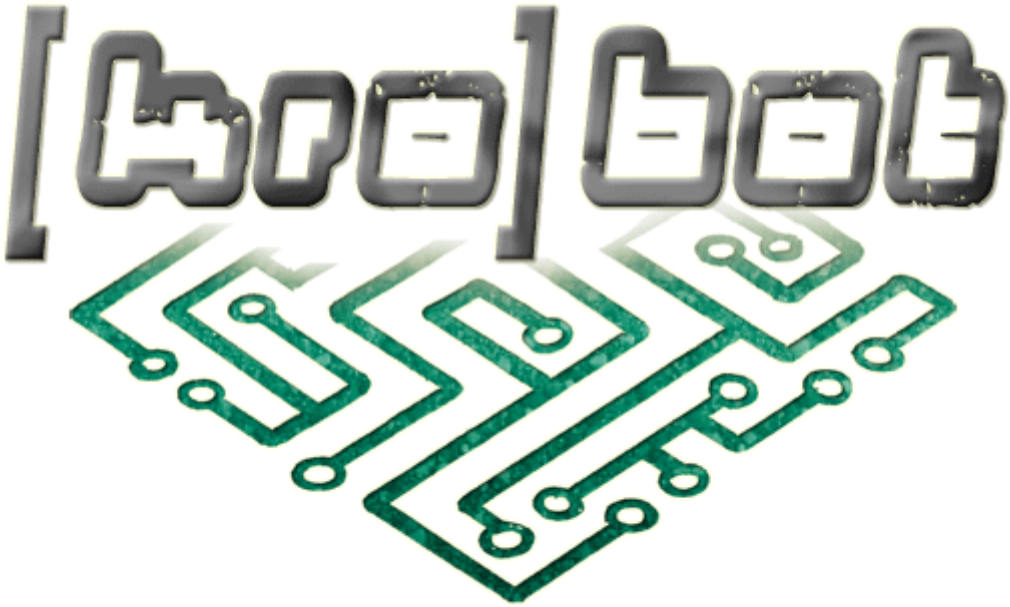
De la conception à la réalisation d'un robot

Olivier Bichler*

21 avril 2007



Résumé

Le guide officiel du club de robotique de l'ENS de Cachan !
Les bases pour concevoir des circuits électroniques analogiques et numériques, les réaliser en utilisant les logiciels de la suite Cadence OrCAD et les tester, avec un seul objectif : concevoir un robot qui fonctionne...

*École Normale Supérieure de Cachan. E-mail : olivier.bichler@ens-cachan.fr.

Table des matières

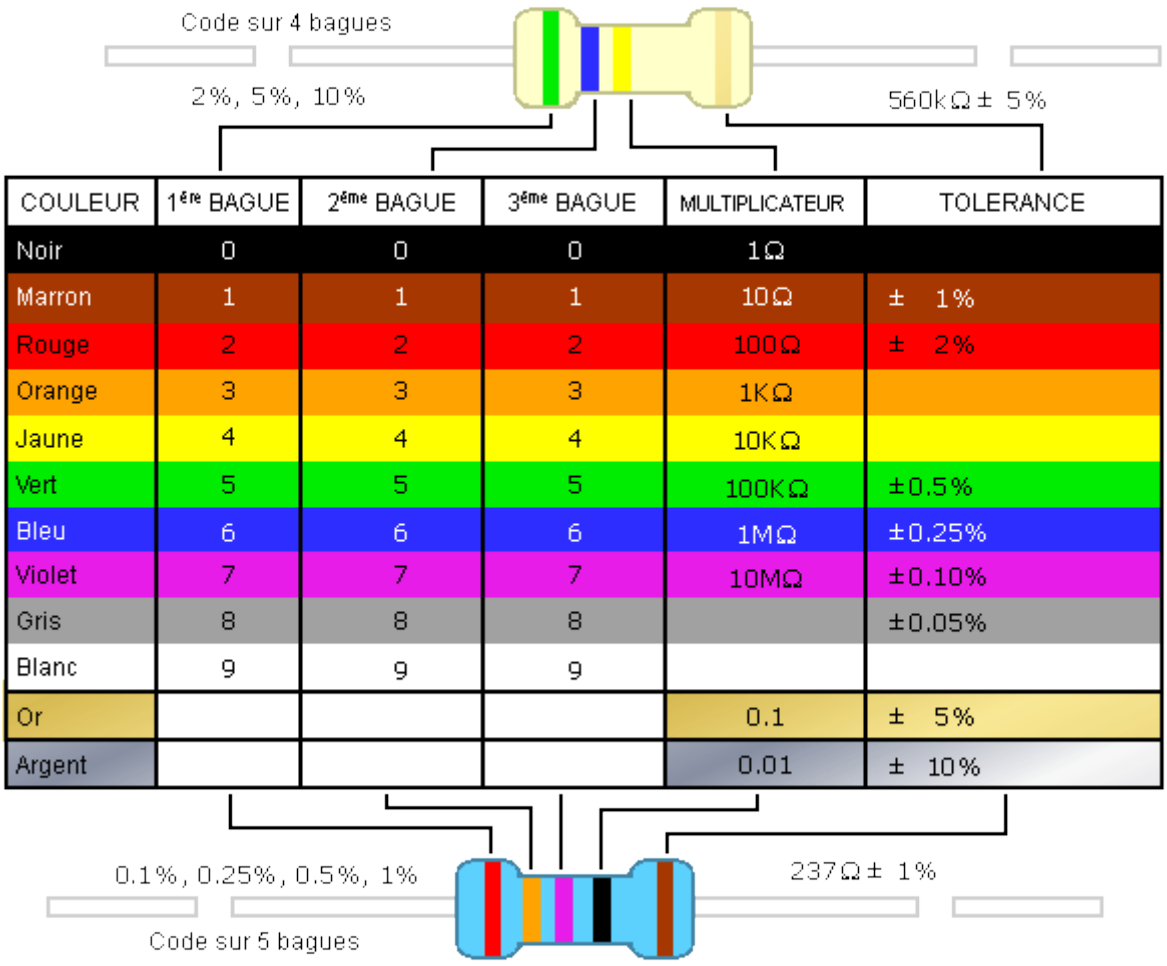
Première partie
Composants et théorie

1 Les résistances

Les résistances de base ont une puissance de 1/4 W et une tolérance de 5% (bague or).
Bien que les résistances n'ont pas de sens, il est bien de les souder dans le sens de la lecture afin de faciliter le décodage de leur valeur.

1.1 Code des couleurs

Parce qu'il est toujours utile de savoir lire rapidement la valeur d'une résistance !



COULEUR	1ère BAGUE	2ème BAGUE	3ème BAGUE	MULTIPLICATEUR	TOLERANCE
Noir	0	0	0	1Ω	
Marron	1	1	1	10Ω	± 1%
Rouge	2	2	2	100Ω	± 2%
Orange	3	3	3	1KΩ	
Jaune	4	4	4	10KΩ	
Vert	5	5	5	100KΩ	±0.5%
Bleu	6	6	6	1MΩ	±0.25%
Violet	7	7	7	10MΩ	±0.10%
Gris	8	8	8		±0.05%
Blanc	9	9	9		
Or				0.1	± 5%
Argent				0.01	± 10%

FIG. 1 – Code des couleurs des résistances

La première bague est celle au ras du boitier, les autres bagues étant accolées à celle-ci.
Et voici une petite astuce mnémotechnique pour se rappeler de l'ordre des couleurs : « Ne Manger Rien Ou Jeuner Voilà Bien Votre Grande Bêtise ».

1.2 Séries normalisées

Les valeurs des résistances sont standardisées et plusieurs séries de valeur existent :
- E6 : 10, 15, 22, 33, 47, 68 ;
- E12 : 10, 12, 15, 18, 22, 27, 33, 39, 47, 56, 68, 82 ;
- E24 : 10, 11, 12, 13, 15, 16, 18, 20, 22, 24, 27, 30, 33, 36, 39, 43, 47, 51, 56, 62, 68, 75, 82, 91 ;
- E48 ...

Remarque : pour un meilleur étalement, les séries sont telles que le rapport entre deux valeurs successives est identique. Par exemple, dans la série E6, ce rapport est égal à 1,47 environ (10 à la puissance 1/6).
Il est intéressant de se limiter à de petites séries, si possible E6, ce qui permet de faire des économies et être assuré d'avoir ces résistances de disponibles.

1.3 La résistance de tirage

La fonction de base d'une résistance de tirage est d'assurer que le circuit prenne une valeur logique par défaut en entrée, lorsque celle-ci n'est pas fixée par ailleurs. On parle de résistance pull-up (*tirage vers le haut*), ainsi que de résistance pull-down (*tirage vers le bas*), selon que l'entrée est fixée à l'état haut ou à l'état bas respectivement.

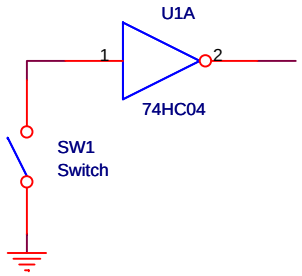


FIG. 2 – Switch sans pull-up

Sur le schéma de la figure 2, la porte U1A a une entrée (pin 1) et une sortie (pin 2). La plupart des circuits logiques ont leurs entrées en état de haute impédance, ce qui signifie que si rien n'est connecté à la pin 1 du circuit logique précédent (ce qui est le cas lorsque l'interrupteur est ouvert), le potentiel à cette broche est flottant. Un potentiel flottant peut être considéré comme étant un 1 logique par la porte (par exemple, sachant que c'est généralement le cas), mais le moindre bruit électronique peut l'amener tantôt à l'état haut, tantôt à l'état bas. Ainsi, un simple bout de piste peut faire antenne et engendrer des perturbations suffisantes pour faire changer l'état du circuit logique.
Afin de fixer le potentiel même lorsque l'interrupteur est ouvert, il faut ajouter une résistance de tirage, comme illustré sur la figure 3.

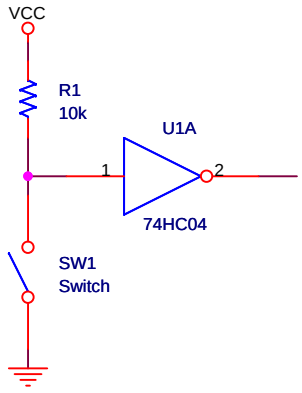


FIG. 3 – Switch avec pull-up

Sans résistance on aurait un court-circuit franc lors de la fermeture de l'interrupteur, ce qui peut engendrer un courant très important de VCC à la masse susceptible de faire chauffer et bruler les pistes (qui font alors office de fusibles) ainsi que les composants. De plus, VCC serait ramené à un potentiel quasiment nul ce qui empêchera le circuit logique de fonctionner !
La résistance permet donc de limiter le courant lorsque l'interrupteur est fermé, tout en assurant la mise à 0 de la pin 1 du circuit logique sans perturbation du potentiel de VCC.

Une rapide loi d'Ohm permet de calculer le courant circulant alors dans la résistance :

$$I = \frac{VCC}{R1} = \frac{5V}{10k\Omega} = 0,5mA$$

Lorsque l'interrupteur est ouvert, la pin 1 est mise à l'état haut au travers de la résistance de tirage.

Il est clair que plus la résistance de tirage est grande, plus le courant la traversant sera faible et moins elle consommera :

$$P = \frac{VCC^2}{R1} = \frac{5^2}{10000} = 2,5mW$$

Cependant, les circuits logiques ont généralement besoin d'un courant minimum en entrée pour fixer leur niveau logique (courant certes très faible et souvent approximé à 0), si bien qu'une résistance trop importante ne fonctionnera pas. Le courant minimum à fournir en entrée varie d'un circuit logique à l'autre et est en général indiqué dans sa datasheet, cependant des résistances de 10 kΩ à 47 kΩ peuvent être utilisées sans risque avec la grande majorité des circuits logiques (10 kΩ étant la valeur la plus courante).

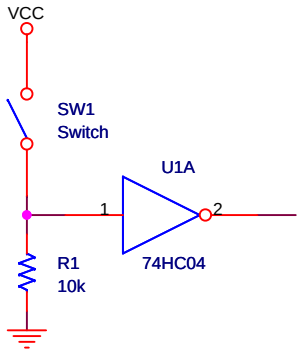


FIG. 4 – Switch avec pull-down

La figure 4 présente sur le même principe une résistance pull-down, qui permet d'imposer un niveau bas par défaut contrairement à la pull-up qui impose par défaut un niveau haut. Bien que moins courante, la résistance de pull-down est tout à fait valable, par exemple dans le cas où une pin non inverseuse d'un circuit logique doit être mise à 0 par défaut.

1.4 La résistance de limitation de courant

Dans cette section, nous allons voir un exemple simple d'utilisation d'une résistance de limitation de courant avec une broche de sortie d'un circuit logique et une Light Emitting Diode (LED). Contrairement à une broche d'entrée, qui est en haute impédance, une broche de sortie a toujours au moins deux états possibles : état haut et état bas.

1.4.1 Structure interne des sorties des circuits logiques

Afin de comprendre pourquoi on utilise des résistances de limitation de courant, nous allons d'abord voir ce qui se passe lorsque l'on connecte une sortie d'un circuit logique sur une entrée d'un circuit logique. Pour cela, prenons l'exemple du 74HC04, qui n'est q'une simple porte inverseuse. On peut voir figure 5 la structure interne, très simplifiée, mais tout à fait valide sur le principe, du circuit 74HC04. Quand un courant est appliqué sur la pin 1, le transistor devient passant et la pin 2 se retrouve au potentiel 0. Le transistor joue ici le rôle d'un interrupteur commandé.

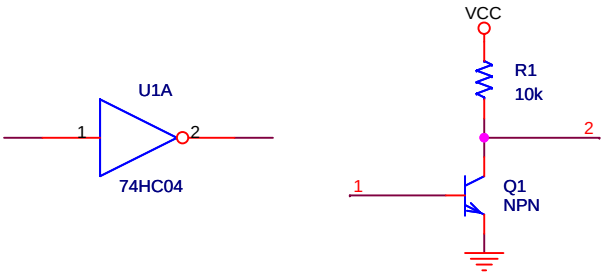


FIG. 5 – Structure interne du 74HC04

Cette structure est intéressante, car elle représente également la structure interne de base des sorties (2 états) de n'importe quel circuit logique, qui sont en fait des interrupteurs commandés par des transistors. On constate donc que les broches de sortie utilisent une résistance pull-up.

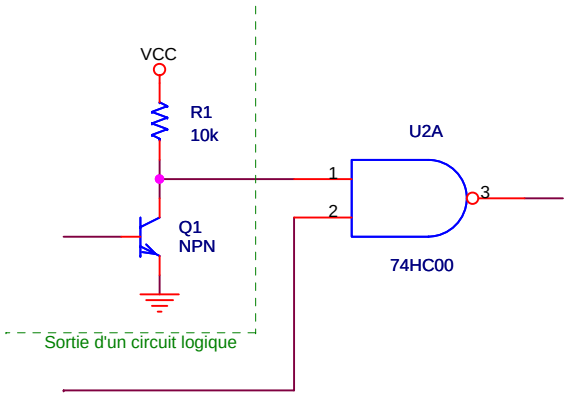


FIG. 6 – Structure interne d'une sortie d'un circuit logique

Si l'on branche une sortie sur une entrée d'un deuxième circuit logique, comme illustré figure 6, lorsque Q1 est bloqué, R1 assure le niveau logique haut sur l'entrée de U2A, tandis que lorsque Q1 est passant, l'entrée d'U2A est mise à 0. La résistance pull-up R1 (qui est rappelons-le une résistance interne) assure le passage d'un courant suffisant pour imposer les bons niveaux logiques au circuit U2A, sans avoir à ajouter de résistance de pull-up supplémentaire. Ainsi, pour la plupart des circuits logiques, on peut directement connecter une sortie sur une entrée d'un autre circuit.

1.4.2 Montage d'une LED sur la sortie d'un circuit logique

Considérons désormais le cas où la sortie d'un circuit logique n'est pas une entrée d'un autre circuit logique. Par exemple, on aimerait commander une LED avec cette sortie. Il est alors nécessaire de prendre certaines précautions et de considérer le cas où la sortie est mise à l'état bas, ce qui équivaut à une connexion directe avec la masse. Dans ce cas, il est important de contrôler le courant qui va circuler dans la sortie du circuit logique. La plupart des circuit logiques peuvent supporter sur leurs sorties un courant allant jusqu'à 20 mA (par broche).

Il faut savoir qu'une LED par exemple, a une résistance interne très faible. Lorsqu'elle est passante, elle ne va donc pas limiter le courant et risque de fondre si celui-ci est trop important. De même, si l'on connecte directement cette LED entre VCC et la sortie du circuit logique, la mise à l'état bas de celle-ci équivaudrait pratiquement à court-circuiter VCC avec la masse, ce qui pourrait provoquer la destruction du circuit logique en plus de la LED. Avant de connecter la LED sur une sortie d'un circuit logique, il faut donc connaitre le courant maximum admissible par celle-ci, sachant qu'il ne faut de toute manière pas dépasser les 20 mA que peut supporter la sortie.

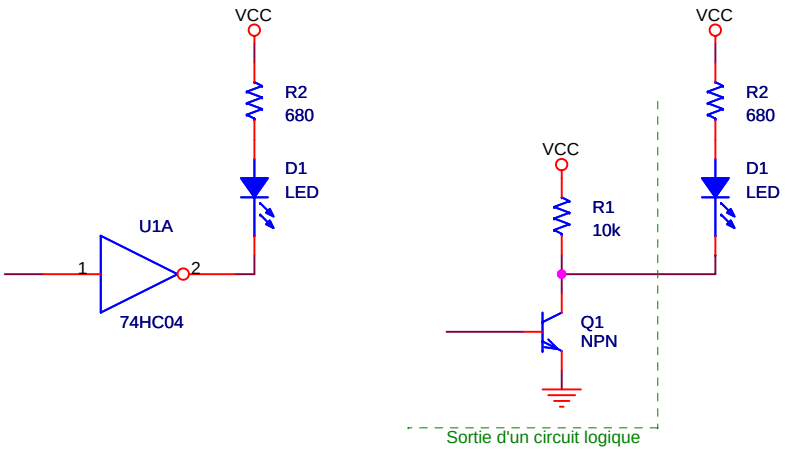


FIG. 7 – Montage d'une LED avec résistance de limitation

Le courant maximal est fixé avec une résistance de limitation de courant. En considérant que la plupart des LEDs supportent au moins 15 mA, si VCC = 5V, la loi d'Ohm nous donne la valeur minimum de la résistance à utiliser :

$$R = \frac{VCC}{I} = \frac{5V}{15mA} = 333\Omega$$

On peut se donner une marge et limiter la puissance consommée en prenant une valeur de résistance plus importante (mais pas trop quand même pour ne pas trop diminuer la luminosité de la LED). Une valeur couramment utilisée pour une LED de base est 680 Ω.

Le montage avec résistance de limitation est donné figure 7 : quand la broche de sortie est à l'état haut, la LED est bloquée (différence de potentiel à ses bornes nulle) et lorsque la broche de sortie est à l'état bas, la LED est passante et tout se passe comme si la pin 2 du circuit logique était directement connecté à la masse.

1.5 Les résistances de puissance

La figure 8 montre quelques résistances de puissance, rangées dans l'ordre croissant de puissance, de gauche à droite. Parce qu'il n'est pas toujours évident de deviner qu'il s'agit de résistances...

FIG. 8 – Quelques résistances de puissance

1.6 Les réseaux de résistances

Les réseaux de résistances sont très pratiques pour des résistances de tirage, entre un switch et un circuit logique par exemple, ou lorsqu'on a besoin de plusieurs résistances identiques et que l'on veut gagner de la place.

Il y a deux types principaux de réseaux de résistances : plusieurs résistances indépendantes (schéma de gauche figure 9), ou des résistances avec un commun (schéma de droite).

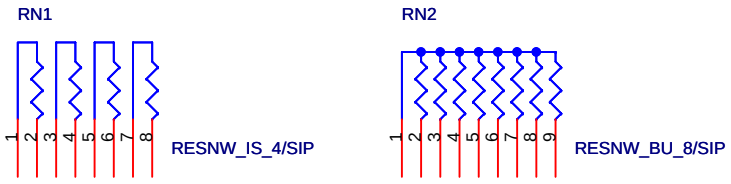


FIG. 9 – Schéma de réseaux de résistances

Le point sur le réseau permet de repérer la première broche (le commun, s'il s'agit du deuxième type de réseau), figure 10.

FIG. 10 – Un réseau de résistances

2 Les condensateurs

2.1 Principales caractéristiques

- **La capacité**, exprimée en farads (F) ;
- **La tension nominale** en courant continu : exprimée en volts continus, il s'agit de la tension maximale que peut supporter en permanence le condensateur à ses bornes. Une tension supérieure peut provoquer l'explosion du composant ;
- **La tolérance**, qui indique les écarts de valeur maximum que peut avoir la capacité du condensateur, exprimée en % par rapport à sa valeur idéale ;
- **Déviation en température** : la capacité des condensateurs varie avec la température. Cette variation peut être positive (augmentation de la capacité), ou négative (diminution). Elle s'exprime en ppm/°C (pour rappel, 1 ppm = une partie par million = 0,000 1%) ;
- **Courant de fuite** : le condensateur, même isolé, se décharge au cours du temps, via un courant de fuite circulant entre ses deux broches, qui peut être de l'ordre du mA ;
- **Résistance série** : les condensateurs n'étant pas parfaits, on associe en série à la capacité une résistance, de très faible valeur (généralement très inférieur à l'ohm).

2.2 Les types de condensateurs

2.2.1 Céramiques

FIG. 11 – Condensateurs céramique

1. A plaquette ;
2. A disque ;
3. Tubulaire (pas courant).

Classe	Capacité	Série	Tolérance	Tension supportée
I	1 pF à 100 nF	E12	±5 à ±10%	50 V à 100 V
II	100 pF à 4,7 μF	E6	±10%	50 V à 100 V
III	1 nF à 4,7 μF	E3	±20% ou plus	50 V à 100 V

Tab. 1 – Caractéristiques typiques des condensateurs céramique

Il existe en tout 4 classes de condensateurs céramique, dans la classification de l'Electronic Industries Alliance (EIA). Plus la classe est petite, meilleur sont les caractéristiques du condensateur.

