que les 11 premiers bits du résultats car celui-ci apparaît après un délai de 5 clock. Ce retard ne serait pas facile à prendre en compte dans la programmation et j'ai jugé qu'une résolution de 2048 pas était suffisante pour une grande majorité de cas.



Notons enfin que l'entrée ch0 du convertisseur est attaquée par un potentiomètre de 20k qui permet de disposer sur la carte d'une tension analogique modifiable manuellement. Cet accessoire est très appréciable et permet de programmer des applications réglables manuellement (variateur de vitesse, variation de vitesse d'échantillonnage, variation de la vitesse de défilement d'un chenillard etc.)

LA LIAISON RS232

Rappelons que les liaisons RS232 sont des liaisons asynchrones très utilisées en informatique. Elle nécessite que l'émetteur et le récepteur soit informé de la vitesse choisie pour le transfert. Dès lors, chaque octet transmis est encadré d'un bit de start et d'un bit de stop qui en précise le début et la fin. Puisque le récepteur connaît la vitesse du transfert il peut se passer de signal de synchronisation.



Trois lignes sont nécessaire à cette liaison.

TD (transmission de donnée), RD (réception de donnée) et la masse.

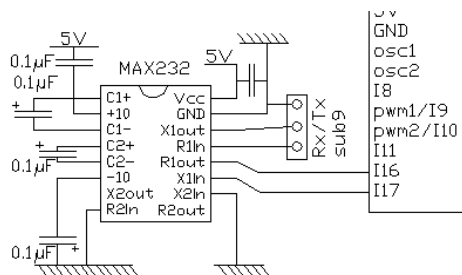
Les vitesses de transfert possible avec le PB3H sont de :

4800 bps (600 octets/s)

9600bps (1.2 ko/s)

19200bps (2.4 ko/s)

Pour plus d'information sur les liaison asynchrone consultez la bibliographie.



Grâce à cette liaison la carte peut devenir une sonde de mesure ou bien servir d'interface entre un PC et un montage extérieur. Afin d'adapter les signaux TTL du μ C au standard RS232 un MAX232 est monté de façon classique. Les lignes RX, TX et la masse sont disponible sur un connecteur SUB9 mâle qui permet ainsi de relier la carte au PC avec un simple câble série.

Une instruction Basic est là encore prévue pour piloter très facilement ce port

```
serout port, param1, mode,param2,[var1]
serin port, param1, mode,param2, adress,[var1]
```

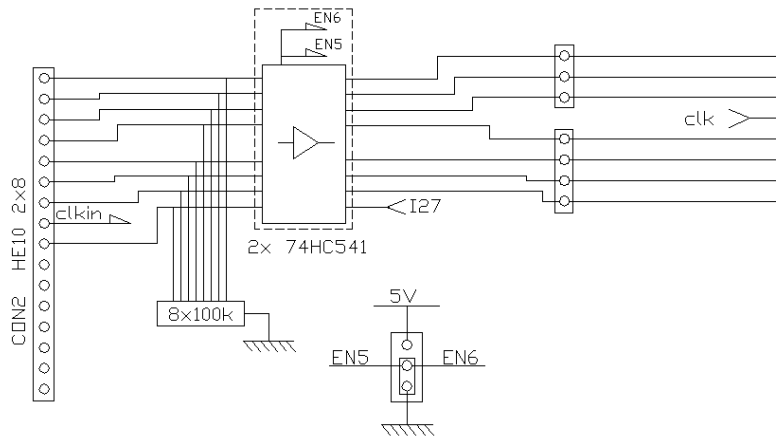
Voici un exemple de programme qui envoie à un PC munis d'un terminal adéquat le message « INSA Strasbourg » lorsque l'on presse une touche.

vitesse 9600bps ; pas de parité ; 1 bits de stop

```
10 dim touche as byte
   touche = adkeyin(port)
   if touche=1 then
     serout 16,103,0,1,["INSA Strasbourg"]
     delay 1000
   end if
   goto 10
```

LES ENTREES NUMERIQUES

La carte dispose de 8 entrées numériques avec buffers (74HCT541). Ces buffers non inverseurs possèdent des triggers, la tension minimum des états haut est de 2V et la tension maximum des états bas est 0.8V. Ces entrées numériques sont tirées à la masse par un réseau de résistance de 100k. Elles sont toutes disponibles sur le connecteur HE10 des entrées. Le réseau de résistance est monté sur un support tulipe, peut ainsi changer la tension de tirage si nécessaire.