Emploi L'utilisation des condensateurs céramique est très large, voici plutôt les cas où l'on utilise de préférence d'autres types de condensateurs :
- dans les oscillateurs où une grande stabilité de capacité est requise, on préfère les condensateurs au mica, au polystyrène ou au polycarbonate ;
- dans les circuits de filtrage et de découplage où une très grande capacité est requise, on utilise les condensateurs électrolytiques (aluminium et tantale) ;
- dans les circuits à basse fréquence car leur capacité est généralement trop faible.

2.2.2 A film plastique (MK)

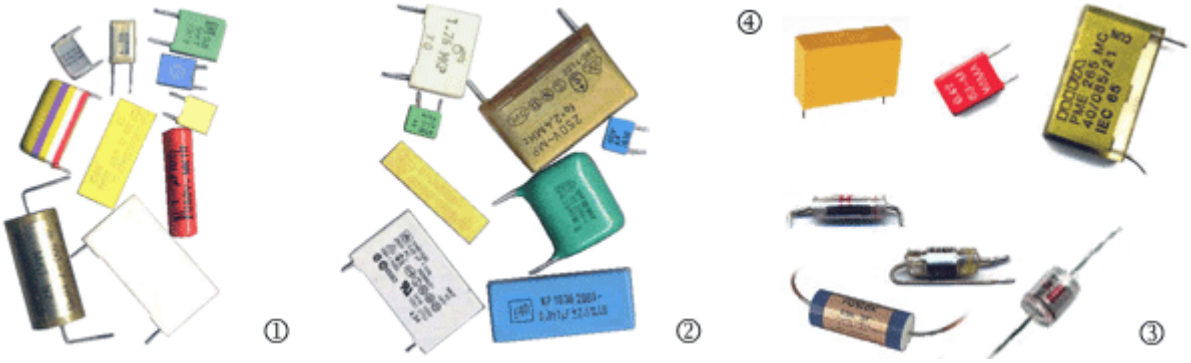


Fig. 12 – Condensateurs plastique

1. (M)KT : polyester (polyéthylène ou mylar) ;
2. (M)KP : polypropylène ;
3. (M)KS : polystyrène (styroflex) ;
4. (M)KC : polycarbonate.

La présence d'un « M » signifie que le condensateur est à film métallisé.

Type	Capacité	Série	Tolérance	Tension supportée
Polyester	1 nF à 15 μF	E12	10%	50 V à 1 500 V
Polypropylène	100 pF à 10 μF	E12	10%	63 V à 2 000 V
Polystyrène	10 pF à 47 nF	E12	10%	30 V à 630 V
Polycarbonate	100 pF à 15 μF	E12	10%	63 V à 1 000 V

Tab. 2 – Caractéristiques typiques des condensateurs plastique

Emploi
- Polyester (caractéristiques plutôt médiocres) : circuits BF, découplage, filtrage et applications où les rapports performances / prix et capacité / encombrement doivent être grands ;
- Polypropylène (ses propriétés se dégradent en HF) : circuits accordés, alimentations, BF ;
- Polystyrène (bonnes caractéristiques, peu de pertes) : circuits accordés HF, alimentations, BF, timer, filtres ;
- Polycarbonate (très stable) : circuits BF, découplage, filtrage.

Pris ensembles, les condensateurs à film plastique satisfont à la plupart des usages de l'électronique courant.

2.2.3 Électrolytiques

Électrolytiques aluminium (chimique) Ce type de condensateur est polarisé dans la plupart des cas.

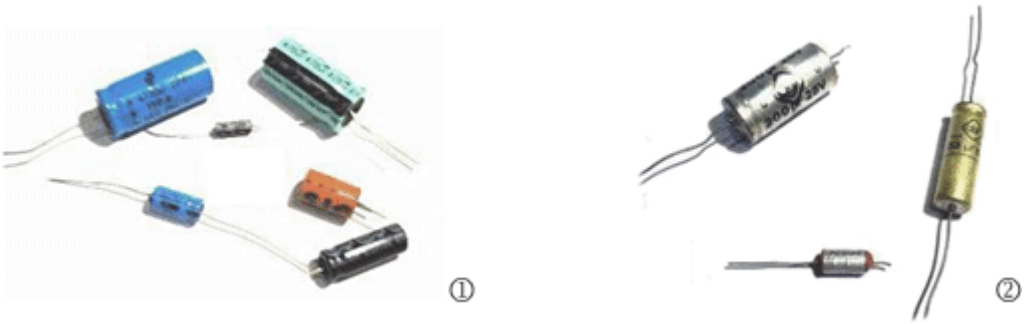


Fig. 13 – Condensateurs chimique

1. Polarisé ;
2. Non polarisé.

Capacité	Série	Tolérance	Tension supportée
0,1 μF à 68 mF	E6	20%	jusqu'à 450 V

Tab. 3 – Caractéristiques typiques des condensateurs chimique

Emploi Partout où une forte valeur est exigée, filtrage, découplage, lissage, ligne de retard...

Problèmes connus Ce type de condensateur vieilli (la capacité diminue au fil du temps) et il faut se méfier particulièrement des récupérations, surtout si le montage à effectuer requiert de la précision, d'autant plus qu'un condensateur récent de ce type sera plus petit qu'un vieux aux caractéristiques identiques, comme le montre la figure 14.

Fig. 14 – Comparaison entre un ancien et un condensateur chimique récent, aux caractéristiques identiques (25 V, 470 μF)

Un autre problème qui peut survenir avec ce type de condensateur, est son gonflement, dû à une fabrication défectueuse. Il s'en suit une perte significative de la capacité, qui a tendance à provoquer des instabilité par exemple sur des cartes de PC, où l'on rencontre parfois ce problème. Un condensateur ayant « gonflé » ne peut plus être utilisé et doit être remplacé (voir figure 15).

FIG. 15 – Condensateurs électrolytiques aluminium ayant gonflés (le 1er par dessus, le 2e par dessous et du liquide électrolytique s'est échappé du 3e)

Électrolytiques tantale Ces condensateurs sont également polarisés. La plus est généralement indiqué par un signe « + » du coté de la broche correspondante.

FIG. 16 – Condensateurs tantale

Capacité	Série	Tolérance	Tension supportée
0,1 μF à 150 μF	E6	20%	6,3 V à 35 V

TAB. 4 – Caractéristiques typiques des condensateurs tantale

Emploi Temporisateurs de longue durée. Inconvénient : leur prix assez élevé.

2.2.4 Au mica

Capacité	Tolérance	Tension supportée
1 pF à 100 nF	1 à 20%	300 V à plus de 2 500 V

TAB. 5 – Caractéristiques typiques des condensateurs au mica

Les condensateurs au mica sont réservés à des applications nécessitant une grande stabilité, peu de pertes ou des tensions élevées, en raison de leur cout élevé.

2.2.5 Au papier

Capacité	Tolérance	Tension supportée
500 pF à 0,5 μF	20%	125 V à 1 000 V

TAB. 6 – Caractéristiques typiques des condensateurs au papier

Les condensateurs au papier ne s'utilisent plus que rarement, mais on le rencontre encore notamment dans les applications à haute tension où un condensateur non polarisé est nécessaire.

2.3 Marquage des condensateurs

Lorsque la taille le permet, la valeur des capacités est généralement clairement indiquée, comme sur les condensateurs électrolytiques par exemple. Sinon, il existe une large variété de codes, utilisant des caractères alphanumériques ou des couleurs, dont voici les principaux :

2.3.1 Indication de la capacité

Valeur sur 2 chiffres Il s'agit directement de la valeur du condensateur, en pico-farads (pF).

Valeur sur 3 chiffres Les 2 premiers chiffres indiquent une valeur en pico-farads (pF), tandis que le troisième est le multiplicateur (qui correspond dans la plupart des cas, mais pas toujours, au nombre de 0 à ajouter pour obtenir la valeur du condensateur, voir tableau 7).

Chiffre	Multiplicateur
0	×1
1	×10
2	×100
3	×1 000
4	×10 000
5	×100 000
6 (non utilisé)	
7 (non utilisé)	
8	×0,01
9	×0,1

Tab. 7 – Chiffre multiplicateur

Il y a ambiguïté lorsque le 3e chiffre est 0. Par exemple, « 560 » pourrait signifier 560 pF, ou 56×1 = 56 pF. En l'absence d'autre indication, il s'agit généralement de la première valeur qui est la bonne, le 3e chiffre étant alors le dernier chiffre significatif, mais ce n'est pas certain.

Autres indications de la valeur Une lettre correspondant à un certain multiple du pico-farad (pF) est à la place de la virgule décimale (marquage européen).

Lettre	Multiplicateur
p (pico) ou R (virgule décimale)	×1
n (nano)	×1 000
μ ou u (micro)	×1 000 000
m (milli)	×1 000 000 000

Tab. 8 – Symbole multiplicateur

Par exemple, « 4n7 » signifie 4,7 nF, ou encore, « R47 » signifie 0,47 pF.

2.3.2 Indication de la tolérance

Lorsque celle-ci n'est pas indiquée, la tolérance est généralement de ±20%.

Code tolérance Une lettre supplémentaire, juste après la valeur, indique en plus la tolérance, voir tableau 9.

Lettre	Tolérance
A	±0,05 pF
B	±0,1 pF
C	±0,25 pF
D	±0,5 pF
E	±0,5%
F	±1%
G	±2%
H	±3%
J	±5%
K	±10%
M	±20%
N	±30%
P	-0% à +100%
S	-20% à +50%
W	-0% à +200%
X	-20% à +40%
Z	-20% à +80%

Tab. 9 – Lettre de tolérance

Autres indications de la tolérance Elle peut être directement exprimée en %. « 5% » correspondra donc à une tolérance de ±5%.
On l'a trouve également sans le symbole %, séparée de la valeur de la capacité par un « / », quand il ne s'agit pas de la tension nominale, si elle n'est pas indiquée ailleurs (le nombre étant alors généralement un peu élevé pour une tolérance).

2.3.3 Indication de la tension nominale

La tension est généralement indiquée après le code de tolérance lorsque celui-ci est présent, ou directement après la valeur de la capacité. Lorsqu'il n'y a pas d'unité de précisée, il s'agit de volts (V). On l'a trouve également séparée du code de tolérance ou de la valeur de la capacité par un « / ».

Lorsque le signe « ~ » est présent, il s'agit de la tension nominale alternative.

2.3.4 Indication de la dérive de la capacité

Codification EIA condensateurs céramique classe I La dérive de la capacité des condensateurs céramique de classe I est codifiée à l'aide de 3 caractères alphanumériques. Les deux premiers caractères indiquent la dérive (positive ou négative) et le troisième la tolérance applicable sur cette dérive entre +25 et +85°C. Voir le tableau 10.

1er symbole (lettre)	Dérive en ppm/°C chiffre significatif	2e symbole (chiffre)	Dérive en ppm/°C multiplicateur	3e symbole (lettre)	Tolérance en ppm/°C
C	0,0	1	×(-1)	G	±30
B	0,3	2	×(-10)	H	±60
L	0,8	3	×(-100)	J	±120
A	0,9	4	×(-1 000)	K	±250
M	1,0	5	×1	L	±500
P	1,5	6	×10	M	±1 000
R	2,2	7	×100	N	±2 500
S	3,3	8	×1 000		
T	4,7				
V	5,6				
U	7,5				

Tab. 10 – Codification EIA condensateurs céramique classe I

Équivalences code Industry-Standards et code couleur Le code Industry-Standards est composé d'une lettre : N (coefficient de température négatif) ou P (positif), suivie d'un nombre représentant la dérive en ppm/°C.

Dérive en ppm/°C	Code EIA	Code Industry-Standards	Code couleur
0 ppm/°C	C0G	NP0	Noir
-33 ppm/°C	S1G	N033	Marron
-75 ppm/°C	U1G	N075	Rouge
-150 ppm/°C	P2G	N150	Orange
-220 ppm/°C	R2G	N220	Jaune
-330 ppm/°C	S2H	N330	Vert
-470 ppm/°C	T2H	N470	Bleu
-750 ppm/°C	U2J	N750	Violet
100 ppm/°C	M7G	P100	

Tab. 11 – Equivalences code Industry-Standards et code couleur, dérive de la capacité

Fig. 17 – Code couleur dérive de la capacité

L'unique bague de couleur au sommet des condensateurs céramiques code la dérive de la tolérance. Sur la figure 17, les condensateurs céramiques ont respectivement une dérive de 0 ppm/°C, -150 ppm/°C et -750 ppm/°C (de gauche à droite).

Codification EIA condensateurs céramique classe II et supérieur Ce code indique la variation maximale de la capacité à l'intérieur de la plage de température. La capacité de référence est celle mesurée à 25°C. Voir le tableau 12.

1er symbole (lettre)	Température minimale	2e symbole (chiffre)	Température maximale	3e symbole (lettre)	Variation max. de la capacité sur la plage de température
Z	10°C	2	45°C	A	+1,0%
Y	-30°C	4	65°C	B	±1,5%
X	-55°C	5	85°C	C	±2,2%
		6	105°C	D	±3,3%
		7	125°C	E	±4,7%
				F	±7,5%
				P	±10%
				R	±15%
				S	±22%
				T	+22%, -33%
				U	+22%, -56%
				V	+22%, -82%

Tab. 12 – Codification EIA condensateurs céramique classe II

2.3.5 Codification couleur des condensateurs céramique et plastique

Le marquage à l'aide de couleurs sur les condensateurs est passé en désuétude, et ne se trouve plus que sur certains anciens condensateurs. Ce codage ne sera donc pas détaillé ici.

2.3.6 Codification des condensateurs au mica

Code EIA Ce code est facilement identifiable car il commence par la lettre « R ». Il se lit comme suit :
- **R** : préfixe EIA ;
- **CM/DM** : forme du composant (moulé/plongé) ;
- **XX** : nombre codant la taille du composant ;
- **X** : dérive de la capacité et coefficient de la température (voir tableau 13) ;
- **XXX** : valeur sur 3 chiffres (le 3^{e} chiffre étant le multiplicateur, correspondant au nombre de 0 à ajouter, la valeur étant exprimée en pico-farads) ;
- (**R**) : point décimal (facultatif) ;
- **X** : lettre codant la tolérance (voir tableau 9) ;
- **X** : chiffre indiquant la tension de service, en centaines de volt ;
- **X** : lettre indiquant la gamme de température (voir tableau 14) ;
- **S/C** : connexions droites/pliées.

Lettre	Dérive max. en capacité	Coefficient de température
B	Non spécifié	Non spécifié
C	±(0,5% + 0,1 pF)	±200 ppm/°C
D	±(0,3% + 0,1 pF)	±100 ppm/°C
E	±(0,1% + 0,1 pF)	-20 à +100 ppm/°C
F	±(0,05% + 0,1 pF)	-0 à +70 ppm/°C

Tab. 13 – Dérive capacité et température, code EIA

Lettre	Gamme de température
M	-55 à 70°C
N	-55 à 85°C
O	-55 à 125°C
P	-55 à 150°C

Tab. 14 – Gamme de température, code EIA

Exemple, lecture de « RCM05E391RJ5OS » :
- **R** : préfixe EIA ;
- **CM05** : forme et taille du composant ;
- **E** : dérive de capacité de ±(0,1% + 0,1 pF) et coefficient de température de -20 à +100 ppm/°C ;
- **39** : capacité (1er et 2^{e} chiffres significatifs) ;
- **1** : capacité (nombre de zéros à ajouter aux chiffres ci-dessus) ;
- **R** : point décimal (facultatif).
  La capacité du condensateur est donc ici de 390 pF ;
- **J** : tolérance de 5% ;
- **5** : tension de service de 500 V ;
- **O** : température de -55 à +125°C ;
- **S** : connexions droites.

Military code Ce code est destiné aux composants à usage militaire. Il est proche du code EIA, qui l'utilise en partie, mais ne commence pas par la lettre « R ». Les principales différences concernent la tension de service et la résistance aux vibrations.

Marquage par points de couleur Ce marquage n'est plus guère utilisé est ne sera pas détaillé ici.

2.4 Les condensateurs de découplage

On parle également couramment, par abus de langage, de capacités de découplage.

Lors de la transition d'un circuit logique (passage d'une broche d'un état à l'autre), un courant transitoire ΔI circule dans le circuit d'alimentation (voir figure 18). Cela provoque une chute de tension aux bornes de son inductance équivalente, qui existe toujours, la ligne d'alimentation étant un fil, mais qui

est généralement négligée. Cette chute de tension peut être relativement importante (plusieurs volts), au point de perturber le fonctionnement des circuits logiques. De plus, l'inductance du circuit d'alimentation a tendance à s'opposer aux brusques variations de courant ce qui a pour conséquence un allongement du temps de transition du circuit logique, comme illustré figure 19.

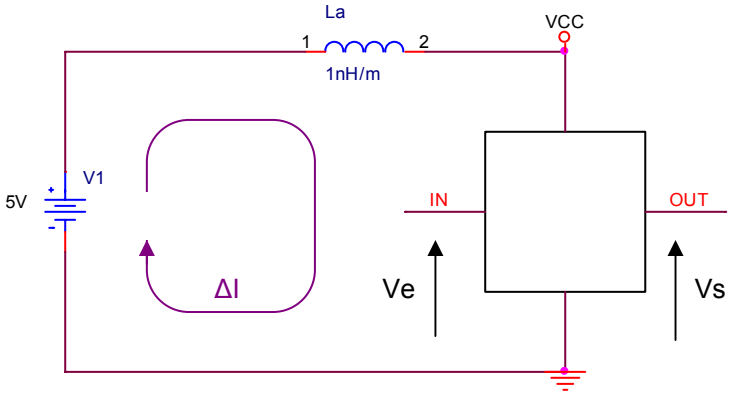


FIG. 18 – Alimentation d'un circuit sans capacité de découplage

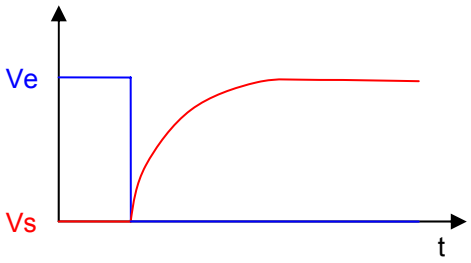


FIG. 19 – Tensions aux bornes d'un circuit logique sans découplage

L'ajout d'une capacité de découplage aux bornes de l'alimentation du circuit logique permet de limiter ces phénomènes, en agissant comme une source locale de charge et en diminuant l'impédance équivalente, sur la figure 20. Elle joue alors le rôle de stabilisateur de tension et contribue à réduire la boucle de circulation du courant transitoire ΔI, également à l'origine de rayonnement électromagnétique.

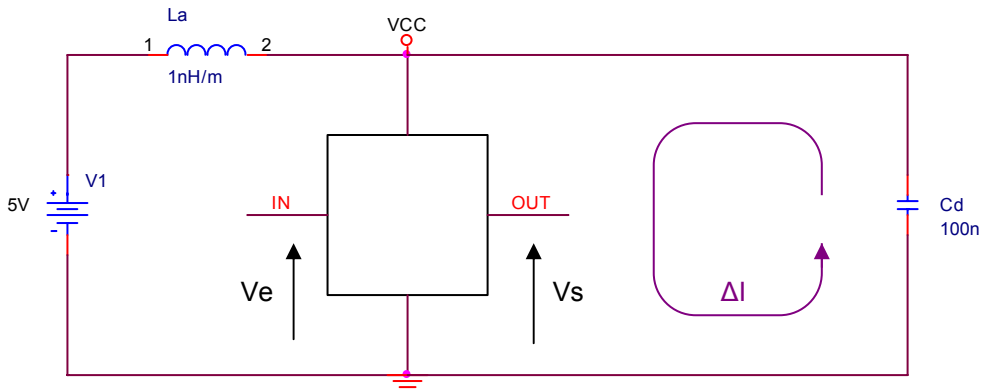


FIG. 20 – Alimentation d'un circuit avec capacité de découplage

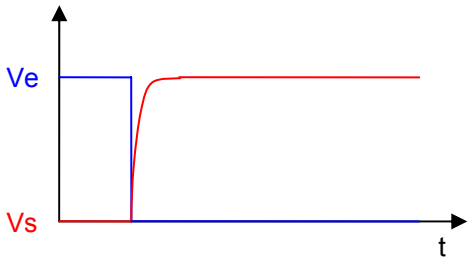


FIG. 21 – Tensions aux bornes d'un circuit logique avec découplage

Le condensateur de découplage va en plus permettre une commutation rapide des circuits logiques en fournissant l'énergie requise par le brusque appel de courant ΔI lors de la commutation (figure 21).

On comprend que pour qu'une capacité de découplage soit efficace, il faut la placer au plus prêt de l'alimentation du circuit logique, afin d'avoir l'inductance la plus faible possible entre l'alimentation du circuit logique et le condensateur. Ainsi, il est clair que chaque circuit logique doit avoir sa propre capacité de découplage.

Une valeur de capacité de découplage couramment utilisée est 100 nF.

2.5 Les condensateurs de lissage

Un condensateur de lissage (en anglais « smoothing capacitor »), comme son nom l'indique, « lisse » la tension, c'est-à-dire qu'il agit comme un réservoir, permettant de compenser les variations de tension. Un condensateur de lissage est ajouté sur l'alimentation, en parallèle à une charge, figure 22.

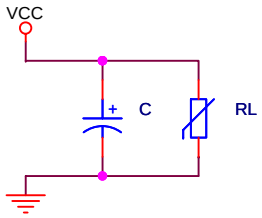


FIG. 22 – Mise en place d'un condensateur de lissage

La figure 23 montre comment la présence d'un condensateur, de capacité relativement importante, lisse la tension : la courbe 1 représente la tension d'alimentation VCC sans condensateur de lissage et la courbe 2 la tension aux bornes du condensateur.