Notons enfin que 7 d'entre-elles sont utilisables en entrées analogiques 5v/10bits après avoir passé les buffers en haute impédance grâce au cavalier prévu. Ceci fait, les entrées analogiques sont accessibles via les supports tulipes prévus à cet usage, il conviendra d'être particulièrement vigilant sur la nature des signaux appliqués sur ces broches. En effet elles sont alors reliées au μ C sans aucune interface et ne supportent pas de tension supérieure à 5v. Si une telle tension devait être appliquée il conviendrait de réaliser un pont diviseur de tension afin qu'elle ne dépasse jamais 5v.

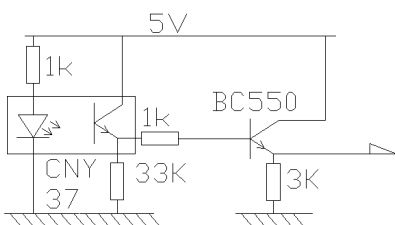
La mesure de tension analogique devra de préférence ce faire grâce au MAX1270 cité précédemment qui est protégés des surtensions et qui préserve le μ C des erreurs de manipulation.

L'ENTREE DE COMPTAGE D'IMPULSION

La carte possède une entrée spécialisée dans le comptage d'impulsion. Les signaux acceptés auront une fréquence maximum de 20kHz et seront comptabilisés sur front montant, cette entrée est elle aussi suivie d'un buffer avec trigger. Cette entrée est particulièrement utile pour comptabiliser les impulsions gérée par un montage externe comme par exemple une fourche optique sur un dispositif odomètre.

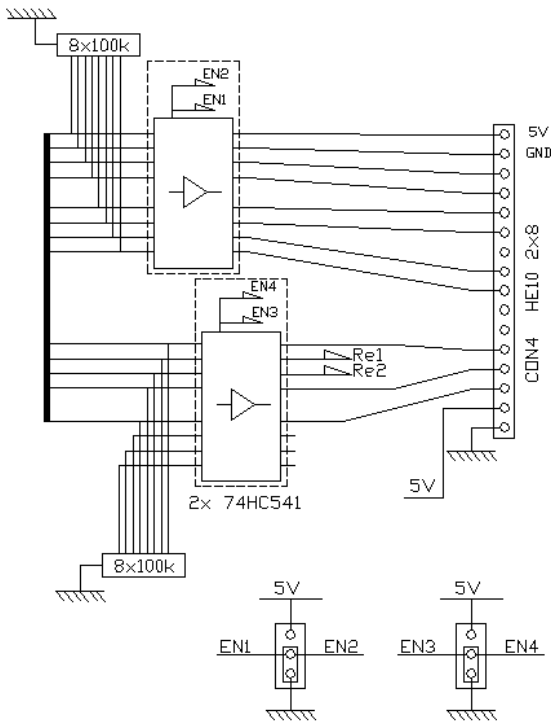
L'instruction dédiée à l'usage de cette entrée sera appelé quand on le souhaite dans le programme puisque le comptage s'effectue en « tâche de fond ». La valeur du compteur est accessible en rappelant la fonction `count`. notons enfin que cette fonction est gérée sur 16 bits (cad 65536 impulsions) et qu'au delà elle repasse à 0.

```
dim compt as byte
compt = count(0)
```



On peut utiliser le montage ci-contre pour générer les impulsions à comptabiliser. Il utilise une fourche optique CNY37 et un transistor BC550 pour l'interfaçage.

LES SORTIES NUMERIQUES

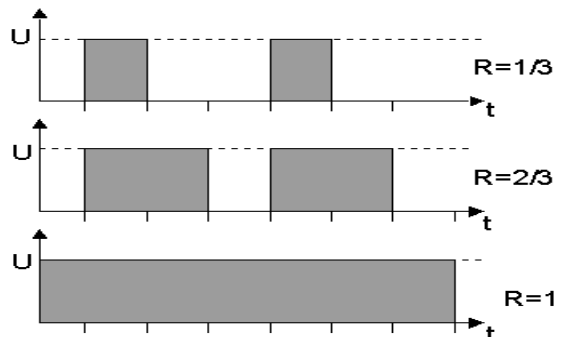


La carte dispose de 11 sorties numériques avec buffers capable de supporter -15mA à l'état haut et 20mA à l'état bas. Ces 11 lignes sont accessibles via le connecteur HE10 des sorties. Je rappelle que ces signaux sont de type TTL et qu'en conséquence ils doivent être interfacés pour pouvoir débiter un courant plus important. Consultez la bibliographie pour plus de détails sur les interfaces de puissance.

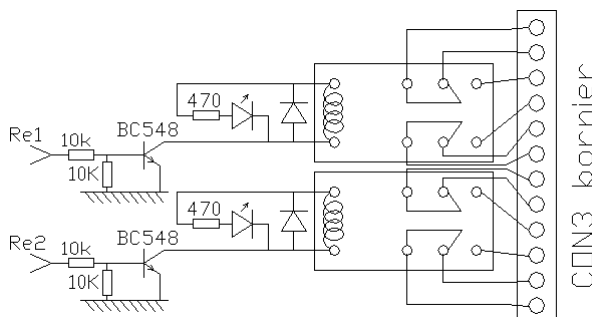
Deux cavaliers sont prévus pour pouvoir passer les sorties en haute impédance et ainsi isoler le μC de ses périphériques. Des connecteurs tulipes sont également prévus entre les buffers et le microcontrôleur, il est donc possible d'utiliser les sorties comme entrées si nécessaire (avec les précautions de rigueur) en se repiquant à cet endroit.

Deux de ces lignes de sortie peuvent produire des signaux PWM à 1.22kHz qui seront utilisés pour alimenter des moteurs via un étage de puissance adapté. Rappelons que la PWM est un signal qui ne connaît que deux états, haut et bas. C'est la différence de temps passé entre l'état haut et bas qui détermine la vitesse du moteur. Ce rapport est appelé rapport cyclique. Dans le langage Picbasic ce rapport cyclique est codé sur 8 bits. L'instruction de pilotage des PWM est des plus simple à utiliser :

```
pwm, port, val
```



LES SORTIES SUR RELAIS



2 sorties sur relais 2 RT sont disponibles sur un bornier à vis. Les caractéristiques des relais dépendent bien sur des modèles choisis mais les modèles courant pour CI ont les spécificités suivantes :

1.25 A / 125 VAC 2 A / 30 VDC délai de réaction < 5ms

Les relais sont piloté via un transistor de commande et leur état est visualisable par une

led, précieuse lors de la programmation.

Ne pas omettre la diode « de roue libre » montée en inverse au borne du relais sous peine de destruction du transistor.

REALISATION PRATIQUE DE LA CARTE

La carte est réalisée en époxy simple face. Le circuit est réalisé de façon classique mais avec soin en raison de la densité importante de piste. Après gravure le circuit pourra être étamé afin de le préserver de l'oxydation. L'ensemble des trous sera percé à 0.8mm et les trous des composants ayant des broches plus importantes (diodes, inter, prise sub9 etc) seront percés à 1mm. Enfin nous percerons les trous de fixation de la carte, de l'écran LCD et de passage de l'interrupteur. J'ai également passé une couche de peinture en bombe sur le coté composant du circuit pour en améliorer l'aspect.

La soudure débutera avec les straps très nombreux sur cette carte, ensuite les résistances, les supports de CI et ainsi de suite par ordre croissant de taille.

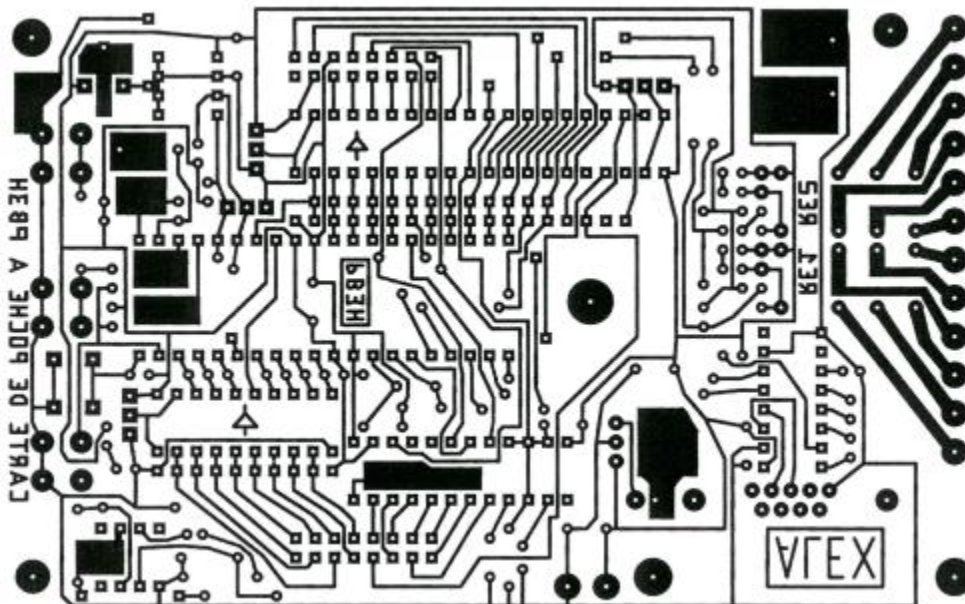
Plusieurs éléments sont à connecter au circuit par soudure de leurs fils directement sur les zones cuivrées prévues à cet effet. (voir figure d'implantation)

L'écran LCD sera fixé grâce au deux trous prévus via deux tiges filetée M3 misent en forme.