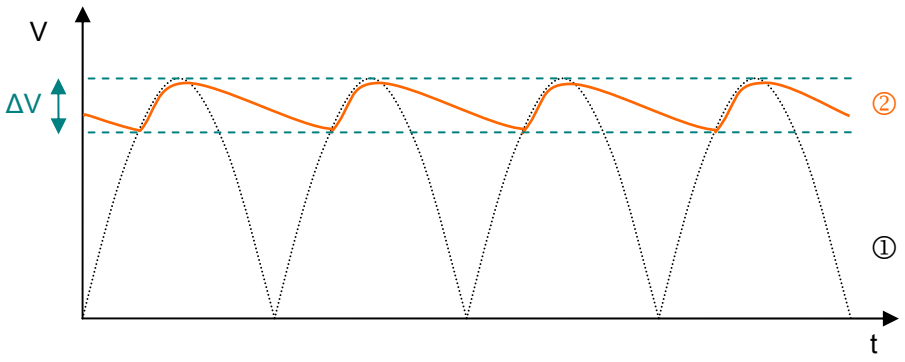


FIG. 23 – Lissage de la tension avec un condensateur de lissage

La capacité C est donnée par la formule suivante :

$$C = \frac{I}{\Delta V \times f}$$

où

– I est le courant provenant de l'alimentation ;
– ΔV est la variation maximale de tension que l'on veut avoir ;
– f est la fréquence du signal, ou l'ordre de grandeur de la fréquence de fluctuation du signal, fluctuations qui peuvent être dues aux transitions des circuits logiques (voir les condensateurs de découplage).

Si l'on ne veut pas que la variation de tension n'excède les 10% de la tension maximale, C est donné par :

$$C = \frac{I}{0,1 \times V_{max} \times f}$$

Si $V_{max} = 5$ V, $I = 100$ mA et $f \approx 1$ MHz, on trouve C ≈ 200 nF, ce qui est bien l'ordre de grandeur des capacités de découplage étudiées précédemment.

Si $V_{max} = 12$ V, $I = 0{,}5$ A et $f \approx 100$ Hz, on trouve C $\approx 4\,200$ μF, qui conviendrait pour stabiliser la tension aux bornes de batteries par exemple, avant un régulateur de tension. Évidemment, le condensateur ne compensera toutefois pas la lente baisse de tension de la batterie dû à sa décharge.

Ces valeurs de capacité ne sont que des ordres de grandeur, mais permettent de mieux appréhender pourquoi l'on met tel capacité à tel endroit, notamment au niveau de l'alimentation.

Maintenant vous pourriez vous dire que qui peut le plus, peut le moins. Ainsi, un condensateur de capacité élevée devrait en principe également stabiliser des variations de tension très rapides. Mais en pratique ce n'est pas le cas ! Pourquoi ? Car les condensateurs de grosse capacité sont forcément des condensateurs électrochimiques, qui sont caractérisés par une certaine inertie. Leur temps de réaction est en fait trop élevé pour qu'ils puissent suivre des fluctuations rapides de tension, ce qui les rend inefficaces dans les hautes fréquences. C'est pourquoi on utilise en parallèle à ceux-ci des condensateurs de capacité plus faible, généralement des céramiques, pour stabiliser les hautes fréquences, comme illustré par la figure 24.

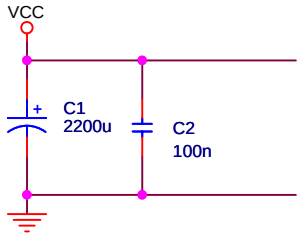


Fig. 24 – Stabilisation de l'alimentation (classique)

3 Les diodes

3.1 Repérage anode - cathode

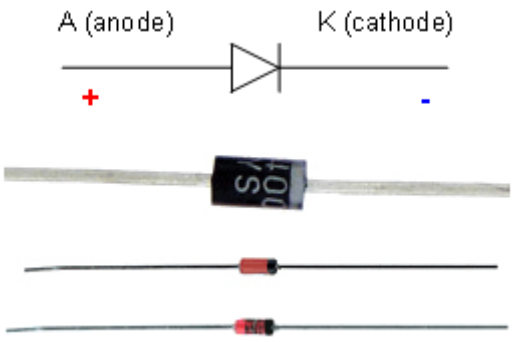


Fig. 25 – Repérage de l'anode et de la cathode des diodes

La cathode est généralement repérée par une bague. Lorsqu'il y a plusieurs bagues de couleur sur la diode, c'est la bague la plus grosse qui repère la cathode, ou alors les bagues sont plus proches de la

cathode que de l'anode.

3.2 Marquage des diodes

3.2.1 Le code JEDEC

Code couleur	Valeur
Noir	0
Marron	1
Rouge	2
Orange	3
Jaune	4
Vert	5
Bleu	6
Violet	7
Gris	8
Blanc	9

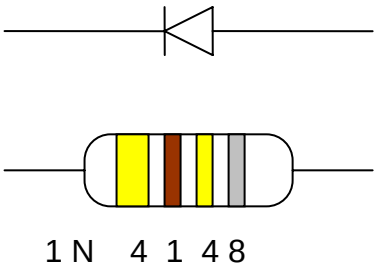


FIG. 26 – Le codage JEDEC

FIG. 27 – Une diode 1N4148 avec le codage JEDEC

4 Les circuits intégrés

4.1 Les familles de circuits logiques

Les deux technologies les plus connues pour les circuits intégrés sont les technologies Transistor Transistor Logic (TTL) et Complementary Metal Oxide Semi-conductor (CMOS). Actuellement, la technologie CMOS est la plus utilisée au détriment de la technologie TTL.

74 : série commerciale (0° à 70°C)
54 : série militaire (-55° à 125°C)

$\overbrace{XX}$ Identifiant constructeur — $\overbrace{74/54}$ — $\overbrace{XXX}$ Famille techno. — Référence fonction $\overbrace{XXX}$ — $\overbrace{XXX}$ Packaging

FIG. 28 – Photo d'un 74LS00

On lit sur la figure 28 :
- Identifiant constructeur : SN. Série standard propre à Texas Instruments ;
- Série : 74 - série commerciale ;
- Famille technologique : LS. Il s'agit donc d'un composant de technologie TTL, le LS signifiant Low-power Schottky (voir tableau 15) ;
- Référence fonction : 00. Ce numéro permet d'identifier la fonction réalisé par le circuit logique. Ici il s'agit d'un quadruple NAND ;
- Type de boitier : NDS. Identifie le packaging du composant (se rapporter à la datasheet du composant pour plus de détails).

4.1.1 Technologie TTL

Désignations et sous familles

Sous famille (séries commerciales)	Tension d'alim.	Temps prop.	Conso.	Fréq. max.
74*	5 V ± 5%	10 ns	10 mW	25 MHz
74L (Low-power)*		33 ns	1 mW	5 MHz
74H (High-speed)*		6 ns	22 mW	
74S (Schottky)		3 ns	19 mW	75 MHz
74LS (Low-power Schottky)		9 ns	2 mW	30 MHz
74AS (Advanced Schottky)		7 ns	5 mW	
74ALS (Advanced Low-power Schottky)		7 ns	5 mW	
74F(AST) (Fast (Advanced Schottky))		6 ns	5 mW	100 MHz

Tab. 15 – Liste des sous familles TTL

* Devenue aujourd'hui partiellement, voir totalement, obsolète

4.1.2 Technologie CMOS

Désignations et sous familles

Sous famille (séries commerciales)	Tension d'alim.	Temps prop.	Conso.	Fréq. max.
4000*	3 à 15 V	40 ns	0,1 mW	12 MHz
74C (CMOS pin-compatible with TTL)*	5 V ± 10%	50 ns	0,1 mW	50 MHz
74HC (High-speed C)	2 à 6 V	18 ns	0,06 mW	50 MHz
74HCT (HC TTL compatible)	5 V ± 10%	18 ns	0,06 mW	50 MHz
74AC (Advanced C)	2 à 6 V	5 ns	0,1 mW	70 MHz
74ACT (AC TTL compatible)	6 V	5 ns	0,1 mW	70 MHz
74AHC (Advanced HC)	2 à 6 V	5 ns	0,1 mW	70 MHz
74AHCT (Advanced HCT)	6 V	5 ns	0,1 mW	70 MHz
74FCT (Fast CMOS TTL compatible)	6 V	5 ns	0,1 mW	70 MHz

Tab. 16 – Liste des sous familles CMOS

* Devenue aujourd'hui partiellement, voir totalement, obsolète
La liste des sous familles est loin d'être exhaustive !

4.2 Compatibilité TTL - CMOS

Il est toujours préférable de ne pas mélanger les technologies sur un même circuit électronique, et travailler exclusivement avec du CMOS ou avec du TTL, dans la mesure où l'on trouve généralement des équivalents dans chaque technologie.

Si toutefois nous sommes amené à mélanger plusieurs technologies, il faut être prudent et s'assurer entre autre que les tensions de seuil sont compatibles d'une technologie à l'autre (mais cela n'est pas forcément suffisant).

4.2.1 TTL vers CMOS

Il n'y a pas compatibilité sauf si la porte CMOS est du type TTL compatible (typiquement, les 74HCT) : la tension de sortie minimale au niveau haut des circuits TTL, de 2,4 V, n'est pas compatible avec la tension minimale d'entrée des circuits CMOS, qui vaut 3,5 V. Sinon, il est possible d'assurer la compatibilité en ajoutant une résistance de tirage entre la sortie du TTL et l'entrée du CMOS, comme sur la figure 29 (ce qui, comme nous l'avons vu, n'est pas nécessaire entre deux composants de la même technologie). Il est également possible d'intercaler une porte TTL compatible entre le TTL et le CMOS.

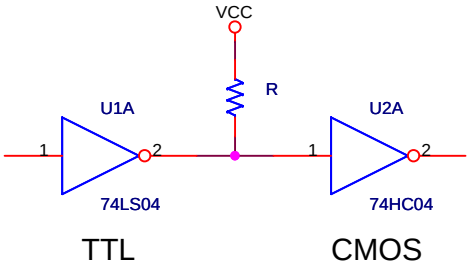


FIG. 29 – TTL vers CMOS avec résistance de tirage

On doit avoir

$$R > \frac{(VCC - V_{OL_{max}}^{TTL})}{I_{OL}^{TTL} - n \times I_{IL}^{CMOS}} = \frac{(5 - 0,5)}{0,008 - n \times 0} \approx 600\ \Omega$$

où n est le nombre d'entrées CMOS connectés à la sortie TTL. En pratique, une résistance de 1 à 10 kΩ convient.

4.2.2 CMOS vers TTL

Il y a compatibilité en règle générale.

4.3 Les entrées inutilisées

Si vous avez bien compris à quoi sert une résistance de tirage, vous savez qu'il ne faut jamais laisser de broche d'entrée d'un circuit logique en haute impédance. Une quantité très faible de charge peut provoquer le changement d'état de l'entrée, par exemple à l'approche d'un objet ou d'un doigt. De plus, la variation des charges au niveau de l'entrée risque de la mettre dans un état instable et la faire constamment basculer d'un état logique à l'autre, ce qui va provoquer des appels de courant sur le circuit d'alimentation perturbant les autres circuits logiques, et le composant va chauffer anormalement.

4.4 L'électricité statique

La plupart des circuits intégrés sont sensibles à l'électricité statique (particulièrement les CMOS) et peuvent être endommagés en les manipulant, si vous les touchez avec vos doigt et que vous êtes chargé en électricité statique, provenant de vos habits par exemple. Les circuits sensibles à l'électricité statique sont généralement vendu dans des sachets antistatiques, comme ceux représentés figure 30.

FIG. 30 – Sachets antistatiques

Afin de garantir le bon état des circuits intégrés, il est important de les conserver dans leurs emballages antistatiques adaptés jusqu'à leur utilisation effective.

Il peut également être prudent de se décharger en électricité statique avant de les manipuler, en touchant un conducteur relié à la terre, comme une conduite d'eau en métal par exemple.

5 Les PICs

Avant toute chose, voici la documentation minimum dont vous aurez besoin pour apprendre à connaitre les PICs :

– La datasheet du PIC (nous nous intéressons ici à la famille PIC18FXX2) [10] ;
– MPLAB C18 C Compiler Libraries (librairies disponibles en C pour les PICs) ;
– Éventuellement, un manuel de langage C à porté de main.

Logiciels nécessaires :

– MPLAB Integrated Development Environment (IDE)[1] (Integrated Development Environment) : environnement de développement en C ;
– MPLAB C18[2] : compilateur C ;
– ICPROG[3] : permet de transférer le programme compilé sur le PIC, via le programmateur.

5.1 Installation de l'environnement de développement

Pour mettre en place l'environnement de développement sur PC pour notre PIC, il y a donc trois logiciels à installer. Le mieux est d'installer dans l'ordre, MPLAB IDE, MPLAB C18 (en prenant garde de ne pas mettre à jour les variables locales, nous le ferons manuellement plus tard) et ICPROG.

5.1.1 Intégrer MPLAB C18 à MPLAB IDE

Une fois MPLAB IDE et MPLAB C18 installés, il faut configurer IDE pour utiliser le compilateur C18. Pour se faire, voici la procédure à suivre, dans MPLAB IDE :

1. Ouvrir le menu « Project » → « Set Language Tool Locations » et développer la section « Microchip C18 Toolsuite ».

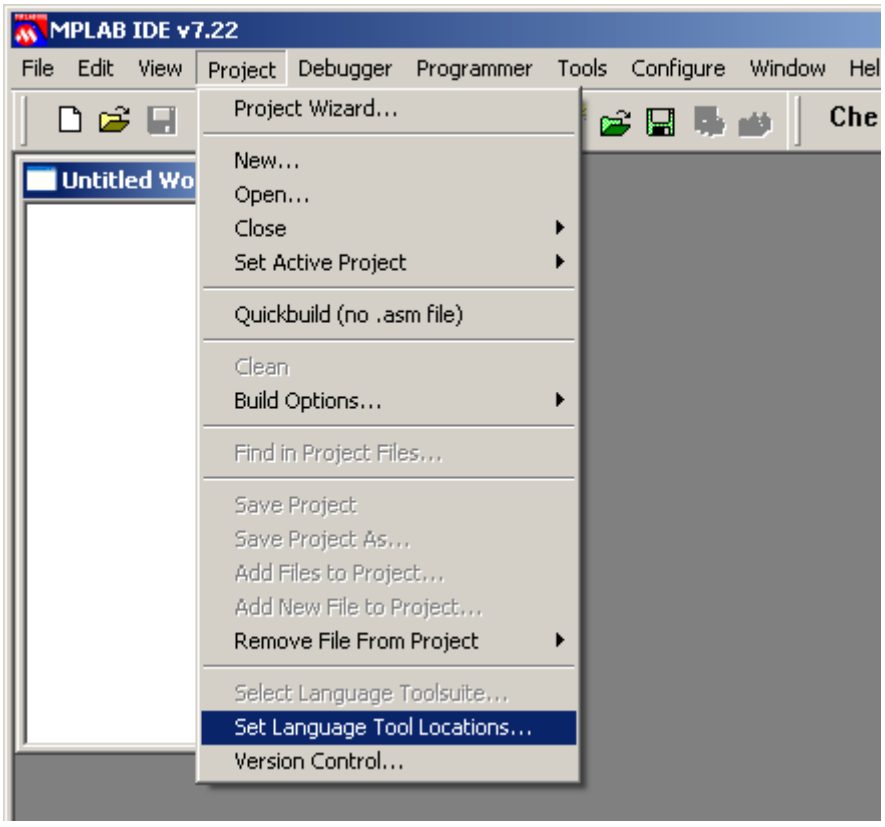


Fig. 31 – Le logiciel MPLAB IDE

On considère par la suite que le répertoire d'installation de MPLAB IDE est `C:\Program Files\MPLAB IDE` et que celui du compilateur MPLAB C18 est `C:\mcc18` ;

[1] Téléchargeable gratuitement sur le site de Microchip : http://www.microchip.com/
[2] http://www.microchip.com/
[3] http://www.ic-prog.com/

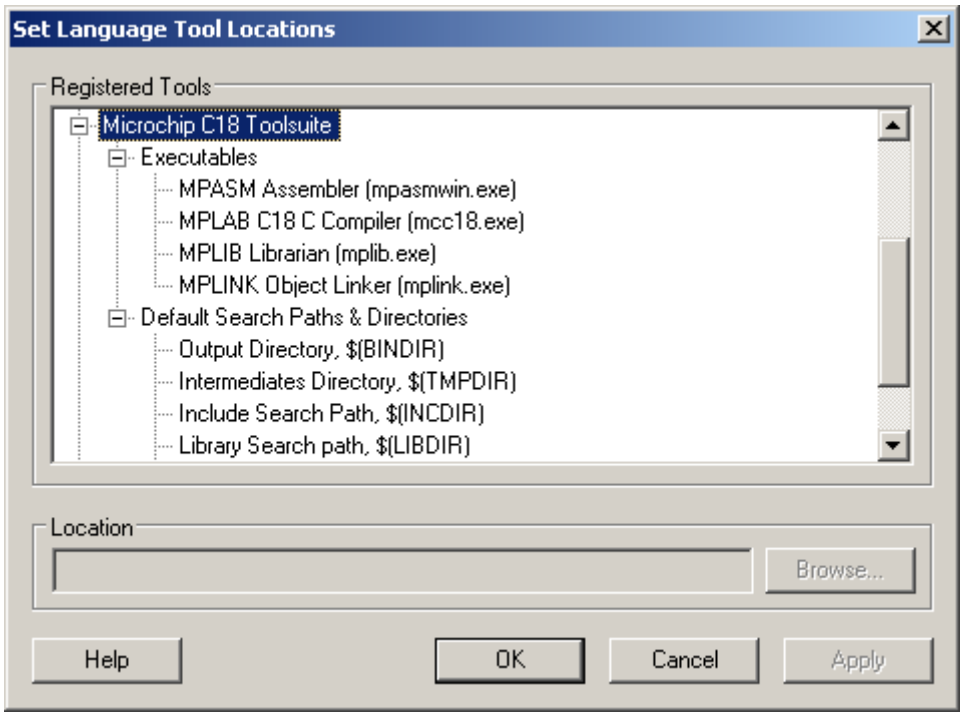


FIG. 32 – La boite de dialogue « Set Language Tool Locations »

2. Complétez alors les différentes lignes de cette section comme suit :
   - MPASM Assembler (mpasmwin.exe) : `C:\Program Files\MPLAB IDE\MPASM Suite\MPASMWIN.EXE` ;
   - MPLAB C18 C Compiler (mcc18.exe) : `C:\mcc18\bin\mcc18.exe` ;
   - MPLIB Librarian (mplib.exe) : `C:\mcc18\bin\mplib.exe` ;
   - MPLINK Object Linker (mplink.exe) : `C:\mcc18\bin\mplink.exe` ;
   - Include Search Path, $(INCDIR) : `C:\mcc18\h` ;
   - Library Search Path, $(LIBDIR) : `C:\mcc18\lib` ;
   - Linker-Script Search Path, $(LKRDIR) : `C:\mcc18\lkr`.

Désormais, IDE sait où trouver les librairies et le compilateur.

5.1.2 Créer un projet avec MPLAB IDE

1. Dans MPLAB IDE, ouvrir le menu « Project » → « New ». Donner un nom et un répertoire au projet ;

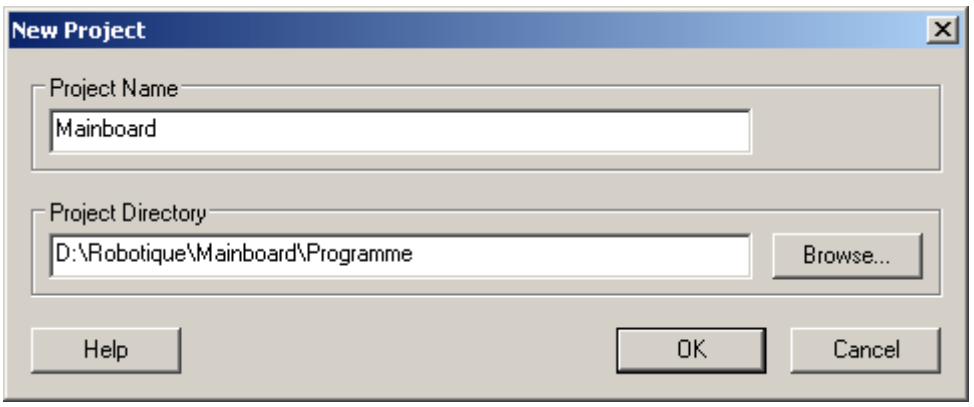


FIG. 33 – Fenêtre « nouveau projet » dans MPLAB IDE

2. Ouvrir le menu « Project » → « Build Options » → « Projet » ;

3. Dans l'onglet « General », renseignez les champs suivant, comme illustré figure 34 :
   – Include Search Path, $(INCDIR)
   – Library Search Path, $(LIBDIR)
   – Linker-Script Search Path, $(LKRDIR)
   Astuce : pour chaque champ, cliquez sur « Browse », puis appuyez sur Entrée, cela le renseignera avec la valeur par défaut que nous avons configurées auparavant dans « Language Tool Locations » ;