Avant de mettre en place les CI on procèdera aux contrôles d'usage en vérifiant que les tension en divers points du montage sont bien conformes au schéma de principe. Enfin on pourra mettre en place les différents CI et charger un premier programme.

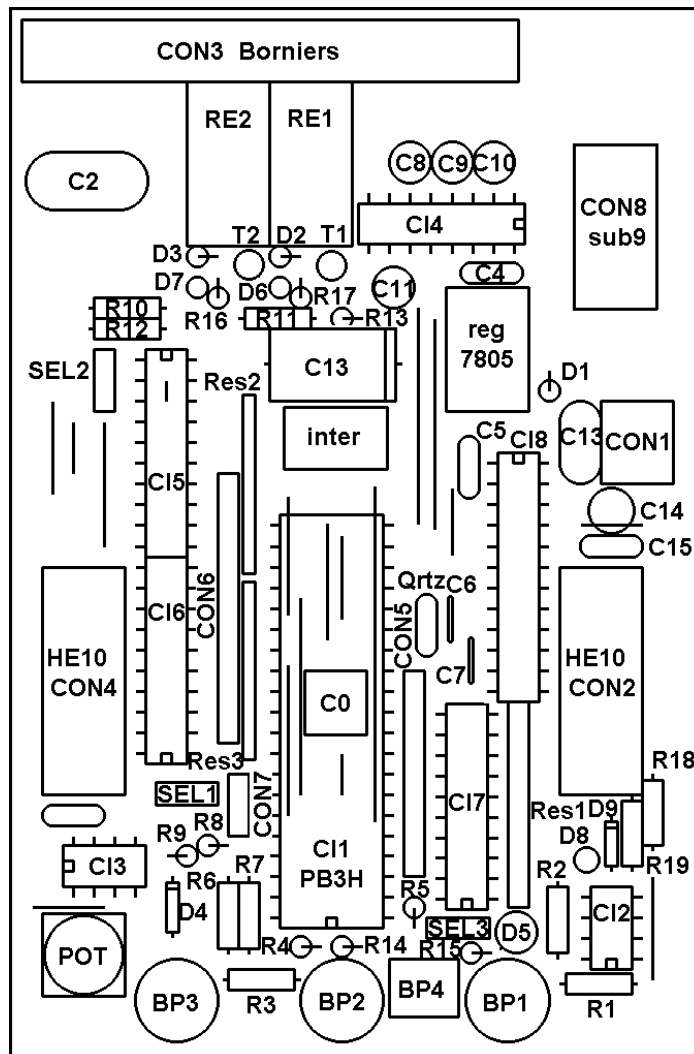
Pour finir la carte sera montée dans un boîtier adapté qui renfermera tous les éléments (buzzer, accus, connections diverses, prise jack de charge et de programmation) et qui participera à finir proprement ce montage.

Voici le typon de la carte à l'échelle 1 :



R1-R13	10kΩ	PROG	prise jack stéréo
R14	100kΩ	D1-D3	1N4001
R15	270Ω	D4	1N4148
R16-R17	470Ω	D5-D8	del
R18-R19	1K	D9	zener 3,6V
Res1-Res2	8 x 100kΩ	CI1	PB3H
C0-C5	100nF	CI2	LM741
C6-C7	22pF	CI3	24LC16
C8-C12	0,1μF tantale	CI4	max232
C13	470μF	CI5-CI7	74HC541
C14	4,7μF	CI8	max1270
C15	10nF	Pot1	20kΩ
CON1	bornier x2	RE1-RE2	Relais 2RT
CON2	HE10 2x8	BUZZER	buzzer
CON3	bornier x12	BP1-BP4	boutons poussoirs
CON4	HE10 2x8	SEL1-SEL3	cavalier 2 positions
CON5	tulipe x7	REG	7805
CON6	tulipe x13	T1-T2	BC548
CON7	tulipe x3	QRTZ	quartz 20Mhz
CON8	SUB 9		écran LCD 16x2 série
CHARGE	prise jack mono		support DIL

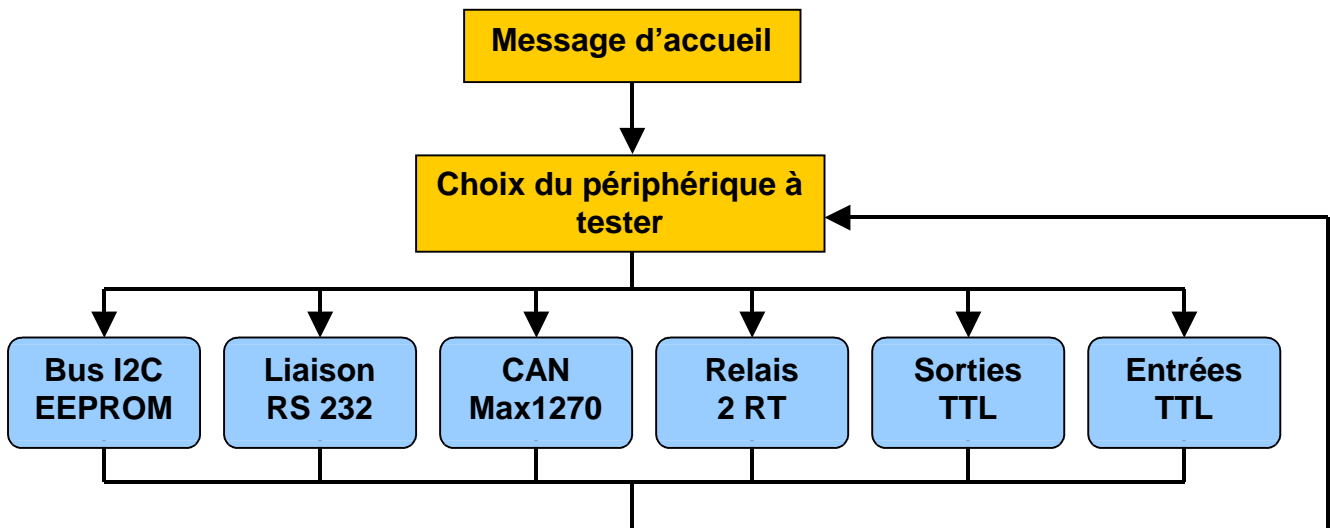
Et voici le schéma d'implantation des composants :



PROGRAMME DE TEST

Je propose ici un programme de test qui permet de vérifier l'ensemble des fonctions de la carte et de s'assurer de leur bon fonctionnement.

Le programme propose à l'utilisateur via un menu déroulant manipulable par les trois touches les différentes fonctions de la carte. Une fois la fonction à tester sélectionnée le programme de test débute et se termine par un rapport de fonctionnement que l'on quitte pour retourner au menu de départ.



Les routines de fin de programme pourront être facilement réutilisées pour d'autres applications, à cette fin le programme est disponible en téléchargement à cette adresse :