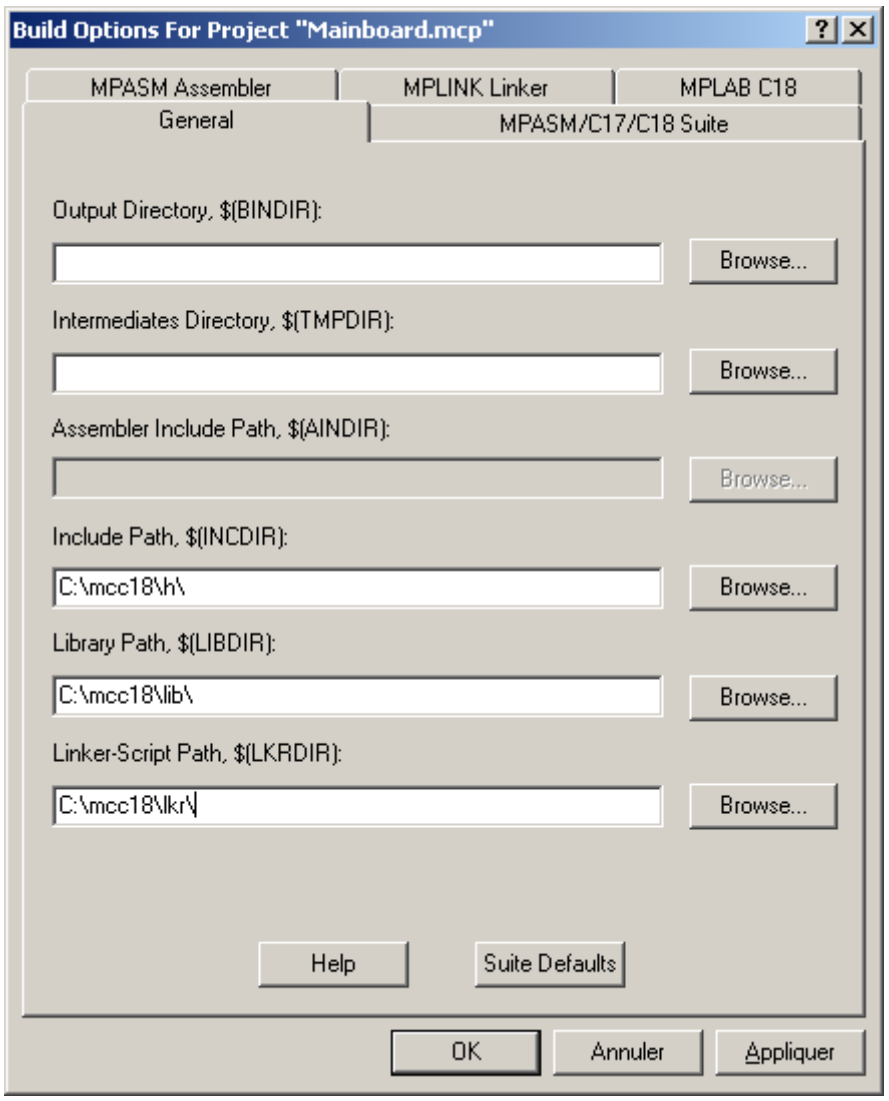


Fig. 34 – Fenêtre « Build Options » du projet

4. Dans l'onglet « MPLINK Linker » de la même fenêtre, vous pouvez si vous le souhaitez cocher la case « Generate map file ». Ainsi, le compilateur génèrera un fichier .map contenant toutes les variables utilisées dans le programme et leur adresse physique dans le PIC, ce qui peut avoir son utilité ;
5. Il nous faut ensuite ajouter le fichier de script du linker qui correspond au PIC pour lequel on écrira le programme (18f452.lkr dans notre cas), en cliquant du bouton droit de la souris sur l'intitulé « Linker Scripts » du volet contenant les fichiers du projet, comme illustré figure 35. Les fichiers .lkr se trouvent dans le sous-répertoire `lkr` du répertoire d'installation de MPLAB C18 (par exemple, `C:\mcc18\lkr`) ;

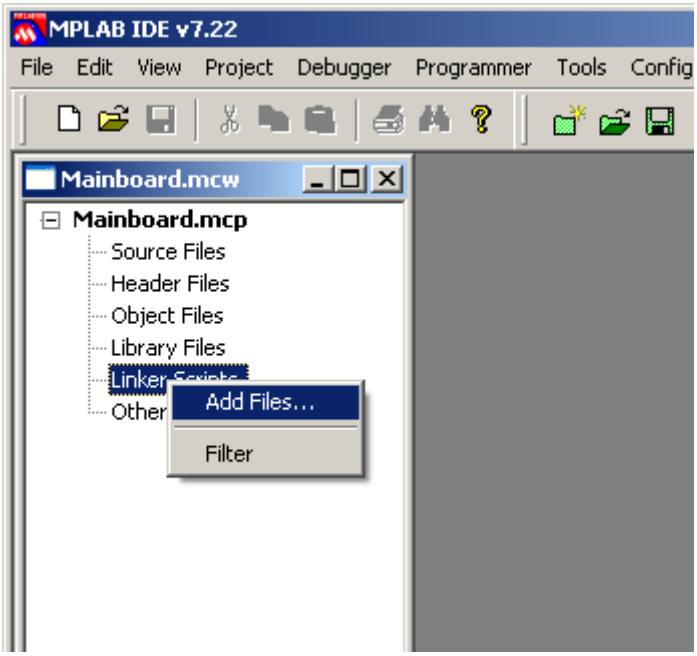


FIG. 35 – Ajout du script du linker dans MPLAB IDE

6. Créer un fichier source via le menu « File » → « New » et enregistrez-le dans le répertoire du projet avec l'extension « .c » (à l'aide du menu « File » → « Save As... »). Ajoutez ensuite ce fichier à l'intitulé « Source Files » du volet contenant les fichiers du projet, en procédant de la même manière que pour le fichier de script du linker ;
7. Configurer le projet pour qu'il utilise le langage C18 : ouvrez le menu « Project » → « Select Language Toolsuite » et choisissez « Microchip C18 Toolsuite » comme « Active Toolsuite » dans la boite de dialogue qui apparaît alors ;
8. Préciser le modèle de PIC du projet, dans le menu « Configure » → « Select Device... », sélectionnez le PIC que vous allez utiliser (nous utiliserons par la suite le PIC18F452).

Vous avez désormais en principe correctement configuré votre projet. Il ne reste plus qu'à écrire le programme. Pour le compiler, sélectionnez « Project » → « Make » ou « Project » → « Build all ». Un fichier hexadécimal .hex est alors créé dans le répertoire du projet (ou dans le répertoire Output si vous l'avez spécifié). C'est ce fichier qu'il faudra par la suite transférer sur le PIC.

5.1.3 Transfert du programme sur le PIC

Pour transférer le programme sur le PIC, nous allons utiliser ICPROG. La procédure pour programmer le PIC18F452 décrire ci-après suppose que les options de configuration sont au départ celles par défaut. Il vous est toujours possible de réinitialiser la configuration par défaut dans le menu « Settings » → « Clear Settings ».

Installation de ICPROG sous Windows NT/2000/XP Si vous êtes sous Windows NT/2000/XP, en plus d'avoir les droits administrateur, il est nécessaire d'installer un driver spécifique pour que le programme fonctionne. Pour cela, il vous faut télécharger le fichier icprog.sys correspondant à votre version d'ICPROG, sur le site de l'auteur, et le placer dans le même répertoire que le programme.

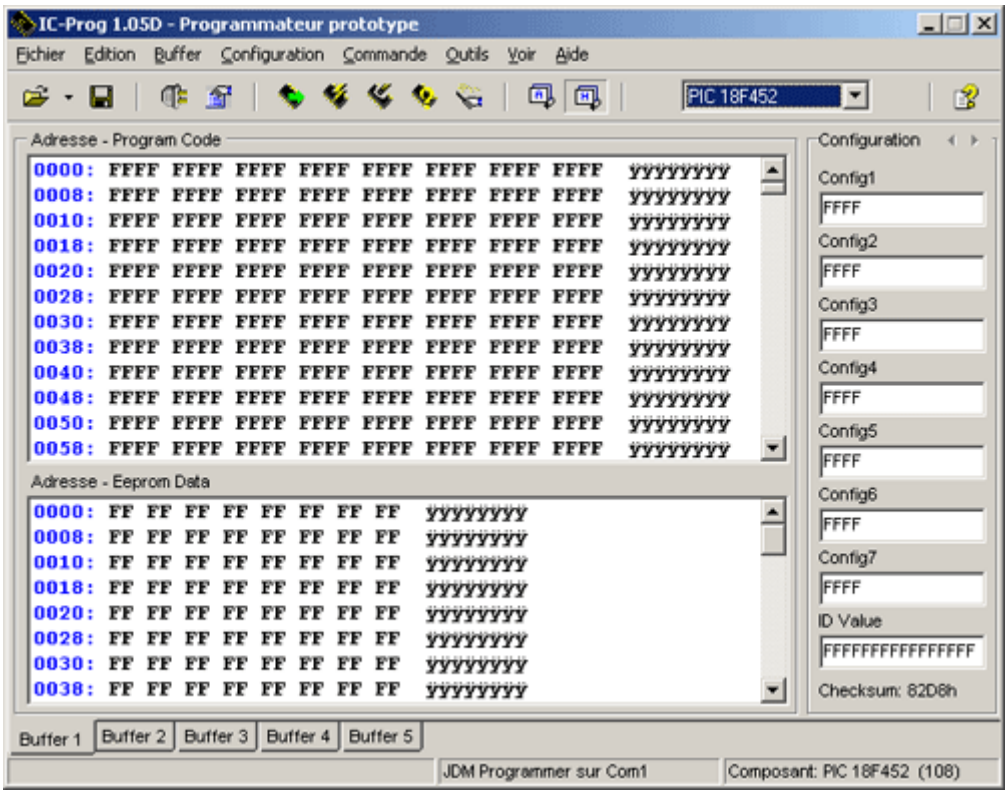


FIG. 36 – Le programme ICPROG (configuré pour être en langue française)

Vous devez ensuite configurer ICPROG pour utiliser ce driver. Pour cela, allez dans le menu « Settings » → « Options », puis dans l'onglet « Misc ». Cochez alors la case « Enable NT/2000/XP Driver » (voir figure 37). Relancer le programme, qui vous demandera alors si vous désirez installer le driver, auquel vous répondrez donc oui.

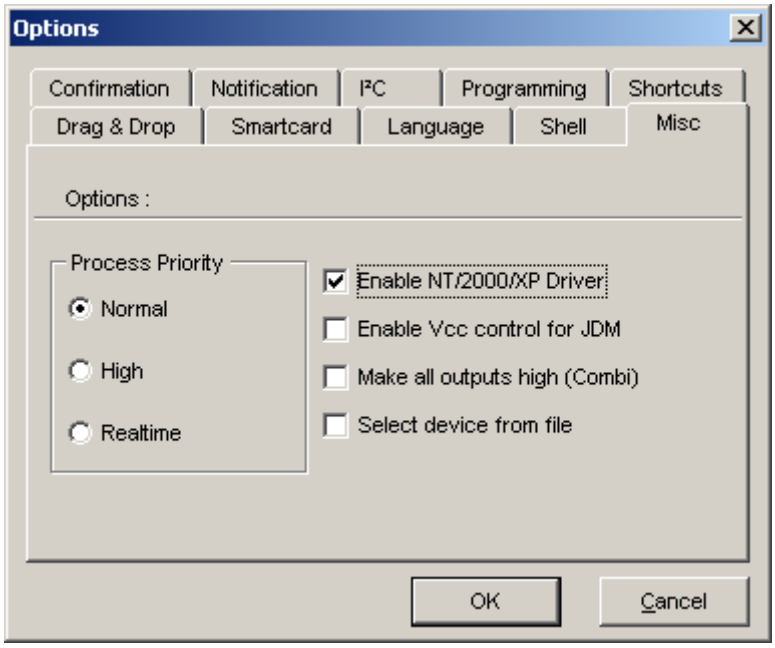


FIG. 37 – Activer le driver Windows NT avec ICPROG

Vous pouvez en profiter pour éventuellement mettre ICPROG en français, dans l'onglet « Language » de la même fenêtre et nous supposerons pour la suite qu'ICPROG est configuré en français.

Allez ensuite dans le menu « Configuration » → « Hardware » et sélectionnez « Windows API » dans

l'encadré intitulé « Interface » (voir figure 38). C'est indispensable pour que le programme utilise le driver que nous venons d'installer.

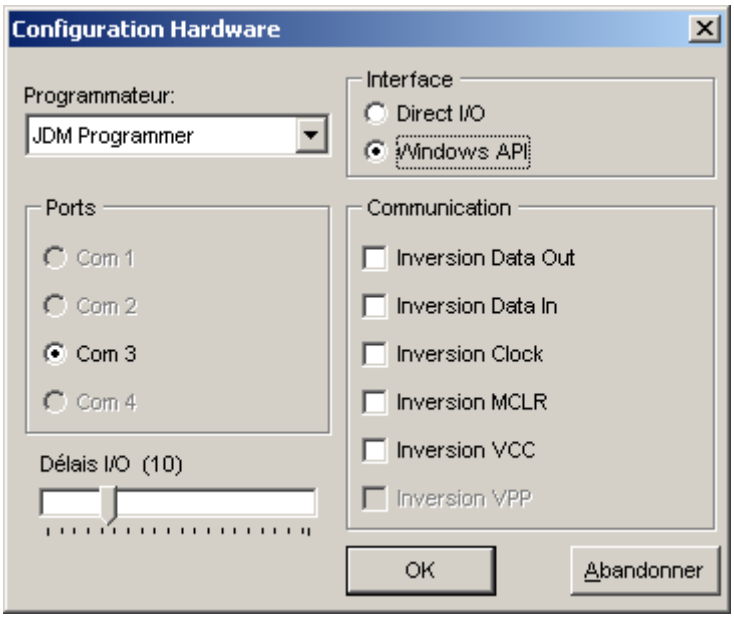


FIG. 38 – Fenêtre de configuration Hardware dans ICPROG

Choix du programmateur Vous trouverez un schéma de programmateur série de type JDM figure 39. Ce programmateur a l'avantage d'être très simple et il permet de programmation de PIC au format DIP18 et au format PLCC44 (notamment pour le PIC18F452, auquel nous nous intéresserons plus particulièrement).

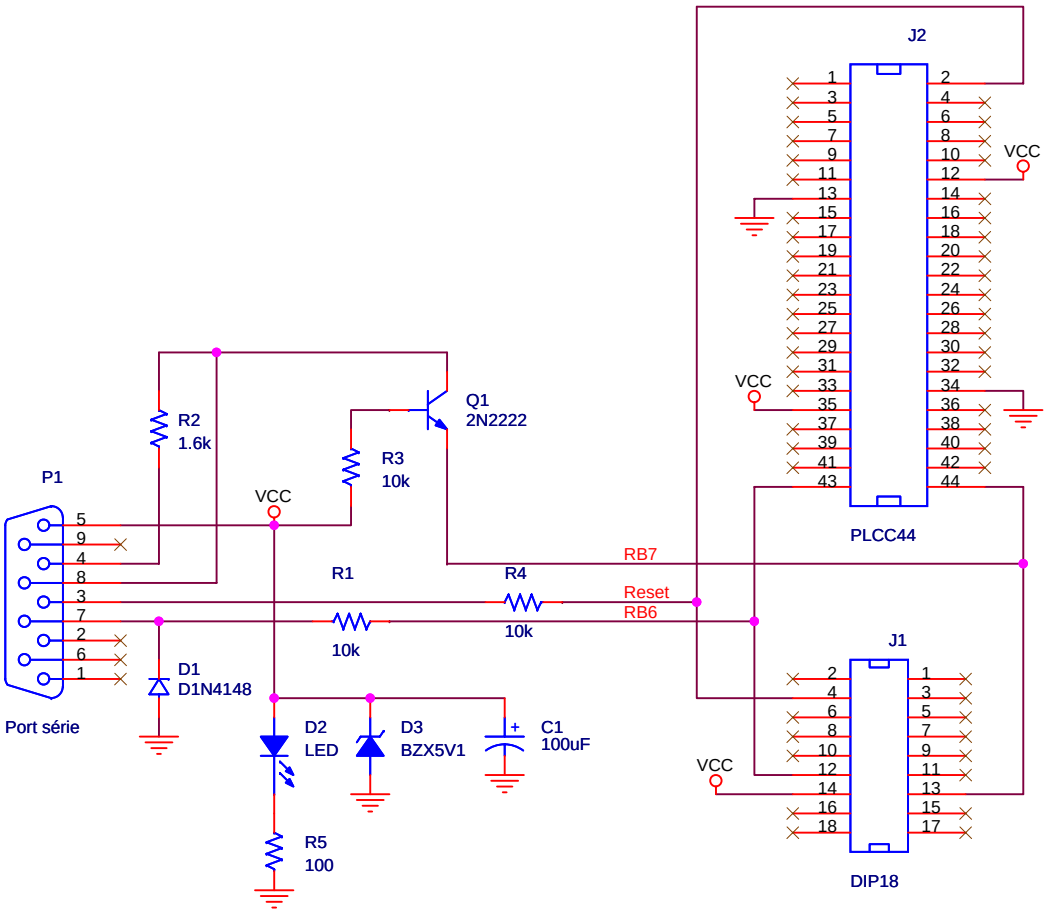


FIG. 39 – Programmateur de PIC sur port série, type JDM

Transfert du programme Avant de transférer le programme, il vous faut correctement configurer ICPROG pour le programmateur et le PIC que vous utilisez. Cela se fait dans la fenêtre de configuration Hardware d'ICPROG. Pour le programmateur présenté précédemment, la configuration est celle par défaut, illustrée figure 38. Pour le PIC, il vous suffit de sélectionner le bon modèle dans le menu déroulant de la fenêtre principale, en surbrillance, en haut à droite figure 36.

Ensuite, vous n'avez plus qu'à charger le fichier .hex généré par le compilateur (« Fichier » → « Ouvrir Fichier... ») et lancer la programmation (« Commande » → « Tout Programmer », ou F5), après s'être assuré que le PIC est correctement en place sur le programmateur (attention au sens) et que celui-ci est bien connecté au port série du PC. Vérifiez que le bon port COM est sélectionné dans la fenêtre figure 38.

Il ne doit y avoir qu'une seule fenêtre d'ICPROG ouverte et aucun autre programme ne doit accéder au port COM sur lequel est connecté le programmateur durant la lecture ou la programmation du PIC.

Afin de contrôler le bon déroulement de la programmation et de gagner du temps, vous pouvez activer l'option « Vérifier pendant la programmation », de l'onglet « Programmation », du menu « Configuration » → « Options ». Ainsi s'il y a un problème, vous n'aurez pas à attendre la fin de la programmation pour le savoir.

5.2 Caractéristiques du PIC18F452

5.2.1 Le composant

Fig. 40 – PIC18F452-I/P (format DIP, 40 broches)

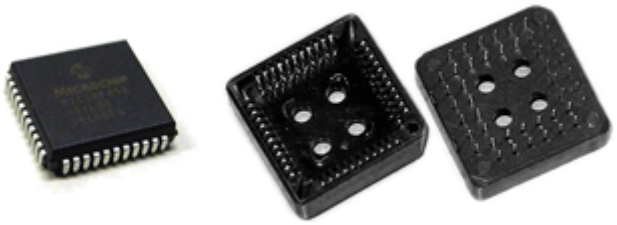


Fig. 41 – PIC18F452-I/L + support (format PLCC, 44 broches, dont 4 broches non connectés)

Le PIC18F452 au format PLCC44 requiert un support spécial, représenté figure 41. Noter que pour retirer le PIC du support proprement et sans risque, il existe une pince spécialement prévu à cet effet, représentée figure 42 (pour extraire le composant du support, il faut presser la pince et ne jamais la tirer).

Fig. 42 – Pince PLCC44

5.2.2 L'horloge (CLOCK)

L'horloge synchronise toutes les actions du processeur (système synchrone).
Il y a 8 types d'oscillateurs possibles et 3 schémas d'horloge :

- Résonateur à quartz ou à céramique ;
- Oscillateur RC - Résistance-Capacité ;
- Horloge externe.

⇒ Voir datasheet §2 [10].

5.2.3 L'initialisation (RESET)

Le RESET force l'état initial du circuit.
Il y a 6 sources de RESET distinctes, le RESET de base étant le « Power-on Reset » (POR), qui initialise le circuit à la mise sous tension.

⇒ Voir datasheet §3 [10].

5.2.4 La mémoire

Il y a trois types de mémoire :

– 32 Ko de mémoire programme, ou mémoire FLASH : contient le programme ;
– 1536 octets de mémoire vive, ou RAM (Random Access Memory) : contient les données en cours de traitement ;
– 256 octets d'EEPROM : mémoire rapide de stockage de données.
⇒ Voir datasheet §4, §5 et §6 [10].

5.2.5 Les entrées/sorties

– 34 entrées/sorties réparties sur 5 ports ;
– 8 canaux de conversion A/N (analogique/numérique) ;
– 2 modules CCP : Capture/Compare/PWM (PWM - Pulse With Modulation = MLI - Modulation de Largeur d'Impulsion) ;
– 18 sources d'interruption.

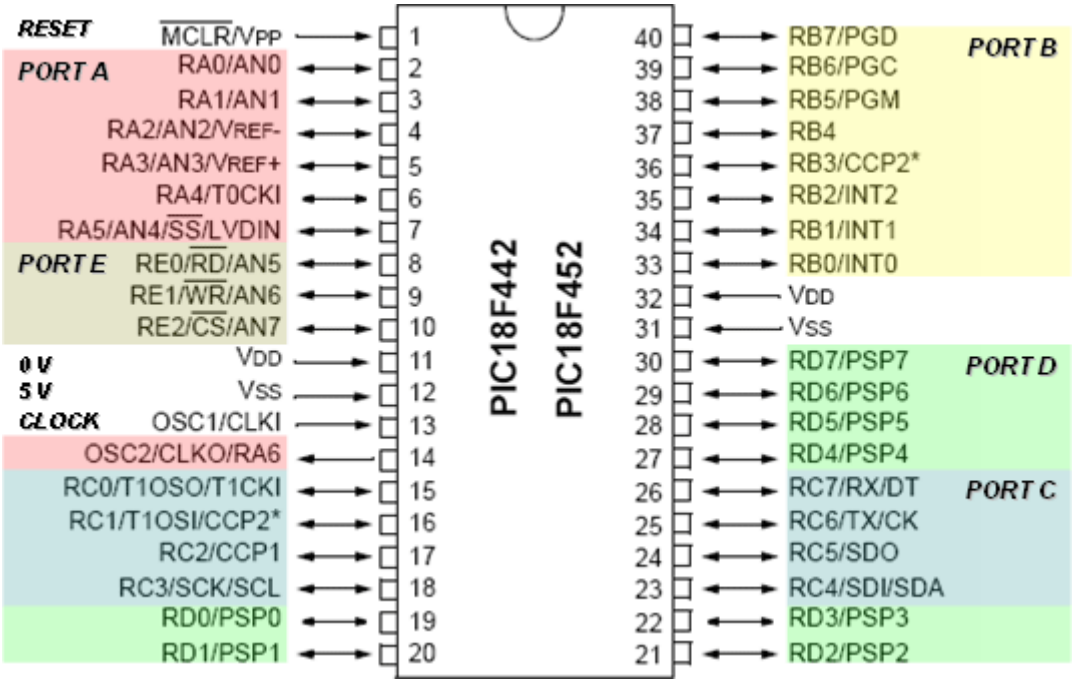


FIG. 43 – Les PINs du PIC18F452 (pour le format DIP)

⇒ Voir datasheet §9, §14 [10].

5.2.6 Les timers

Il y a 4 timers disponibles. Un timer a fonction de compteur programmable, le comptage pouvant se faire sur 8 ou 16 bits. Il y a deux sources de comptage :
– Horloge du PIC ;
– Front montant/descendant sur l'entrée T0CKI ou T1CKI.
Les interruptions sont générées sur les overflow.
⇒ Voir datasheet §10 à §13 [10].

5.3 Un montage de base

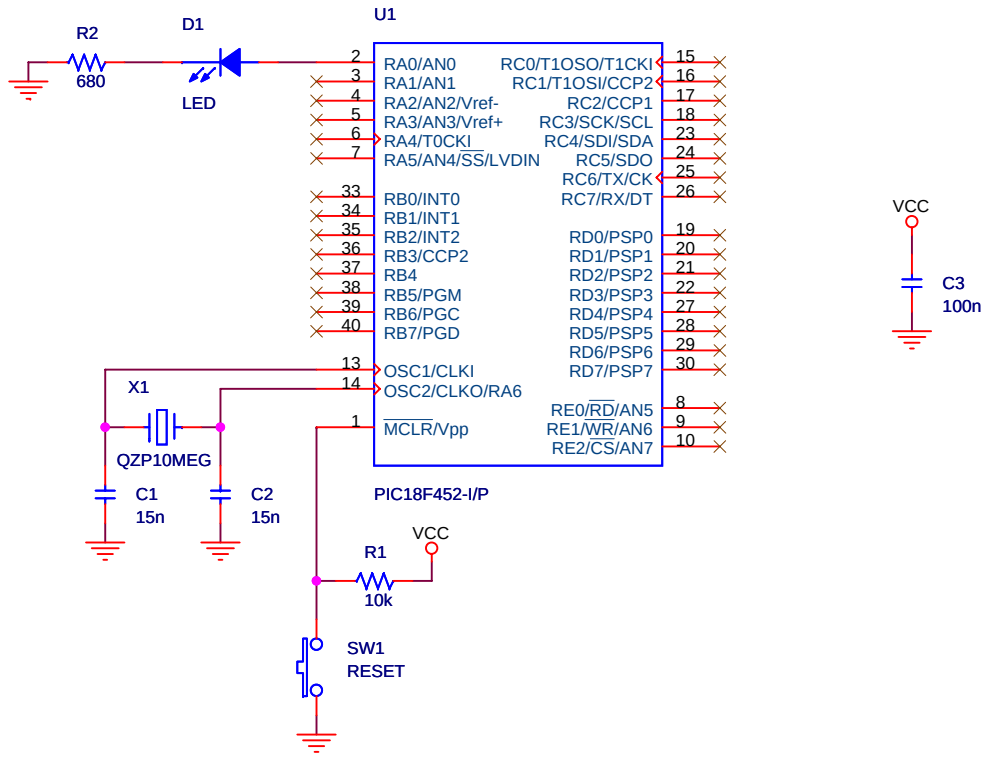


FIG. 44 – Montage de base du PIC18F452

Vous noterez la présence d'une capacité de découplage au niveau de l'alimentation (les broches d'alimentation du PIC ne sont pas représentées sur ce schéma, mais elles existent bien et le logiciel sait qu'elles sont reliées sur VCC et GND). Cette capacité devra être monté en pratique le plus proche possible du PIC.
Une résistance de tirage est utilisée sur le RESET (entrée $\overline{MCLR}$ du PIC). Il s'agit là de la circuiterie de reset la plus basique qu'il soit possible de faire.
Une résistance de limitation de courant est présente pour la LED.
Enfin, le quartz et les deux condensateurs assurent l'horloge du PIC.

5.4 La programmation

5.4.1 Faire clignoter une LED

Nous considérons le montage de la figure 44. Le programme suivant permet de faire clignoter la LED D1. C'est l'un des programmes les plus simples qui soit et qui est très pratique pour tester le bon fonctionnement des montages, qui devraient toujours avoir une LED témoin permettant d'afficher l'état du PIC.

Listing 1 – Programme faisant clignoter une LED

```
1  #include <p18f452.h>  // déclarations pour le PIC18F452
2  #include <delays.h>   // fonctions de délais
3
4  // Configuration
5  #pragma config OSC = HSPLL  // oscillateur HS, avec PLL (fréquence multipliée par 4)
6  #pragma config PWRT = ON    // power up timer (activé)
7  #pragma config WDT = OFF    // watchdog timer (desactivé)
8  #pragma config BOR = ON     // brown out reset (activé)
9  #pragma config BORV = 42    // brown out voltage (4.2V)
10 #pragma config LVP = ON     // low voltage ICSP (activé)
11
12 void main(void) {
13   // On configure le pin 0 du port A en sortie
```

```
14    TRISAbits.TRISA0 = 0;
15
16    while(1) {
17      // On allume la LED en mettant le pin 0 à l'état haut
18      PORTAbits.RA0 = 1;
19      // On attends environ 0,25s à 40Mhz
20      Delay10KTCYx(0);
21
22      // On éteint la LED en mettant le pin 0 à l'état bas
23      PORTAbits.RA0 = 0;
24      // On attends environ 0,25s à 40Mhz
25      Delay10KTCYx(0);
26    }
27  }
```

Algorithme L'algorithme, très simple, est présenté figure 45.

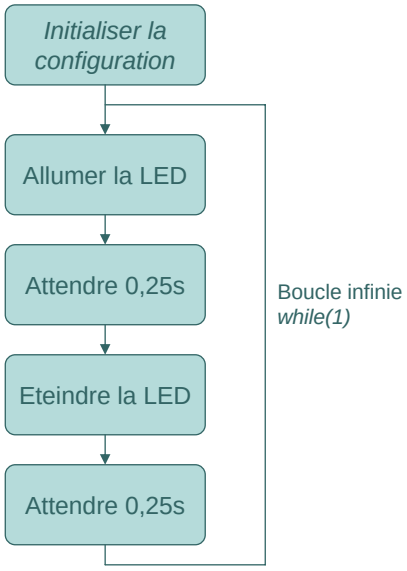


FIG. 45 – Algorithme du programme faisant clignoter une LED

Inclusion des librairies Pour ce programme, deux librairies sont nécessaires :
- « p18f452.h » : déclarations des constantes relatives au PIC18F452, requis pour PORTAbits.RA0.
  ⇒ Voir les fichiers *.h du répertoire « h » de MPLAB C18.
- « delays.h » : fonctions de délais, requis pour la fonction Delay10KTCYx().
  ⇒ Voir MPLAB C18 C Compiler Libraries §4.5.

5.4.2 Les interruptions

Un PIC n'est pas multitâches (comme tout processeur d'ailleurs, puisque le multitâche est une couche logiciel), cela signifie qu'un seul code peut être exécuté à la fois. Les interruptions permettent au processeur d'exécuter une code qui n'est pas dans la boucle principale du programme, en réponse à un évènement, qui est donc appelé une interruption, quelque soit le code entrain d'être exécuté.

Illustration du problème On veut exécuter un calcul prenant 1s tout en faisant clignoter la LED. Problème : durant le calcul, la LED ne clignote pas !

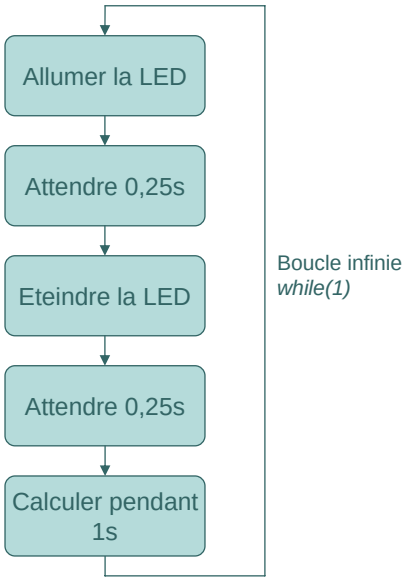


FIG. 46 – Algorithme sans interruption

L'idée Introduction d'une interruption, qui survient lors d'un évènement (configurable) :
- Overflow d'un timer ;
- Sur front montant/descendant d'une entrée INT0, INT1 ou INT2 ;
- Changement d'une entrée du port B ;
- ... .

⇒ Voir datasheet §8 [10].

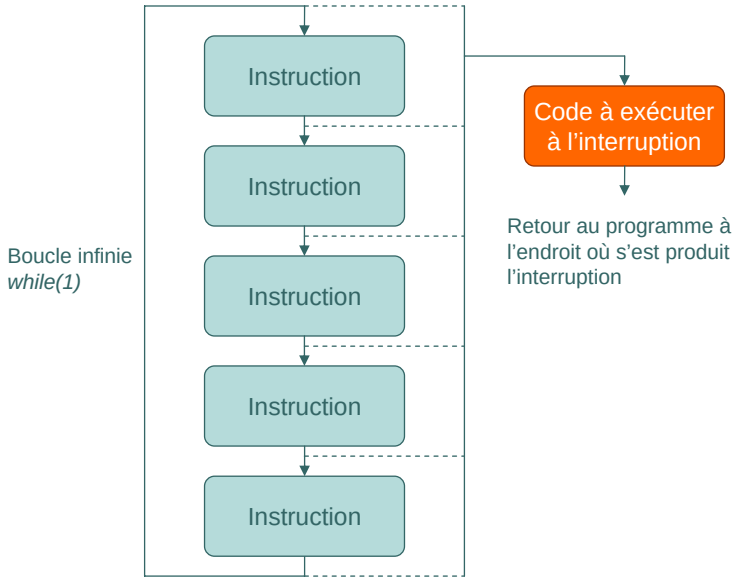


FIG. 47 – Algorithme avec interruption

Retour sur l'exemple de la LED : la solution est d'effectuer le calcul de 1s dans la boucle infinie et de générer une interruption toutes les 0,25s pour allumer ou éteindre la LED. Cette interruption sera générée par un timer par exemple.

Les priorités Introduction de 2 niveaux de priorité, haute et basse. Une interruption de priorité basse peut elle-même être interrompue par une interruption de priorité haute.

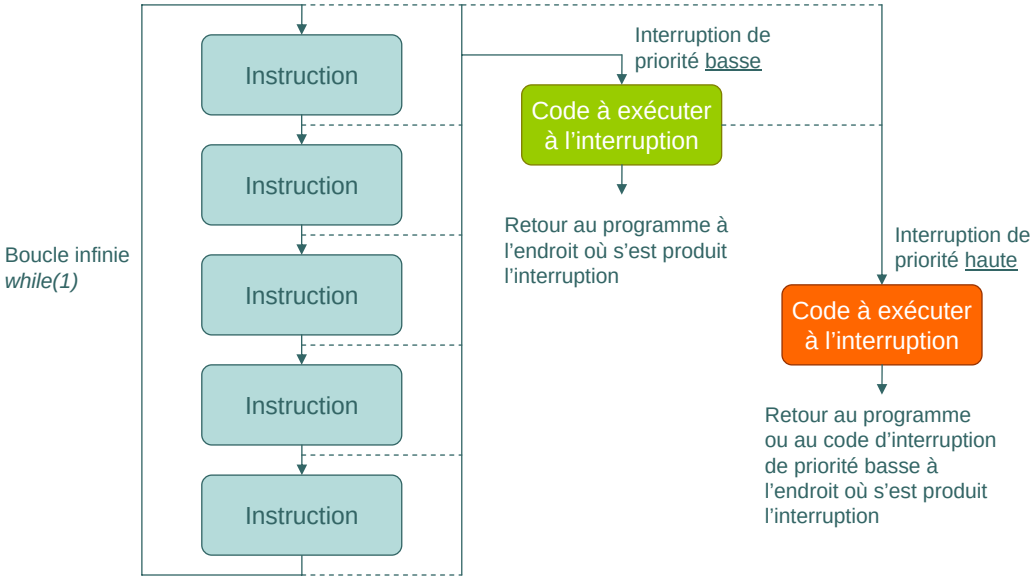


FIG. 48 – Les priorités dans les interruptions

5.4.3 Faire clignoter une LED sur interruption

Nous considérons à nouveau le montage de la figure 44. Le programme suivant permet de faire clignoter la LED D1, cette fois-ci en utilisant les interruptions, ce qui permettra d'exécuter d'autres instructions dans la boucle principale au lieu d'attendre à chaque fois 0,25s sans rien faire.

Listing 2 – Programme faisant clignoter une LED sur interruption

```
1  #include <p18f452.h>  // déclarations pour le PIC18F452
2  #include <timers.h>   // fonctions pour les timers
3
4  // Configuration
5  #pragma config OSC = HSPLL  // oscillateur HS, avec PLL (fréquence interne multipliée par
       4)
6  #pragma config PWRT = ON    // power up timer (activé)
7  #pragma config WDT = OFF    // watchdog timer (désactivé)
8  #pragma config BOR = ON     // brown out reset (activé)
9  #pragma config BORV = 42    // brown out voltage (4.2V)
10 #pragma config LVP = ON     // low voltage ICSP (activé)
11
12 // Prototype de la fonction interruption
13 void highISR(void);
14
15 // Lors d'une interruption (de priorité haute ici)
16 #pragma code highVector=0x008
17 void interruptAtHighVector(void) {
18   // On doit exécuter le code de la fonction highISR()
19   _asm GOTO highISR _endasm
20 }
21 #pragma code // Retour à la zone de code
22
23 volatile short ledCount = 0;  // temporisation pour allumage de la LED toutes les 0,25s
24 volatile char ledStatu = 0;   // stocke l'état de la LED
25
26 #pragma interrupt highISR
27 void highISR(void) {
28   unsigned char sProdL;
29   unsigned char sProdH;
30
31   // Sauvegarde du contenu des registres de calcul
32   sProdL = PRODL;
33   sProdH = PRODH;
34
35   // C'est le Timer1 qui a levé l'interruption
36   if (PIR1bits.TMR1IF) {
```

```
37      WriteTimer1(64536); // 2^16 - 1000 : la prochaine interruption a lieu dans 1000
            cycles = 100us
38
39      ledCount++;
40
41      if (ledCount >= 2500) {
42        ledCount = 0;
43        ledStatu = !ledStatu;
44        PORTAbits.RA0 = ledStatu;
45      }
46
47      // On réautorise l'interruption
48      PIR1bits.TMR1IF = 0;
49    }
50
51    // Restauration des registres de calcul
52    PRODL = sProdL;
53    PRODH = sProdH;
54  }
55
56  void main(void) {
57    // On configure le pin 0 du port A en sortie
58    TRISAbits.TRISA0 = 0;
59
60    // On autorise toutes les interruptions
61    INTCONbits.GIE = 1;
62    INTCONbits.PEIE = 1;
63
64    // Configuration du timer1
65    PIE1bits.TMR1IE = 1;  // autorise les interruption par dépassement du Timer (Timer
          overflow)
66    OpenTimer1(TIMER_INT_ON    // active le timer1
67       & T1_16BIT_RW           // compte sur 16 bits
68       & T1_SOURCE_INT         // utilise l'horloge interne
69       & T1_PS_1_1             // incrémente le compteur à chaque cycle (1:1)
70       & T1_OSC1EN_OFF         // pas d'oscillateur sur le timer1
71       & T1_SYNC_EXT_OFF       // ne pas se synchroniser sur une horloge externe
72    );
73    WriteTimer1(64536); // 2^16 - 1000 : la prochaine interruption a lieu dans 1000 cycles
          = 100us
74
75    while(1) {
76      // Boucle principale, désormais vide !
77    }
78  }
```

Deuxième partie
Conception et réalisation de cartes

6 Utilisation d'OrCAD Capture

6.1 Créer un nouveau projet

Pour créer un nouveau projet sous Capture, il suffit d'aller dans le menu « File » → « New » → « Project... ». Donnez alors un nom et un emplacement au projet et cochez l'option « Analog or Mixed A/D ». Sélectionnez ensuite « Create a blanc project ». Vous vous retrouvez alors avec une page vide, qui correspond à la première (et unique pour le moment) page de votre nouveau projet.

6.2 Prise en main

6.2.1 Les nets

Tout d'abord, il faut comprendre le principe de fonctionnement des connections dans Capture : chaque ensemble de broches reliées entre elles correspond à une net. Capture attribut à chaque net un nom par défaut, composé d'un « N » suivit par un nombre unique. Les connexions sont symbolisées par des fils ou des bus, mais peuvent également ne pas être représentés. Ainsi, si vous donnez le même nom à deux fils, à priori non connectés entre eux, ils appartiendront en réalité à la même net et seront pas conséquent reliés implicitement.

Fig. 49 – Les nets dans Capture

Sur la figure 49, on a attribué un nom à chaque fil, qui vient remplacer le nom de net attribué par défaut par Capture. Une connexion entre deux fils est symbolisée par un point, comme pour la net N1. Les deux fils de droite ont le même nom de net, ils sont donc reliés entre eux, même si aucune connexion n'apparait sur le schéma.

Il est prudent de bien faire la distinction entre majuscules et minuscules pour les noms de net, et de ne pas nommer deux nets de la même manière en ne changeant que la case dans le nom.

Les noms et numéros des broches des composants n'ont pas de rapport avec le nom des nets donnés aux fils connectés dessus, excepté pour les broches d'alimentation (qui sont du type « power »), dont le nom de net par défaut correspond au nom de la broche, tant que l'on a pas connecté de fil dessus, qui vienne changer le nom de la net.

6.2.2 Les alimentations

Dans la mesure où nous ne désirons pas simuler notre projet, il ne faut placer aucun générateur sur le schéma du circuit, en vue de l'exporter ensuite sous Layout afin de créer le Printed Circuit Boards (PCB) de la carte. Il faut par contre évidemment prévoir les connecteurs d'alimentation.

En outre, il existe sous Capture plusieurs symboles pour l'alimentation et la masse. Le 0 V est généralement désigné par GND, tandis que le 5 V est désigné par VCC, en ce qui concerne les noms de nets. Cette convention doit absolument être respecté sous Capture, car de nombreux composants schématisés dans les librairies du programme alimenté en 5 V utilisent ces noms pour les broches d'alimentation et de masse, qui n'apparaissent pas toujours sur le schéma, mais qui appartiennent tout de même aux nets VCC et GND respectivement.

Si une autre tension d'alimentation doit intervenir, il faut donc la nommer autrement, par exemple « VDD » (qui est généralement le nom de la broche d'alimentation des composants lorsque ceux-ci ne sont pas alimentés en 5 V) ou « 12V » et utiliser de préférence un autre symbole d'alimentation.

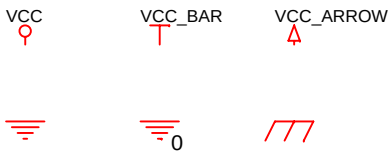


FIG. 50 – Quelques symboles d'alimentation dans Capture

Le nom de net correspond au nom du symbole pour les différents VCC et GND. Pour les symboles de masse, leur nom n'est simplement pas affiché (sauf pour le symbole nommé « 0 », le nom de la net étant alors « 0 »). Le symbole de masse standard est celui représenté en bas à gauche de la figure 50, dont le nom est « GND ».

6.2.3 Les bus

Les bus représentent des paquets de fils et permettent de simplifier les schémas électroniques et faciliter leur lecture. On peut voir un bus, représenté par un trait plus épais dans Capture, comme une gaine dans laquelle place plusieurs fils. Notez que l'on ne peut pas directement relier un fil au bus, pour cela, il faut placer une « bus entry », représenté par un bout de fil tourné de 45°. La figure 51 illustre un bus schématisé sous Capture.

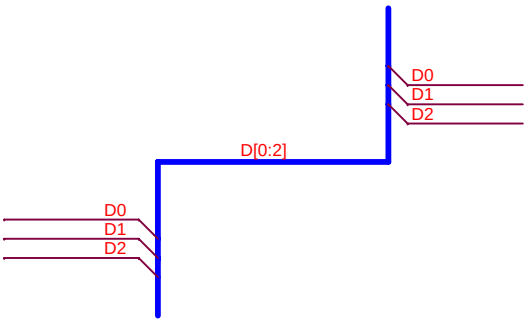


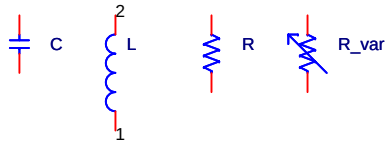
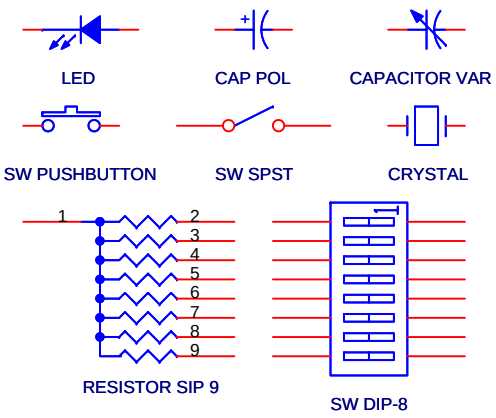
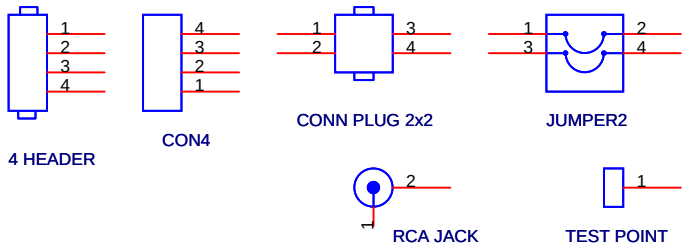
FIG. 51 – Représentation d'un bus sous Capture

Vous devez maintenant vous poser une question : comment savoir qu'à un certain fil entrant sur le bus correspond tel autre fil relié au bus ? Il faut pour cela utiliser des « net alias », et donner le même nom de net aux deux fils, comme cela a été fait figure 51. Connecter les fils à un bus n'est d'ailleurs qu'une mise en forme, puisque les deux fils sont reliés à partir du moment où ils ont le même nom de net.

En principe, il faut nommer les bus dans Capture, avec un nom de la forme « busname[m:n] », où m et n sont des entiers qui indiquent respectivement le premier et le dernier indice des nets connectés au bus (et le nom du bus ne doit pas se terminer par un chiffre). Par exemple, le bus « bus[1:3] » contiendra les nets « bus1 », « bus2 » et « bus3 ».

Il n'est donc pas possible, si l'on respecte scrupuleusement cette règle, de connecter des nets aux noms différents, hormis l'indice. En pratique, il n'est pas nécessaire de nommer les bus et l'on a pas à se soucier des noms des nets qui y sont connectés, si l'on se contente d'exporter la netlist de notre schéma sous Layout. Cela peut toutefois poser problème pour d'autres exports (un bus non nommé provoque d'ailleurs une erreur Design Rules Check (DRC) dans Capture) !