<http://alex.narbonne.free.fr>

```
'=====
'====  C A R T E   D E   P O C H E   ====
'====  A P I C B A S I C   P B 3 H   ====
'=====
'=====  Alexandre Narbonne  =====
'=====
'====  GESTION D'UNE CARTE UNIVERSELLE  ====
'====  A BASE DU MICROCONTROLEUR PICBASIC3H  ====
'=====
```

```
'=====
'===== ALLOCATION DES BROCHES =====
'=====
```

```
CONST clav=0
CONST e2=1
CONST e3=2
CONST e4=3
CONST e7=4
CONST e6=5
CONST e1=6
CONST e5=7
CONST CS=8
CONST pwm1=9
CONST pwm2=10
CONST s1=11
CONST s9=12
CONST s8=13
CONST s7=14
CONST clock=15
CONST rsout=16
CONST rsin=17
CONST re1=18
CONST re2=19
CONST s2=20
CONST s3=21
CONST s4=22
CONST s5=23
CONST e8=24
CONST s6=25
CONST scl=26
CONST dout=26
CONST sda=27
CONST din=27
CONST buz=28
```

```
DIM touche AS BYTE
SET PICBUS HIGH
LCDINIT
```

```
touche=0
DIM I AS integer
LOCATE 0,0 : PRINT "carte a PICBASIC"
LOCATE 0,1 : PRINT "2003 A.Narbonne"
DELAY 2000
LOCATE 0,0 : PRINT " carte autonome "
LOCATE 0,1 : PRINT " programmable "
delay 1000
csroff
play buz, "+E8+D8+C8"
play buz, "+G8+D8+F8"
csroff
I=0
```

```
'=====
'===== VARIABLES ET AUTRES =====
'=====
```

```
dim X as byte
dim DECAL as byte
dim SYMBOL as byte
dim LETTRE as byte
dim adresse as byte
dim donnee as byte
dim rslt as integer
dim entier as byte
dim reste as integer
dim W as byte
DIM Char as BYTE ' Le caractère à dessiner
DIM Pos as BYTE ' L'emplacement du caractère à dessiner
DIM A as BYTE ' 3 variables intermédiaires nécessaires
```

```

DIM B as BYTE          ' car le compilateur n'accepte pas
DIM C as BYTE          ' les opérations trop complexes

out cs,1               'désactivation par défaut du CAN

                        ' Création des caractères spéciaux du bargraphe
BUSOUT &HA5,1,16,16,16,16,16,16,16,16 'I
BUSOUT &HA5,2,24,24,24,24,24,24,24,24 'II
BUSOUT &HA5,3,28,28,28,28,28,28,28,28 'III
BUSOUT &HA5,4,30,30,30,30,30,30,30,30 'IIII
BUSOUT &HA5,5,31,31,31,31,31,31,31,31 'IIIII

'=====
'===== PROGRAMME PRINCIPAL =====
'=====

cls
const byte MESSAGE=("Carte a PB3H - programme de test - validation : rouge - menu suivant
: bleue")
for DECAL = 0 to 92
    'mise en mémoire du message
    'boucle de décalage de message (92 car le message fait 109 termes et 109-16=92)
    for X = 0 to 15
        'boucle de position d'écran
        SYMBOL=X+DECAL
        'chargement du terme n+1
        LETTRE = MESSAGE( SYMBOL )
        'chargement du symbole à afficher
        busout &HA1;X;&H00;&HA2;LETTRE;&H00
        'envoi à l'écran du symbole L à la position X
    next X
    'au suivant
    delay 80
    'ch'tite tempo

    touche=adkeyin(clav)
    'bouton caché pour sauter l'intro
    if touche=1 then goto 10
next DECAL

10  cls          'maintenant se suivent six menus pour accéder aux six fonctions à tester
    beep buz
    LOCATE 0,0 : PRINT "Test des ent ana"
    LOCATE 0,1 : PRINT "V:rouge MS:bleue"
15  touche=keydelay (adkeyin(clav),0,30,10)
    if touche=3 then gosub max
    if touche=2 then goto 20
    goto 15

20  cls
    beep buz
    LOCATE 0,0 : PRINT "Test des entrees"
    LOCATE 0,1 : PRINT "V:rouge MS:bleue"
25  touche=keydelay (adkeyin(clav),0,30,10)
    if touche=3 then gosub entrees
    if touche=2 then goto 30
    goto 25

30  cls
    beep buz
    LOCATE 0,0 : PRINT "Test des sorties"
    LOCATE 0,1 : PRINT "V:rouge MS:bleue"
35  touche=keydelay (adkeyin(clav),0,30,10)
    if touche=3 then gosub sorties
    if touche=2 then goto 40
    goto 35

40  cls
    beep buz
    LOCATE 0,0 : PRINT "Test des relais "
    LOCATE 0,1 : PRINT "V:rouge MS:bleue"
45  touche=keydelay (adkeyin(clav),0,30,10)
    if touche=3 then gosub relais
    if touche=2 then goto 50
    goto 45

50  cls
    beep buz
    LOCATE 0,0 : PRINT "Test du bus I2C "
    LOCATE 0,1 : PRINT "V:rouge MS:bleue"
55  touche=keydelay (adkeyin(clav),0,30,10)
    if touche=3 then gosub i2c
    if touche=2 then goto 60
    goto 55

60  cls
    beep buz
    LOCATE 0,0 : PRINT "Test bus RS232"
    LOCATE 0,1 : PRINT "V:rouge MS:bleue"

```

```
65  touche=keydelay (adkeyin(clav),0,30,10)
    if touche=3 then gosub rs232
    if touche=2 then goto 10
    goto 65
```

```
'=====
'=====  ROUTINES  =====
'=====
```

max:

```
A= 254/10          ' Détermine le nombre de valeurs affichées par caractère
B= A/5            ' Chaque caractère de l'afficheur LCD fait 5 pixels de large
cls
beep buz
LOCATE 0,0 : PRINT "Test des entrees"
LOCATE 0,1 : PRINT "  analogiques  "
delay 1500
cls
beep buz
locate 0,0 : PRINT "  actionnez le  "
locate 0,1 : PRINT "  potentiometre "
delay 1500
cls
locate 1,0 : print "tension :"          'affichage des caractères invariants
locate 15,1 : print "%"
```