6.2.4 Les principales librairies

Librairie (.olb)	Composants
ANALOG	
DISCRETE	 Diodes, transformateurs, relais...
CONNECTOR	 Tous types de connecteurs
ARITHMETIC BUSDRIVERTRANSCEIVER SHIFTREGISTER GATE LATCH MUXDECODER COUNTER	CMOS et TTL séries 54, 74, 4000...
TRANSISTOR	Transistors...

TAB. 17 – Principales librairies sous OrCAD

6.2.5 Les raccourcis clavier indispensables

Touche	Action
P	Placer un composant
W	Placer un fil
B	Placer un bus
E	Placer une entrée de bus
N	Placer un net alias
F ou G	Placer l'alimentation (VCC et GND)
X	Indiquer qu'une broche n'est pas connecté
I	Zoom avant (centré sur la position du curseur)
O	Zoom arrière (centré sur la position du curseur)

TAB. 18 – Raccourcis clavier sous Capture

Certes, un (tout) petit effort de mémoire est nécessaire pour apprendre ces quelques raccourcis clavier, mais les connaitre peut faire gagner énormément de temps !

6.3 Premier schéma (ultra simple)

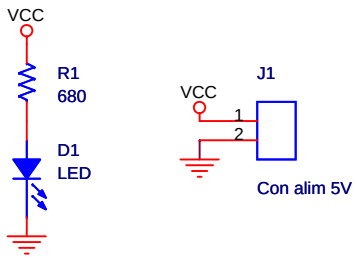


FIG. 52 – Schéma ultra-simple sous Capture

La figure 52 représente un premier schéma très simple sous Capture, permettant d'allumer une LED à partir d'une source d'alimentation 5 V. Vous remarquerez donc la présence d'un connecteur d'alimentation à deux broches.

Concernant la mise en forme du circuit, plutôt que de vouloir relier tous les VCC et les GND par des fils, il est plus propre de relier chaque broche d'alimentation directement au symbole correspondant. Cela évite d'avoir de longs fils inutiles qui viennent compliquer la lecture du schéma.

La LED se trouve dans la librairie `DISCRETE.OLB`, et le connecteur dans la librairie `CONNECTOR.OLB`.

7 Passer d'OrCAD Capture à OrCAD Layout

Une fois votre schéma sous Capture terminé, il faut générer une netlist pour Layout. Mais avant cela, il faut associer à chaque composant de votre schéma une empreinte (footprint en anglais). Ce n'est pas indispensable pour la création de la netlist, mais dans ce cas il vous faudra associer les footprints aux composants au moment de l'importation dans Layout, ce qui n'est pas forcément très pratique et vous fera sans doute perdre plus de temps qu'autre chose, quand vous vous rendrez compte qu'il vous faudra d'abord créer certains footprints...

7.1 Associer un footprint à un composant

Comme nous l'avons dit, cette étape se passe de préférence dans Capture, avec la génération de la netlist.

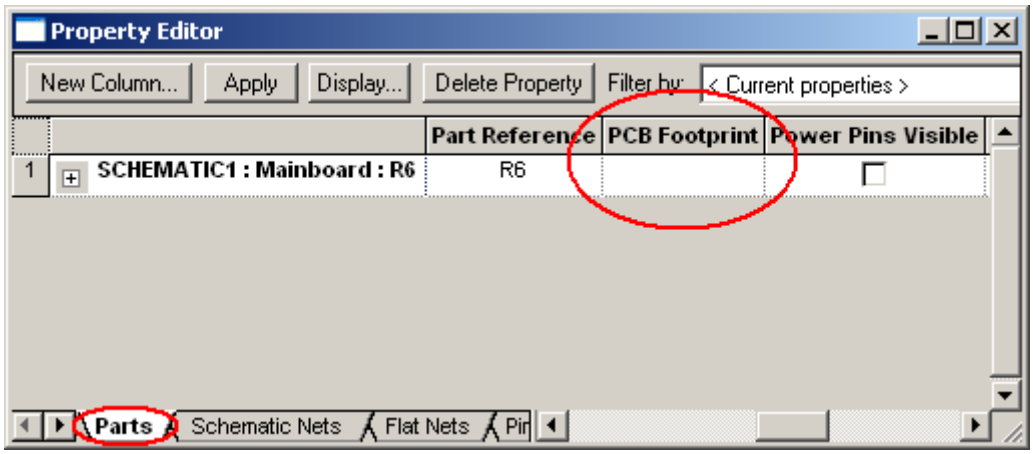


FIG. 53 – Associer un footprint à un composant dans Capture

Pour cela, double-cliquez sur le composant et complétez la colonne « PCB Footprint » de l'onglet « Parts ». Mettez simplement le nom de l'empreinte sans se préoccuper de la librairie dans laquelle elle se trouve.

Pour modifier plusieurs footprints en une fois, sélectionnez simplement plusieurs composants, cliquez sur la sélection avec le bouton droit et sélectionnez « Edit Properties... » (voir figure 54). Pour sélectionner tous les composants présents sur votre schéma, utilisez Ctrl A .

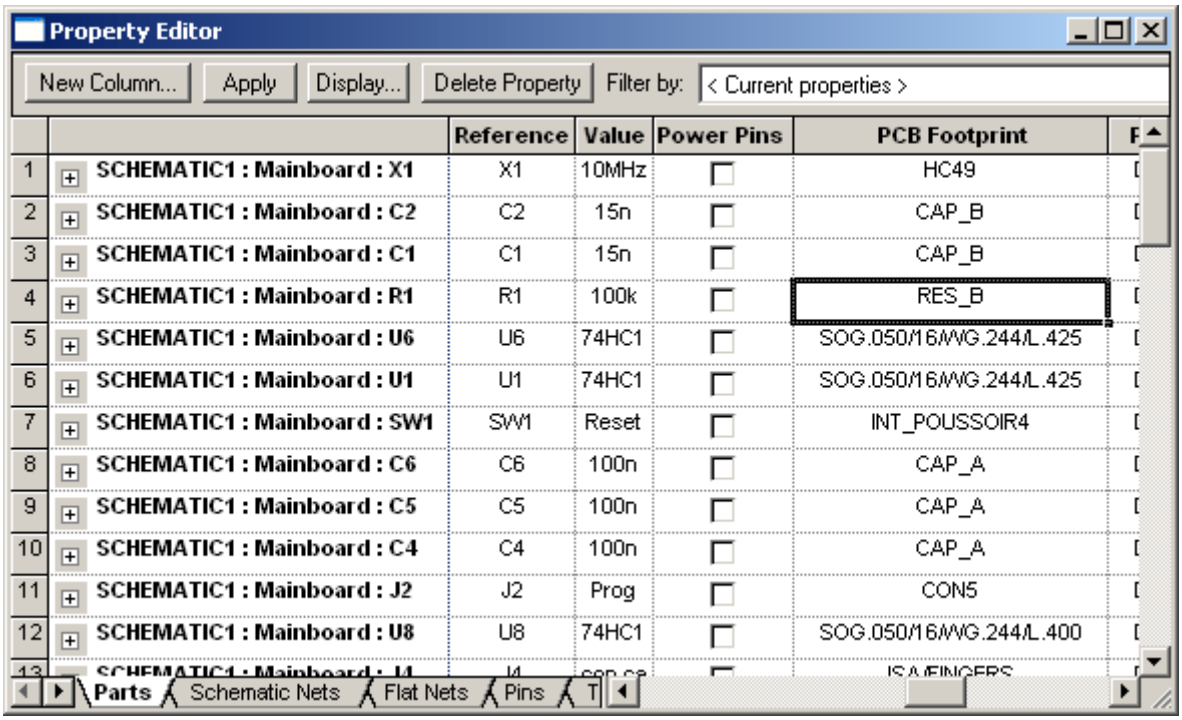


FIG. 54 – Associer un footprint à plusieurs composants en une fois dans Capture

7.2 Générer la netlist pour Layout

Une fois que vous avez associé à chaque composant un footprint, il ne vous reste plus qu'à générer la netlist. Pour cela, allez dans l'arborescence de votre projet et sélectionnez le fichier « .DSN », puis allez dans le menu « Tools » → « Create Netlist... » ou cliquez sur l'icône correspondante dans la barre de menu (figure 55).

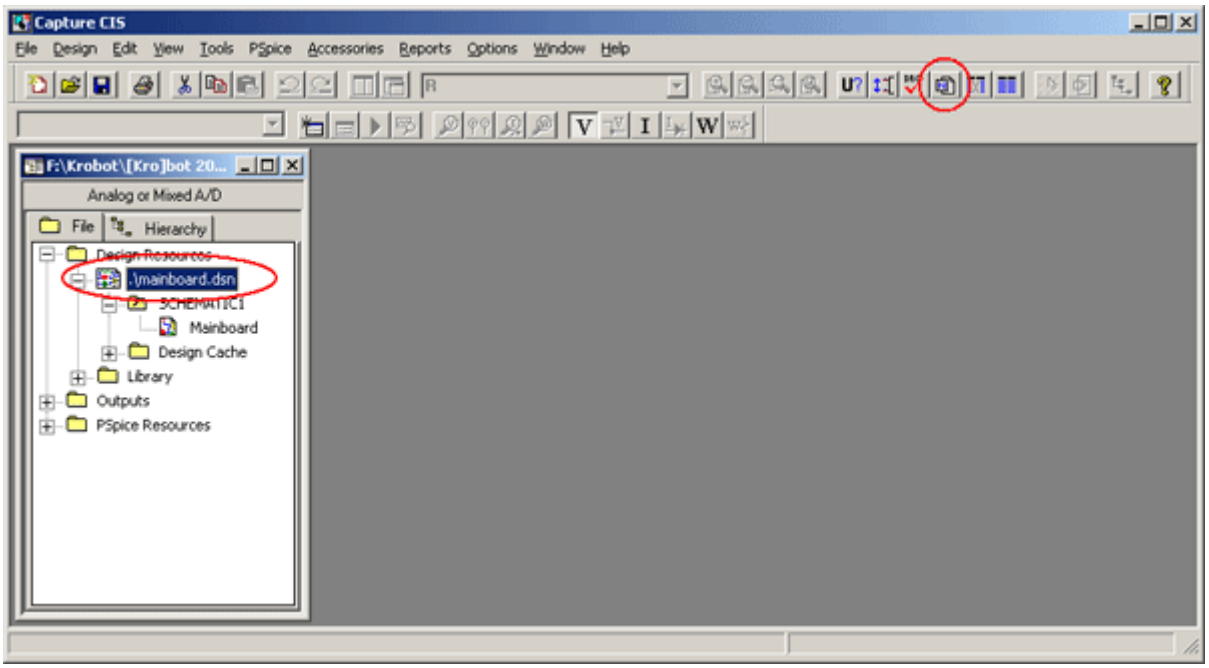


FIG. 55 – Créer la netlist sous Capture

Dans la fenêtre qui apparait alors, sélectionnez en haut l'onglet « Layout », cochez la case « Run ECO to Layout » et sélectionnez l'option « User Proporties are in inches », comme illustré figure 56. Choisissez enfin, en bas de la fenêtre, le nom du fichier contenant la netlist, avec l'extension « .MNL » (vous pouvez laisser la valeur par défaut, votre netlist sera alors enregistrée dans le répertoire du projet OrCAD, avec le nom du projet).

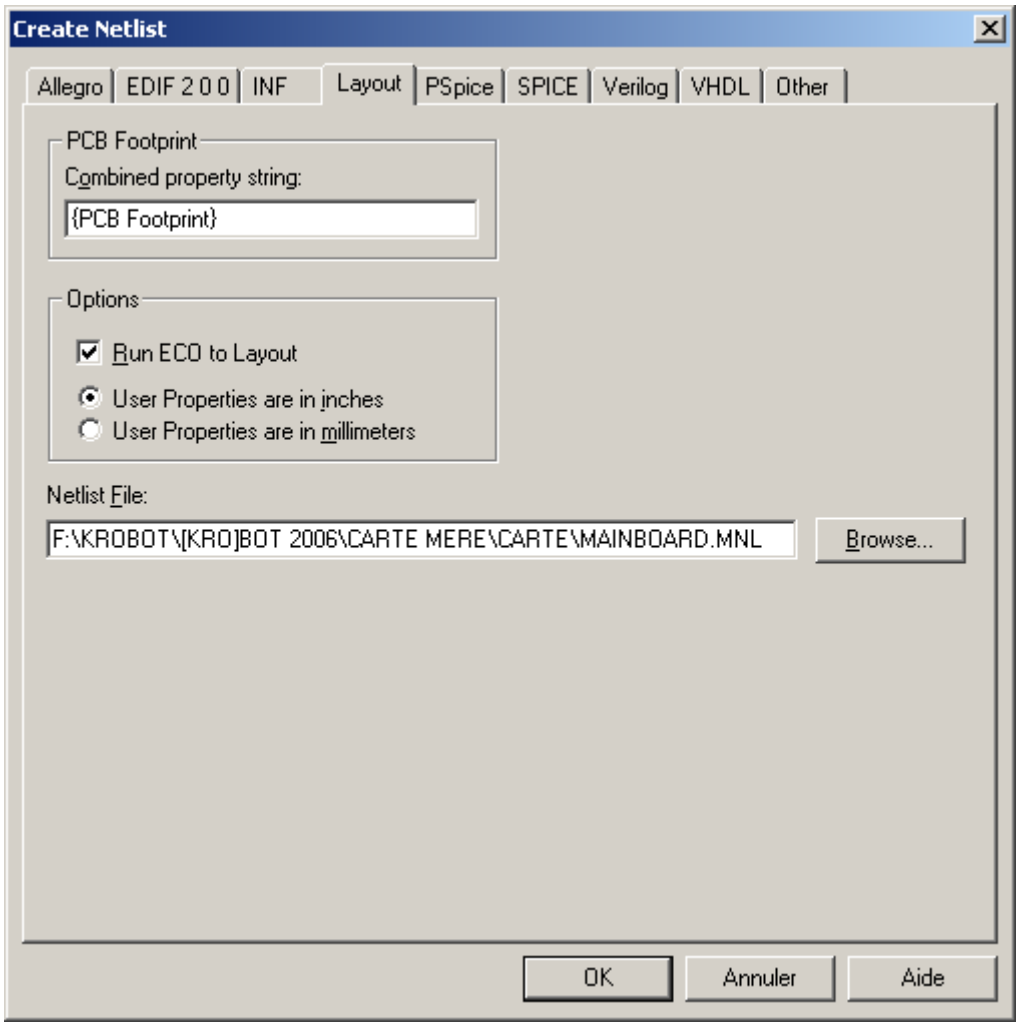


FIG. 56 – Créer la netlist sous Capture, configuration

S'il n'y a pas d'erreur, la netlist est alors créée.

7.3 Créer un nouveau PCB sous Layout

Pour cela, ouvrez Layout et allez dans « File » → « New ». La fenêtre représentée figure 57 apparaît alors.

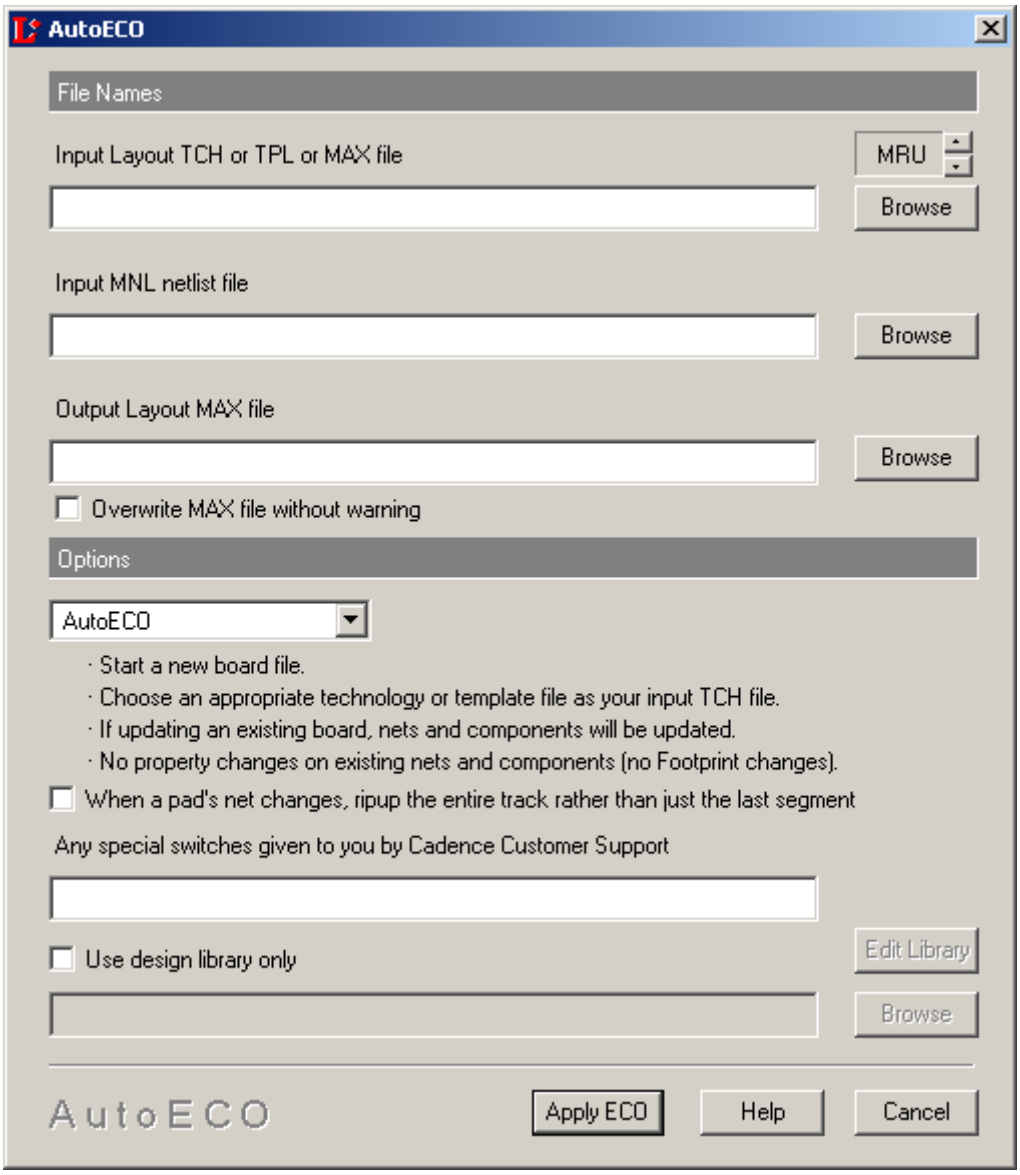


FIG. 57 – Fenêtre nouveau PCB sous Layout

Pour créer un nouveau PCB, Layout a besoin de 3 éléments :
- Un PCB de base (par exemple un « template », extension « .TPL »), qui définit entre autre les couches composant la carte, l'épaisseur des pistes, leur espacement, éventuellement le contour de la carte et d'autres paramètres ;
- La netlist de votre schéma, générée dans Capture (extension « .MNL ») ;
- Le nom du fichier où sera enregistré le PCB (extension « .MAX »).

Une fois les champs correspondant renseignés, figure 57, cliquez sur « Apply ECO ». Si vous avez bien associé à chaque composant un footprint dans Capture et que tous les footprints sont présents dans les librairies incluses dans Layout, l'opération devrait se dérouler sans encombre et les composants devraient alors apparaitre sur la carte, comme sur la figure 58.

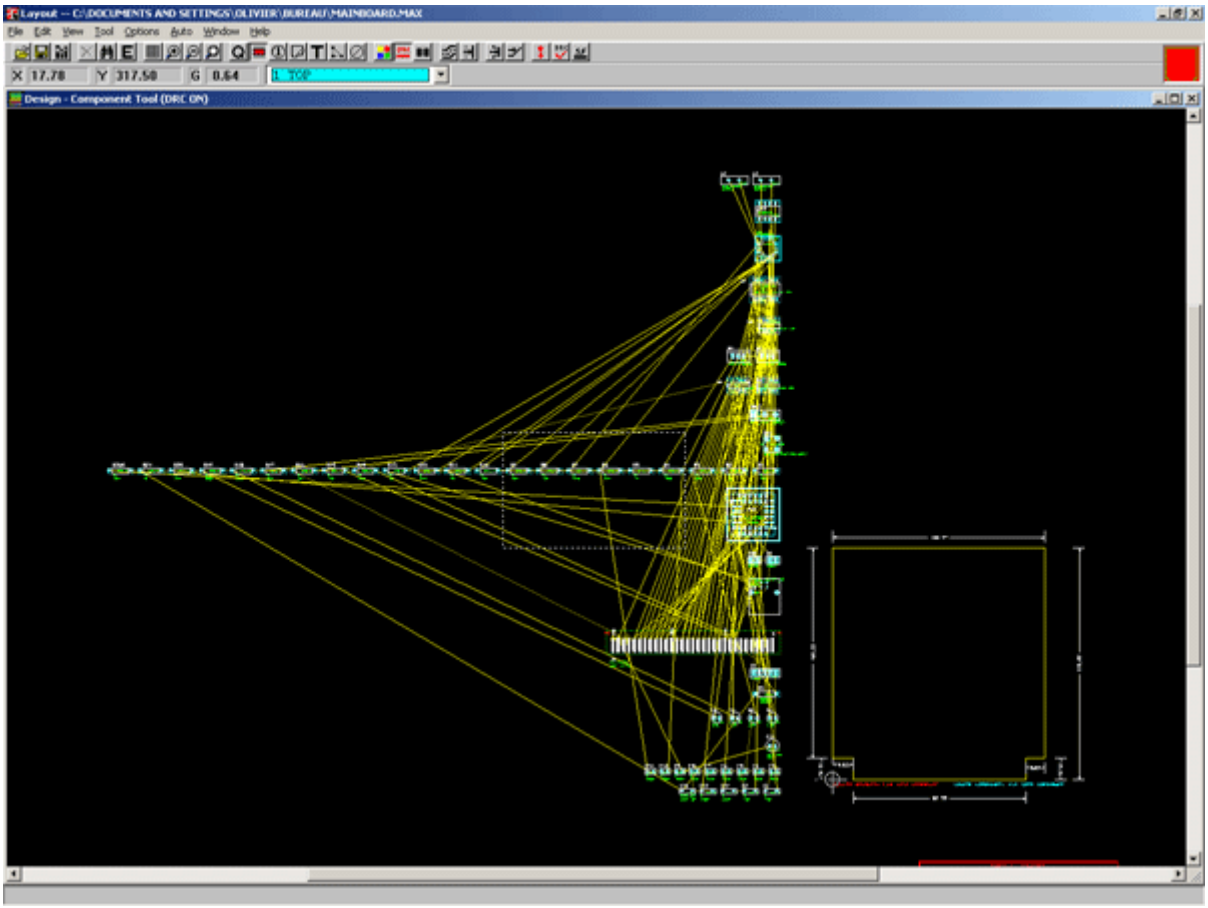


FIG. 58 – Un nouveau PCB dans Layout

8 Conception d'un PCB

Avant même de placer les composants dans Layout, nous allons énoncer ici quelques règles importantes à respecter lors de la conception d'un PCB, ne serait-ce que pour que la carte soit réalisable en pratique. Ne supposons ici que les footprints sont correctes et adaptés pour la réalisation de la carte.