100

```
out cs,0          'acquisition analogique via le max 1270 (voir doc)
shiftout clock,din,1,&b10000001
rslt=shifftin(clock,dout,1,16)
out cs,1

I=rslt/8          'je met de coté le résultat sur 8 bits pour le bargraphe
rslt=rslt+30     'traitement du résultat pour le convertir en une
                                     tension 0-5V

rslt=rslt/10
rslt=rslt*243
entier=rslt/10000
reste=rslt-(entier*10000)
reste=reste/1000

locate 15,0 : print "V"          'affichage de la tension mesurée : « partie_entière,reste »
locate 13,0 : print dec(reste,1,1)
locate 12,0 : print ","
locate 11,0 : print dec(entier,1,1)

IF I=0 THEN      'partie réalisant l'affichage du bargraphe
LOCATE 0,1
print " "
ELSE
Pos= (I-1)/A     ' Tout se joue dans ces trois lignes !!!
C= (I-1) MOD A +1
Char= (C-1)/B
LOCATE Pos,1
BUSOUT &HA2,Char,&H00          'affichage de la brique de bargraphe correspondante à la
                                     valeur de I

I=I*100
I=I/245
if I>100 then I=100          'correction des erreurs d'arrondis
if pos=0 then
locate 11,1
print dec(I,3,1)            'affichage de la valeur de I 0 à 100%
locate 1,1
print " "
goto 100
end if
pos=pos-1
for W=0 to pos          'on complète la partie gauche du bargraphe par des briques pleines
    locate W,1
    BUSOUT &HA2,5,&H00
next W
W=pos+2
for W=W to 10          'on complète la partie droite du bargraphe par des blanc
    locate W,1
    print " "
next W
END IF
locate 11,1
print dec(I,3,1)          'affichage de la valeur de I 0 à 100%

touche=keydelay (adkeyin(clav),0,30,10) 'condition de sortie
if touche=2 then return
delay 10
```

```

goto 100

'
entrees:
  cls
  beep buz
  LOCATE 0,0 : PRINT "Test des entrees"
  LOCATE 0,1 : PRINT "etats logiques :"
  delay 3000
  const byte data = (0,6,1,2,3,7,5,4)      'correspondance entre les broches du µC et les noms
                                           que j'ai donné aux sorties

  cls
  beep buz
200  LOCATE 0,0 : PRINT "entrees: 7654321"
  for I=1 to 7
    W=data(I)
    if in(W)=1 then X=1 else X=0          'lecture des entrées
    decal=16-I
    locate decal,1
    print dec(X,1,1)                     'affichage des résultats
  next I
  LOCATE 0,1 : PRINT "etats  : "

  touche=keydelay (adkeyin(clav),0,30,10) 'condition de sortie
  if touche=2 then goto 30
  goto 200

'
sorties:
  cls
  beep buz
  LOCATE 0,0 : PRINT "Test des sorties"
  LOCATE 0,1 : PRINT "etats logiques :"
  delay 3000
  cls
  beep buz
  LOCATE 0,0 : PRINT "sorties:87654321"
  LOCATE 0,1 : PRINT "etats  :01100100"   'activation des sorties 7,6,3
  out s1,0
  out s2,0
  out s3,1
  out s4,0
  out s5,0
  out s6,1
  out s7,1
  out s8,0
  pwm pwml,190
  pwm pwm2,50
300  touche=keydelay (adkeyin(clav),0,30,10)
  if touche=2 then
    pwmoff pwml
    pwmoff pwm2
    goto 40
  end if
  goto 300

'
relais:
  cls
  beep buz
  LOCATE 0,0 : PRINT "Test des relais "
  for I = 0 to 3
    LOCATE 0,1 : PRINT " RE2:0 RE1:1 "    'activation des relais
    out re2,0
    out re1,1
    delay 800
    toggle re2
    toggle re1
    LOCATE 0,1 : PRINT " RE2:1 RE1:0 "
    delay 800
  next I
  out re2,0
  out re1,0
  goto 50
  return

'
i2c:
  cls
  I=0
  donnee=0
  adresse=0

```

```

for I = 0 to 9                                'écriture et lecture de 10 données de 1 à 10
  adresse=adresse+1
  donnee=donnee+1
  gosub ecriture
  delay 3                                     'tempo nécessaire à la mémoire pour procéder à
                                              l'écriture (ne pas diminuer !)

  gosub lecture
  locate 0,0 : print " Test  EEPROM  "
  LOCATE 7,1 : PRINT dec(donnee,2,1)
  delay 200
next I
if donnee = 10 then
LOCATE 7,1 : PRINT "OK"                       'si le µC parvient à relire ce qu'il à écrit dans
                                              l'eprom, tout est bon !

delay 2000
goto 60
end if
locate 0,0 : print "Probleme EEPROM " 'sinon c'est moins bon !
beep buz

ecriture:                                     'routine d'écriture dans la mémoire I2C
  gosub start                                 'condition de start
  shiftout scl,sda,2,&b10100000              'envoi de l'adresse composant pour écriture
  shiftout scl,sda,2,adresse                 'envoi de l'adresse ou écrire
  shiftout scl,sda,2,donnee                 'envoi de la donnée à écrire
  gosub stop                                 'condition de stop
  return

lecture:
  gosub start                                 'condition de start
  shiftout scl,sda,2,&b10100000              'envoi de l'adresse composant pour écriture
  shiftout scl,sda,2,adresse                 'envoi de l'adresse ou lire
  gosub start                                 'condition de start
  shiftout scl,sda,2,&b10100001              'envoi de l'adresse composant pour lecture
  donnee=shiftin(scl,sda,1)                 'lecture de la donnée
  gosub stop                                 'condition de stop
  return

start:
  out scl,1
  out sda,1
  out sda,0
  out scl,0
  return

stop:
  out sda,0
  out scl,1
  out sda,1
  return
'

```

```

rs232:
  cls
  beep buz
  LOCATE 0,1 : PRINT " Liaison ..."
  serout rsout,103,0,1,["INSA Strasbourg"]   'envoi du message "INSA Strasbourg"
  delay 1000
  goto 10

```

BIBLIOGRAPHIE

Une carte de développement à 68HC11 :

Martin F.G. – The handy board technical reference. MIT, 1997

Une carte entrée/sortie sur port parallèle pour PC :

Wenzler T – Je pilote l'interface parallèle de mon PC. Publitronic

Un livre d'initiation à l'électronique numérique et au Basic Stamp

Mergy Yves – Pour s'initier à l'électronique logique et numérique. ETSF

Un livre traitant de la mise en œuvre des Basic Stamp (concurrent des PicBasic)

Tavernier C – Les Basic Stamp. Dunod

Un livre traitant très en détails toutes sorte de capteurs (IR, US, LDR etc)

Giamarchi F – Petits robots mobiles. ETSF

ADRESSES INTERNET

Le fonctionnement du bus I2C

- La page de Philips : <http://www.semiconductors.philips.com>
- La page de Pierre Col : <http://col2000.free.fr/>
- La page de Christian Tavernier : <http://www.tavernier-c.com/>
- Une page perso : <http://webperso.easynet.fr/~chrisg/dll.htm>
- Une autre : http://1100f.free.fr/tout_sur_le_bus_i2c.htm

Le fonctionnement des liaisons asynchrones RS232

- La page de Christian Tavernier : <http://www.tavernier-c.com/>

Le site de l'importateur du PicPasic PB3H

- Lextronic : <http://www.lextronic.fr>

Le site du fabricant du PIC16F877

- <http://www.microchip.com>

Le site du fabricant du MAX 1270

- <http://www.maxim-ic.com>

Une société commercialisant un compilateur Basic pour PIC

- <http://www.melabs.com>

Une page traitant de la robotique amateur, pleine de bons conseils (interface de puissance, capteurs, pilotage de servomoteurs)

- <http://fribotte.free.fr/>

Un forum de discussion traitant des Picbasic notamment

- <http://elecrob.proboards7.com/index.cgi?board=Basic>

Conclusion

Ce projet m'a particulièrement intéressé dans la mesure où j'en ai choisi le sujet et qu'il répondait à une envie personnelle. La conception de la carte m'a permis d'approfondir mes connaissances et ma pratique de l'électronique numérique et de découvrir des technologies comme les bus I2C, SPI et les liaisons RS232. L'utilisation de modules périphériques tels que le convertisseur analogique/numérique max1270 ou bien l'écran LCD donne des capacités intéressantes à ce montage. Le choix du microcontrôleur Picbasic PB3H qui palie sa relative lenteur par une grande facilité de développement répond bien aux objectifs que je m'étais fixé, une carte didactique et polyvalente. Roboticien amateur depuis plusieurs années je sais que cette carte ne tardera pas à trouver sa place au sein de mes réalisations précédentes et de leur donner un nouvel élan.