8.1 Les classes de fabrication

L'industrie dispose de normes de fabrication des circuits imprimés, qui répartissent les types de circuits imprimés en 5 classes en fonction de la finesse de gravure, dont les principales spécifications sont données tableau 19.

	Largeur min. des conducteurs	Espacement min. entre conducteurs et/ou pastilles	Différence min. entre Ø pastille et Ø trou (non métallisé)
Classe 1	0,8	0,7	1,6
Classe 2	0,5	0,5	1,2
Classe 3	0,4	0,35	0,9
Classe 4	0,25	0,23	0,9
Classe 5	0,15	0,2	0,9

TAB. 19 – Classes normalisées des circuits imprimés (valeurs données en mm)

Une réalisation « artisanale », au perchlorure de fer, permet de travailler à la limite en classe 3 si nécessaire, mais il est préférable de travailler en classe 1 ou 2 lorsque c'est possible.

8.2 Les couches (layers)

Avant de router une carte, il faut définir le nombre de couches de pistes la composant. Sans outil industriel adapté, il n'est pas possible (ou très difficile) de faire plus de 2 couches « à la main », qui sont les deux faces du circuit imprimé. Ces deux couches sont nommées « TOP » et « BOTTOM » dans Layout. Nous ne ferons donc plus la distinction dans la suite entre la couche « TOP » et face du dessus et la couche « BOTTOM » et face du dessous, sachant que les composants sont placés de manière naturelle sur la face du dessus.

Dans la pratique, on essayera de se contenter d'une seule face, donc la face du dessous, mais dès que les circuits deviennent un peu complexes, la deuxième face devient vite nécessaire, ne serait-ce que pour quelques pistes.

8.2.1 Plan de masse

Un plan de masse est une couche entièrement reliée à la masse, le cuivre ayant la plus grande surface possible. C'est donc la face du dessus qui peut faire office de plan de masse. L'intérêt d'un plan de masse est de réduire la longueur de la boucle d'alimentation, à l'origine des parasites, compensés en partie par des condensateurs de découplage. Un plan de masse n'a donc un véritable intérêt qu'en hautes fréquences. Elle permet toutefois de limiter le risque de court-circuit franc sur l'alimentation, le VCC et GND n'étant pas sur la même face. De plus, si le courant dans le circuit devient important (en cas de court-circuit), la masse est protégée.

Le principal inconvénient d'un plan de masse est qu'il est plus difficile d'effectuer des modifications sur la carte par la suite, s'il faut couper ou rajouter une piste par exemple, ce qui n'est pas négligeable !

8.3 Les pistes (nets)

Concernant les pistes, il y a 2 critères essentiels : la largeur des pistes et l'espacement entre celles-ci.

8.3.1 Largeur des pistes

La largeur d'une piste est déterminée à la fois par :
- Les contraintes au niveau de la réalisation de la carte (gravure) ;
- Le courant qu'elle doit supporter.

Avec la méthode de gravure à notre disposition (perchlorure de fer), la largeur d'une piste ne devrait pas être inférieure à 0,5 mm, afin de garantir qu'il n'y ait pas de problème lors de la gravure. Il est possible de passer à une largeur de 0,4 mm au niveau des broches de composants très petits type CMS afin de respecter un espacement minimum entre les pistes.

Évidemment, lorsque la place le permet, même sans considérer l'intensité circulant dans la piste, une largeur plus importante est toujours la bienvenue. Ainsi, si cela est possible, les pistes peuvent tout à fait faire 1 mm de large, voir plus.

Largeur de piste en fonction du courant Le tableau 20 donne l'intensité admissible pour une épaisseur de 35 μm de cuivre (pour une différence de température de 20 °C) [11].

Largeur du conducteur (en mm)	0,4	0,72	1,14	1,8	2,5	3,5	4,5	5,0	7,1
Intensité admissible (en A)	1,3	2,7	3,8	5,2	6,8	8,3	9,7	11,2	13,0

TAB. 20 – Intensité admissible en fonction de la largeur de piste

Si des contraintes de place sont importantes, il est possible de réduire la largeur d'une piste par rapport à la largeur minimale à un courant donné et de passer ensuite une couche d'étain sur celle-ci lors de la réalisation de la carte, voir de souder un fil pardessus.

8.3.2 Espacement entre pistes et/ou pastilles

L'espacement, ou l'isolation, entre les pistes dépend elle de :
- Les contraintes au niveau de la réalisation de la carte (gravure) ;
- La différence de potentiel entre les pistes.

En pratique, comme l'on travaille à des tensions très faibles, ce sont surtout les contraintes lors de la gravure qui vont conditionner l'espacement minimum entre les pistes.

Pour des tensions continues de 24 V, l'espacement doit être d'au moins 0,4 mm. Afin de pouvoir tracer des pistes entre deux pattes d'un circuit intégré (tensions de 0 à 5 V), on pourra prendre un espacement un peu plus faible, jusqu'à 0,35 mm (ce qui correspond à l'espacement minimum pour un circuit imprimé de classe 3).

Pour des tensions alternatives du secteur (230 V), un espacement de 3,2 mm minimum s'impose.

8.3.3 Configuration de l'espacement dans Layout

Pour configurer les espacements, entre les pistes, pastilles et vias dans Layout, allez dans le menu « Options » → « Global Spacing... », pour ouvrir la fenêtre figure 59.

Route Spacing

Layer Name	Track to Track	Track to Via	Track to Pad	Via to Via	Via to Pad	Pad to Pad
TOP	0.35	0.50	0.35	0.50	0.50	0.35
BOTTOM	0.35	0.50	0.35	0.50	0.50	0.35
GND	0.35	0.50	0.35	0.50	0.50	0.35
POWER	0.35	0.50	0.35	0.50	0.50	0.35
INNER1	0.35	0.50	0.35	0.50	0.50	0.35
INNER2	0.35	0.50	0.35	0.50	0.50	0.35
INNER3	0.35	0.50	0.35	0.50	0.50	0.35
INNER4	0.35	0.50	0.35	0.50	0.50	0.35
INNER5	0.35	0.50	0.35	0.50	0.50	0.35
INNER6	0.35	0.50	0.35	0.50	0.50	0.35
INNER7	0.35	0.50	0.35	0.50	0.50	0.35
INNER8	0.35	0.50	0.35	0.50	0.50	0.35
INNER9	0.35	0.50	0.35	0.50	0.50	0.35
INNER10	0.35	0.50	0.35	0.50	0.50	0.35
INNER11	0.35	0.50	0.35	0.50	0.50	0.35
INNER12	0.35	0.50	0.35	0.50	0.50	0.35
DRILL	0.35	0.50	0.35	0.50	0.50	0.35

FIG. 59 – Fenêtre « Route Spacing » dans Layout, les unités étant affichées en mm

- « Track to Track » : espacement entre les pistes ;
- « Track to Via » : espacement entre les pistes et les vias ;
- « Track to Pad » : espacement entre les pistes et les pastilles ;
- « Via to Via » : espacement entre les vias ;
- « Via to Pad » : espacement entre les vias et les pastilles ;
- « Pad to Pad » : espacement entre les pastilles.

Il est bien de laisser plus d'espacement autour des vias, pour faciliter leur soudure. La configuration de l'espacement représentée sur la figure 59 peut être réutilisée pour vos cartes.

8.4 Les pastilles (padstacks)

Les pastilles doivent être assez grosses pour ne pas sauter au perçage et permettre une soudure correcte et robuste. Comme l'indiquent les classes de fabrication, la différence entre le Ø pastille et le Ø trou doit être au moins de 0,9 mm. Cela n'est toutefois pas toujours possible, lorsque deux broches sont très rapprochées et/ou que l'on veut faire passer une piste entre celles-ci. Dans ce cas, il faut prévoir une forme de pastille différente pour permettre une bonne tenue de celle-ci.

A titre d'exemple, voici les pastilles que nous utilisons couramment :

Nom	Ø trou	Forme	Largeur	Hauteur	Exemple d'utilisation
VIA1	0,8	Round	1,8	1,8	Vias
ROUND_0.8	0,8	Round	2,0	2,0	Dipôles...
SQUARE_0.8	0,8	Square	2,0	2,0	Dipôles... (1re broche)
OBLONG_0.8_V	0,8	Oblong	1,4	2,0	DIP, SIP, PLCC
RECTANGLE_0.8_V	0,8	Rectangle	1,4	2,0	DIP, SIP, PLCC (1re broche)
OBLONG_0.8_H	0,8	Oblong	2,0	1,4	PLCC
RECTANGLE_0.8_H	0,8	Rectangle	2,0	1,4	PLCC (1re broche)
OBLONG_0.9_V	0,9	Oblong	1,8	2,0	Connecteurs
RECTANGLE_0.9_V	0,9	Rectangle	1,8	2,0	Connecteurs (1re broche)
ROUND_1.0	1,0	Round	2,6	2,6	Dipôles...
OBLONG_1.0_V	1,0	Oblong	1,8	2,6	Connecteurs
RECTANGLE_1.0_V	1,0	Rectangle	1,8	2,6	Connecteurs (1re broche)
ROUND_1.5	1,5	Round	3,1	3,1	Dipôles...

TAB. 21 – Exemple de pastilles pour différents Ø de trou (valeurs en mm)

Une forme de pastille différente peut également permettre de repérer la 1re broche d'un composant, quand celui-ci a un sens (carrés et rectangles pour les pastilles du tableau 21).

8.4.1 Modifier les pastilles dans Layout

Pour se faire, allez dans le menu « View » → « Database Spreadsheets... » → « Padstacks », pour ouvrir la fenêtre figure 60.

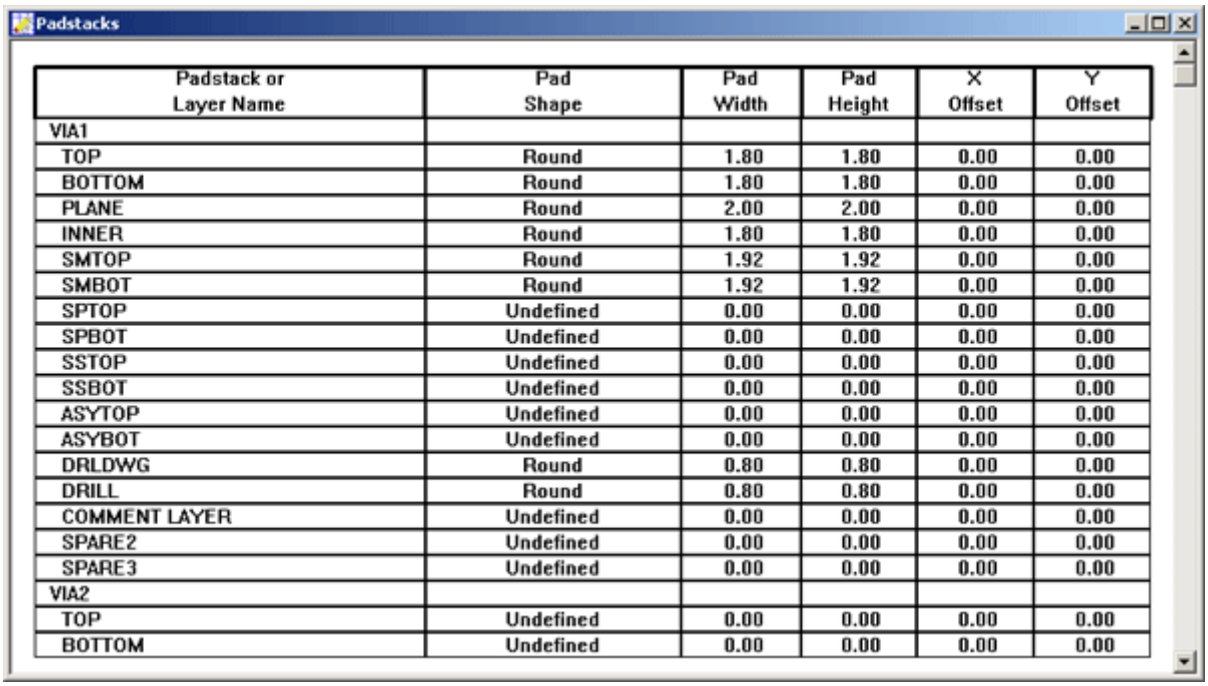


FIG. 60 – Fenêtre « Padstacks » dans Layout, les unités étant affichées en mm

8.5 Routage entre deux broches

Si le routage entre deux broches espacées de 2,54 mm est à éviter autant que possible, il n'y a parfois pas moyen de faire autrement, notamment pour un boitier PLCC par exemple. La figure 61 montre un routage « idéal » pour un DIP, avec les pastilles « OBLONG_0.8_V » (voir tableau 21), où la 3e classe de fabrication est respectée.

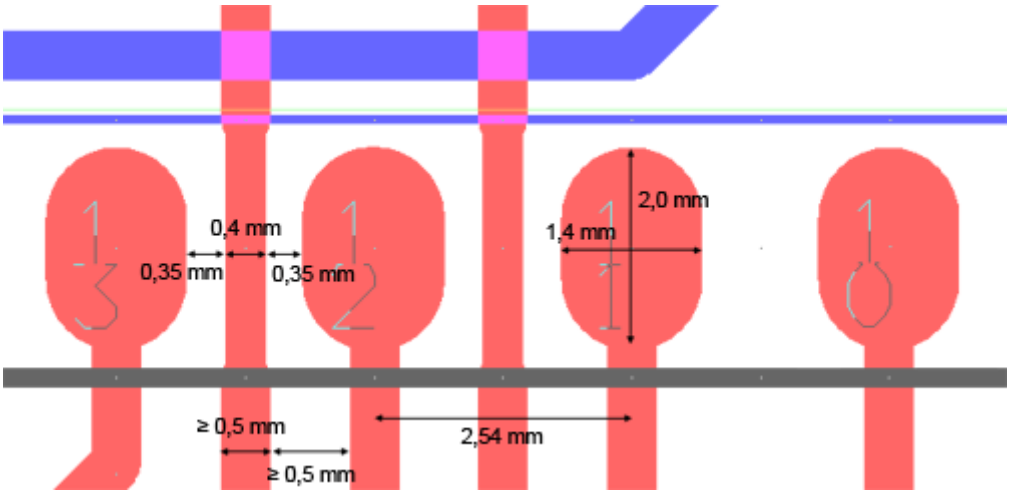


FIG. 61 – Routage entre deux broches d'un DIP

Il ne faut pas faire passer de piste de puissance entre deux broches, car le courant pourrait être trop important par rapport à l'épaisseur de la piste (pour rappel, il ne faut pas dépasser les 1,3 A pour une largeur de piste de 0,4 mm). Le mieux étant de ne router entre deux broches que des nets logiques (0 - 5 V et très faible courant).

8.6 Placement des condensateurs de découplage

Pour qu'un condensateur de découplage soit efficace, il faut, comme nous l'avons vu en théorie, placer le condensateur au plus proche des bornes d'alimentation du circuit intégré. La figure 62 montre comment bien placer un condensateur de découplage.

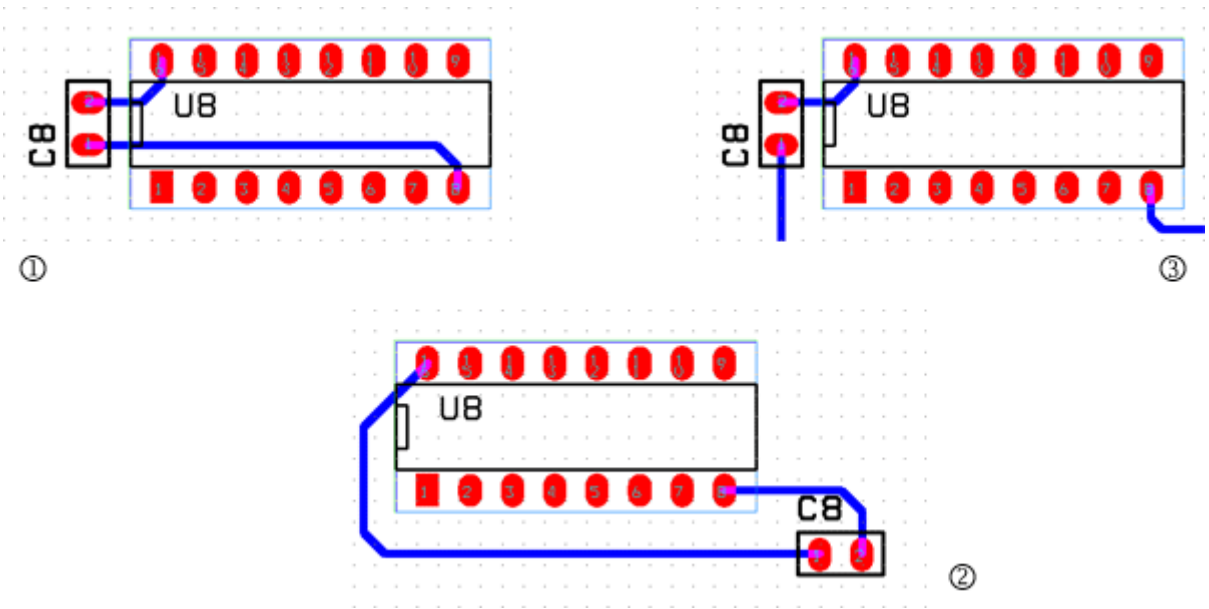


FIG. 62 – Placement des condensateurs de découplage

Les 3 schémas de la figure 62 correspondent à :

1. Un condensateur bien placé et bien routé ;
2. Un condensateur plutôt mal placé et mal routé ;
3. Un condensateur bien placé mais mal routé : il ne remplit pas son rôle de découplage, car il n'est pas directement relié aux deux broches d'alimentation du circuit intégré.

Troisième partie
Schémas classiques

9 Régulateur de tension (5 V)

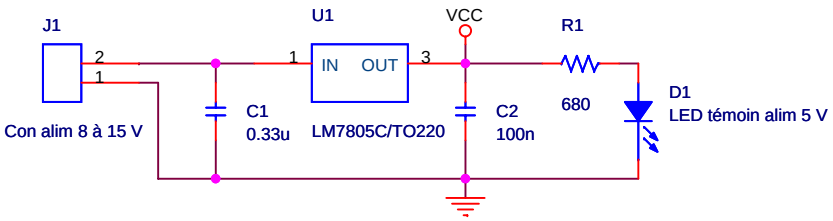


FIG. 63 – Schéma d'un montage régulateur 5 V

9.1 Puissance dissipée

Calcul de la puissance dissipée :

$$P = (Vin - Vout) \times I$$

Ainsi, si le circuit consomme 200 mA, et que la tension d'entrée est de 12 V, la puissance dissipée sera de : $P = (12 - 5) \times 0,2 = 1,4W$

La puissance dissipée dépend donc directement de la chute de tension aux bornes du régulateur, c'est pourquoi une tension d'entrée plus faible induira moins de pertes et ferra moins chauffer le régulateur. A 30°C, la puissance maximale que peut dissiper le régulateur 7805 sans radiateur est d'environ 2 W [6]. Cela correspond à une consommation de 280 mA environ (toujours pour une tension d'alimentation de 12 V). Avec un radiateur adapté, le régulateur peut dissiper jusqu'à 7,5 W à 30°C, ce qui correspond approximativement à 1 A.

On notera donc que pour la plupart des circuits logiques que nous concevrons, qui peuvent être constitués d'un ou deux microprocesseurs, de quelques composants logiques, transistors et LEDs, un 7805 n'a en principe pas besoin de radiateur parce qu'il chauffe trop !

9.2 Dimensionner un radiateur

Afin de dimensionner le radiateur, il nous faut déterminer la résistance thermique totale du composant (c'est-à-dire la résistance entre la puce, à l'intérieur du boitier, et l'air ambiant, notée $R_\theta JA$), via la formule suivante :

$$R_\theta JA = \frac{(Tmax - Tamb)}{P}$$

La température maximale de fonctionnement du régulateur, $Tmax$, vaut 150°C [6].
P correspond à la puissance (maximale) que sera susceptible de dissiper le régulateur (calculée dans le paragraphe précédent).
$Tamb$ est la température ambiante, disons 30°C, du composant en fonctionnement (bien qu'à l'intérieur du boîtier la température soit plus élevée, on peut compter facilement 10°C de plus).

Ainsi, si la puissance à dissiper est par exemple de 5 W, on calcule $R_\theta JA = \frac{(150-30)}{5} = 24$°C/W.

Comment déterminer alors si un radiateur est nécessaire, et dans ce cas, sa résistance thermique ? Pour cela, voyons à quoi correspond exactement $R_\theta JA$, figure 64 :

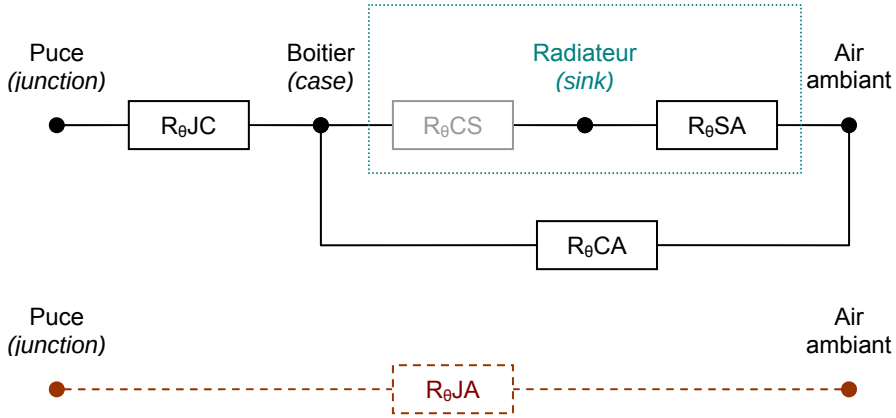


FIG. 64 – Résistance thermique d'un composant

On a alors la relation suivante, sachant que la résistance thermique entre le radiateur et l'air ambiant est $R_\theta SA$:

$$R_\theta JA = R_\theta JC + (R_\theta CS + R_\theta SA) \;//\; R_\theta CA$$

Tout d'abord, on peut négliger la résistance thermique entre le boitier et le radiateur, $R_\theta CS$, qui est en principe très faible si le radiateur est correctement ajusté sur le boitier avec de la pâte thermique (pour fixer les idée, $R_\theta CS = 1/\sigma_\theta CS \approx 0{,}001$°C/W).

FIG. 65 – Tube de pâte thermique ($\sigma_\theta \approx 1$ W/m°C)

Il nous faut encore les résistances thermiques $R_\theta JC$ et $R_\theta CA$ du composant, que l'on trouvera dans sa datasheet [6] (notez que ces données ne sont pas toujours indiquées par certains constructeurs) :
- $R_\theta JC = 5$°C/W ;
- $R_\theta CA = 65$°C/W.

On constate que $R_\theta JC + R_\theta CA > R_\theta JA$, un radiateur est donc bien nécessaire. On a dans ce cas $R_\theta JA = R_\theta JC + R_\theta SA$, car on néglige alors $R_\theta CA$, d'où :

$$R_\theta SA = R_\theta JA - R_\theta JC = 24 - 5 = 19\text{°C/W}$$

Le dissipateur de la figure 66, adapté au boîtier TO220 du régulateur, convient, avec une résistance thermique de 18°C/W (n'oublions pas que plus faible est la résistance, meilleur est la dissipation).

FIG. 66 – Dissipateur pour boîtier TO220, de résistance thermique 18°C/W

10 Commande d'un relais

Il existe plusieurs manières de commander un relais à partir d'une sortie compatible TTL (sortie d'un PIC par exemple). Dans tous les cas, il est indispensable de prévoir une diode de roue libre rapide (type 1N4148) en parallèle avec le relais, afin de court-circuiter les tensions inverses engendrées par l'effet de

self du relais (qui peuvent atteindre 100 V, ce qui est supérieur à la tension inverse que peuvent supporter la plupart des transistors).

Afin de déterminer le montage adapté à la commande du relais, il faut connaitre les niveaux de tension et de courant que peut débiter l'unité de commande. Voici les grandeurs typiques pour une sortie d'un circuit logique :

Logique TTL			
V_{OH}	2,7 V	I_{OH}	-0,4 mA
V_{OL}	0,5 V	I_{OL}	8 mA
Logique CMOS			
V_{OH}	4,5 V	I_{OH}	-20 mA
V_{OL}	0,5 V	I_{OL}	20 mA

TAB. 22 – Niveaux de tension et de courant TTL et CMOS

10.1 Caractéristiques du relais

FIG. 67 – Exemple de relais 12 V

Les caractéristiques intéressantes d'un relais sont sa tension de commande, sa tension maximale de commutation, son courant maximal de commutation ainsi que la résistance de son bobinage, qui permet de déterminer le courant « de commande » :

$$I_{com} = V_{com}/R_{bob}$$

Pour un relais standard alimenté en 12 V, possédant une résistance de bobinage de 600 Ω, le courant de commande sera donc $I_{com} = 12/600 = 20$ mA. C'est le courant qu'il faut pouvoir fournir au relais pour le faire fonctionner.

Le courant de sortie maximal que peuvent débiter la plupart des circuits logique est donc trop faible, ou très limite, pour commander un relais. Il faut donc utiliser un montage à transistor.

10.2 Transistor NPN en commutation

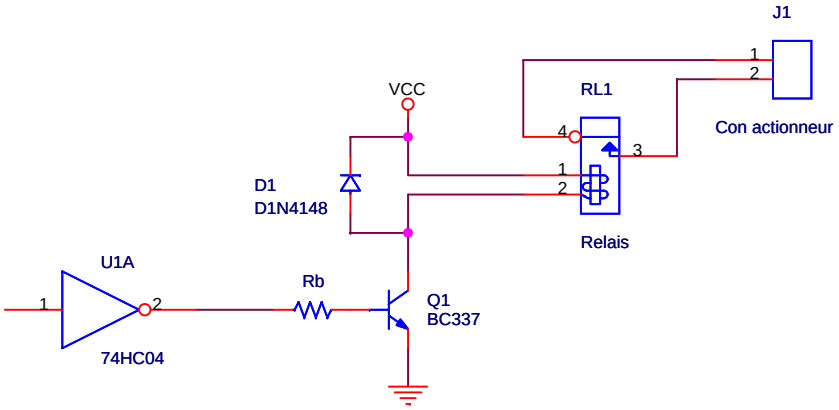


FIG. 68 – Schéma du montage de commande d'un relais avec un transistor NPN

Le transistor Q1 utilisé en commutation réalise l'interface de courant. Afin de dimensionner le montage, il nous faut connaitre les caractéristiques du transistor :

Ic_{max}	800 mA
Gain en courant $h_{FE_{min}}$	100
$V_{BE_{(on)}}$	1,2 V
$V_{CE_{(sat)}}$	0,7 V
Puissance max. P_{max}	625 mW

Tab. 23 – Caractéristiques du transistor BC337 [5]

Lorsque le transistor est saturé, $Ic = \frac{(VCC - V_{CE_{(sat)}})}{R_{bob}} = \frac{(12-0,7)}{600} = 19$ mA ($\approx I_{com}$). La valeur de h_{FE} pouvant grandement varier d'un transistor à l'autre, on prend sa valeur minimale, donc $Ib = Ic/h_{FE_{min}}$ = 0,19 mA. On calcule alors Rb pour avoir $Ib \approx 0{,}38$ mA, afin d'être sûr(e) de la saturation du transistor (on prend un coefficient de sursaturation de 2 minimum, mais on peut monter à $\times 3$ ou $\times 5$). Il faut que $Ib < I_{OH}$, on est donc au cas limite pour une porte TTL.

Pour le calcul de Rb, on se place également dans le pire des cas, où la tension de sortie de la porte est minimale et vaut V_{OH} (V_{OH} = 4,5 V pour une porte CMOS) :

$$Rb = \frac{(V_{OH} - V_{BE_{(on)}})}{Ib} = \frac{(4,5 - 1,2)}{0,0004} = 8,2\ k\Omega$$

La puissance consommée par le transistor vaut environ :

$$P = V_{CE_{(sat)}} \times Ic$$

Lorsque le transistor est bloqué, P $\approx$ 0 (Ic $\approx$ 0) et lorsqu'il est saturé, P = $0,7 \times 0,019 = 13$ mW. La puissance consommée est très faible, la puissance maximale n'est pas un critère important pour le choix du transistor. En pratique, le transistor ne doit donc pas chauffer.

10.3 Transistor PNP en commutation

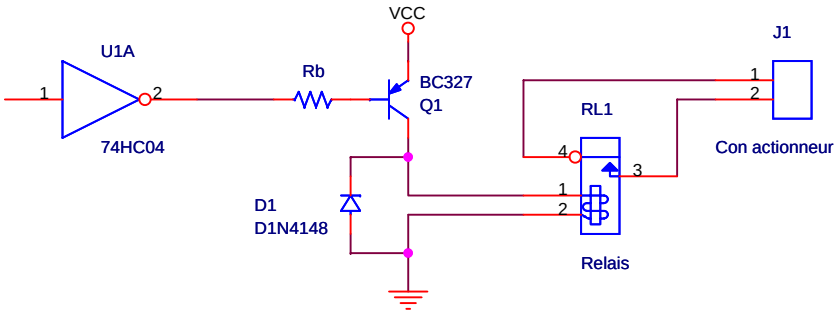


Fig. 69 – Schéma du montage de commande d'un relais avec un transistor PNP

Ce montage est tout à fait analogue au montage précédent, mais cette fois-ci, le courant Ib ne sera plus limité par I_{OH}, mais par I_{OL} : $Ib < I_{OL}$. Ce montage peut donc être avantageux si l'on utilise une porte TTL, mais n'apporte rien pour les portes CMOS.

En outre, la commande du transistor PNP est inversée par rapport au transistor NPN : un 1 logique active le relais pour le NPN tandis que c'est un 0 logique qui l'active pour le PNP (le sens du courant Ib étant inversé).

10.4 Montage Darlington

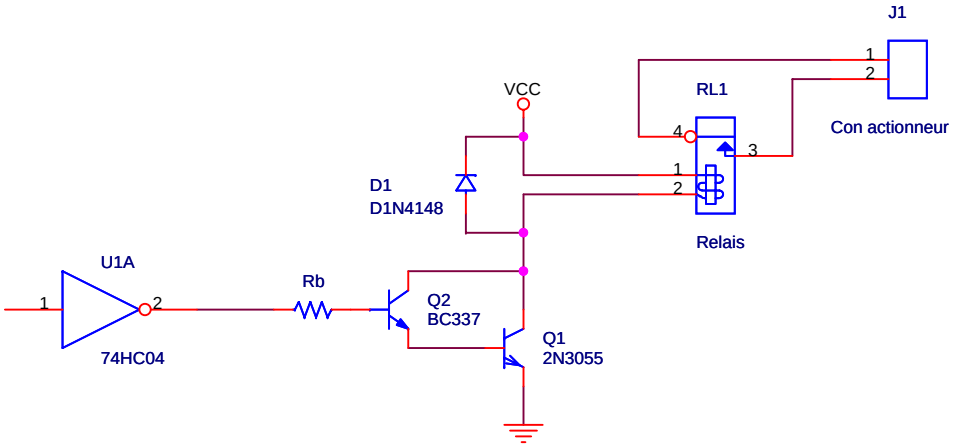


FIG. 70 – Schéma du montage de commande d'un relais avec un Darlington

Ce montage permet d'augmenter le gain en courant. Prenons l'exemple d'un relais 12 V / 10 Ω :

$$Ic = \frac{(VCC - V_{CE_{(sat)}})}{R_{bob}} = \frac{(12 - 1,1)}{10} = 1,13\ A$$

Ic_{max}	15 A
Gain en courant $h_{FE_{typ}}$	50
$V_{BE_{(on)}}$	1,5 V
$V_{CE_{(sat)}}$	1,1 V
Puissance max. P_{max}	115 W

TAB. 24 – Caractéristiques du transistor 2N3055 [1]

Le gain en courant vaut environ $50 \times 100 = 5\,000$, soit un courant $Ib = Ic/5000 = 0{,}23$ mA. Le calcul de Rb nous donne alors :

$$Rb = \frac{(V_{OH} - (V_{BE_{(1)}} + V_{BE_{(2)}}))}{2 \times Ib} = \frac{(4,5 - (1,5 + 1,2))}{0,00046} = 3,9\ k\Omega$$

Comme vous aurez pu le constater, la commande d'un relais ne s'improvise pas, il est nécessaire de connaitre un minimum les caractéristiques du relais ainsi que des transistors que l'on utilise si l'on ne veut pas faire des montages qui marchent au petit bonheur la chance...

11 Commutation de puissance

Si vous avez compris la commande d'un relais, c'est exactement la même chose : on remplace le relais par une charge, Rl et les calculs permettant de déterminer les composants du montage restent les mêmes.

Si la charge est inductive, comme c'est le cas pour le relais, ou pour un moteur par exemple, il est indispensable de prévoir une diode de roue libre, comme sur le schéma 71.

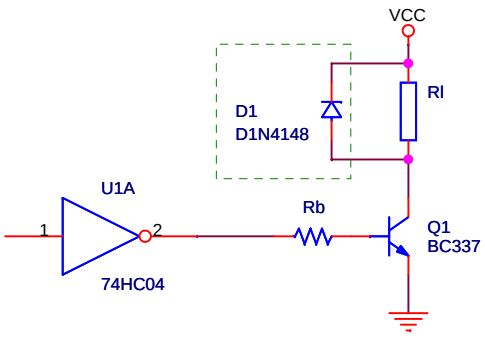


FIG. 71 – Schéma d'un montage de commutation de puissance

12 Conversion série - RS232

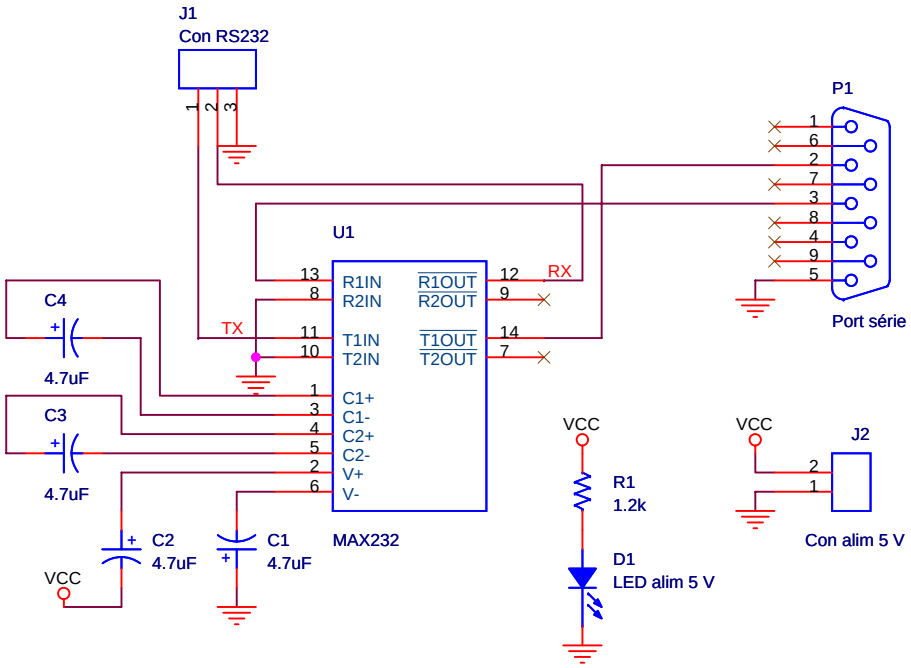


FIG. 72 – Schéma du montage série ↔ RS232

État logique	Transmetteur	Récepteur
0 (espace)	+5 à +15 V	+3 à +25 V
1 (marque)	-5 à -15 V	-3 à -25 V
Indéfini		-3 à +3 V

TAB. 25 – Niveaux de tension de la norme RS232

Le montage de base est représenté figure 72. Il est évidemment possible de l'enrichir d'un régulateur de tension 5 V, en vue de l'alimenter à partir d'une pile 9 V par exemple.

Le MAX232 permet de générer les niveaux de tension requis par le protocole RS232 (voir tableau 25) à partir des condensateurs et de l'alimentation 5 V [3] et vient donc adapter le niveau de tension d'entrée et sortie RS232 d'un circuit TTL ou CMOS (un PIC par exemple), limité de 0 à 5 V, pour une vraie compatibilité RS232 permettant de dialoguer sur le port série d'un PC.

13 Conversion USB - série

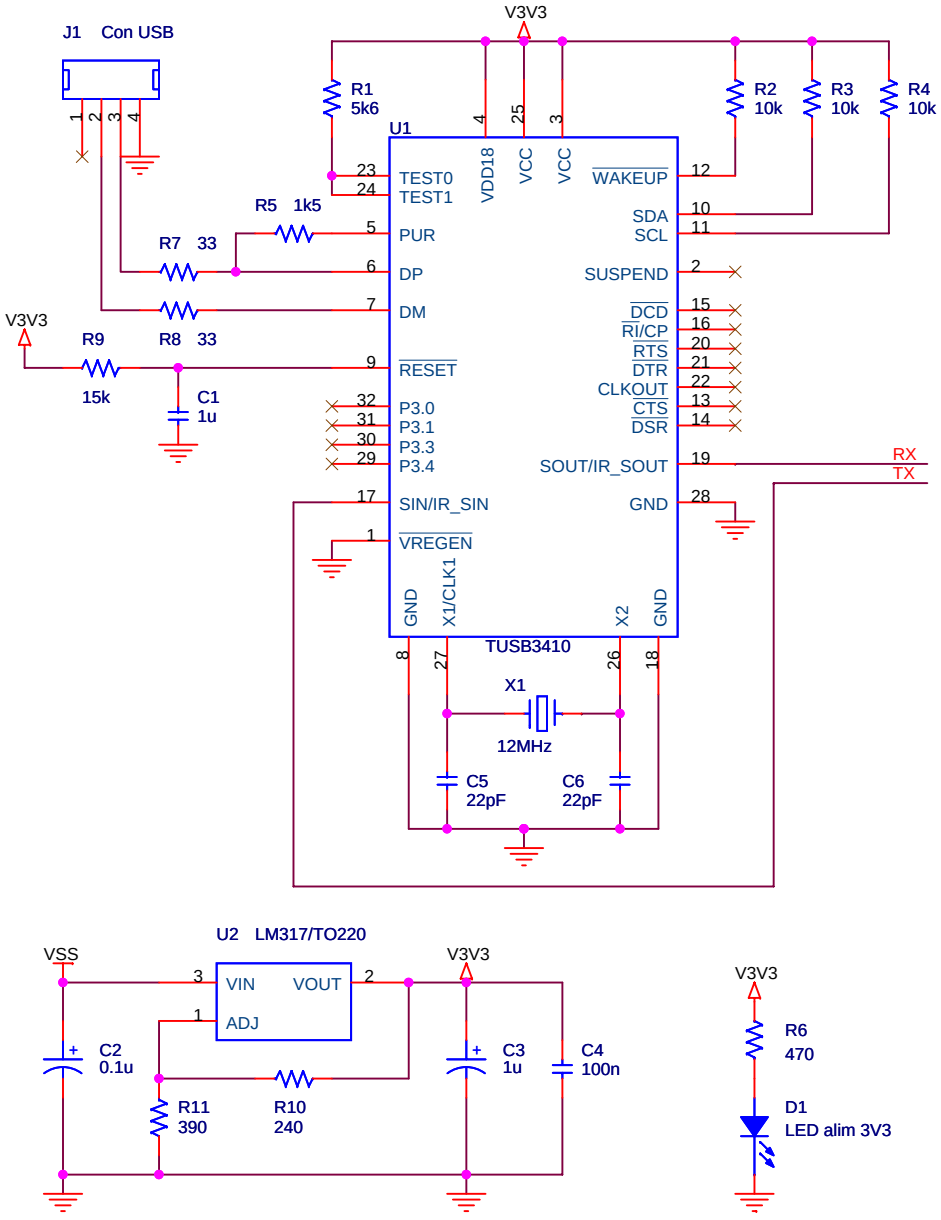


FIG. 73 – Schéma du montage USB ↔ série

Ce montage permet tout simplement de réaliser une conversion USB - série, de manière transparente grâce au composant TUSB3410 [4], qui comme vous l'aurez remarqué, doit être alimenté en 3,3 V.

14 Interface moteur pas à pas bipolaire

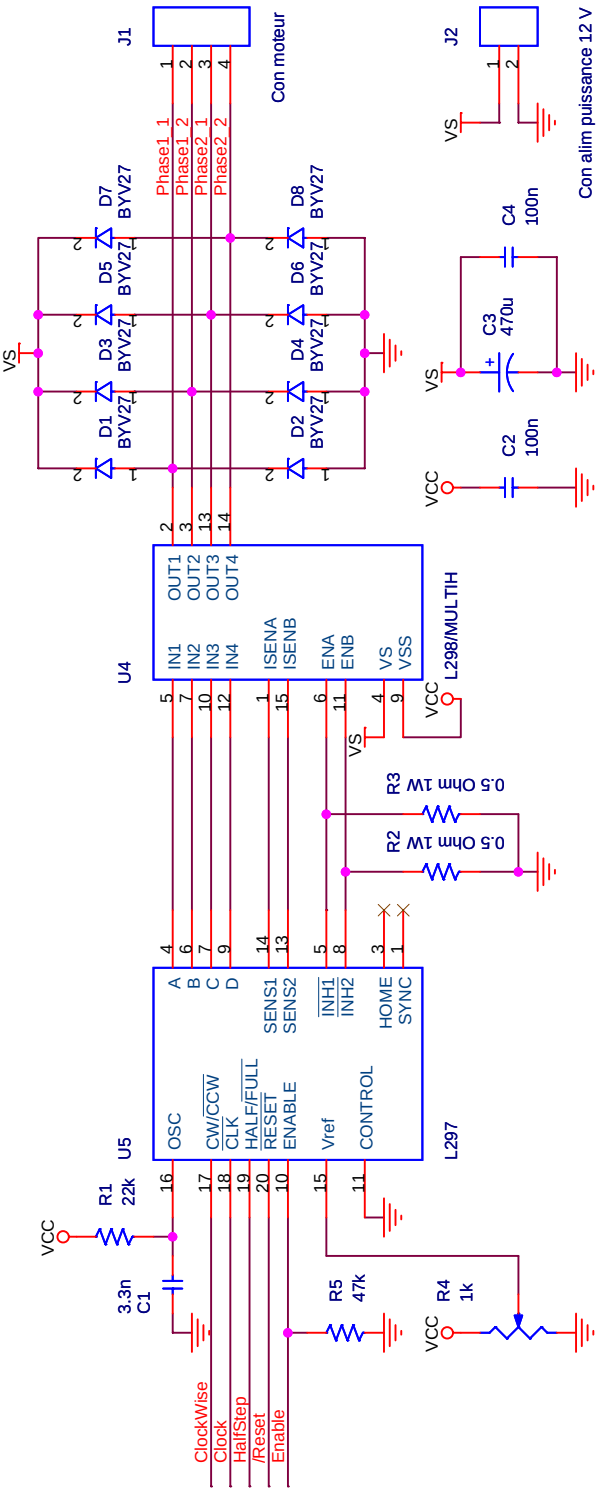


FIG. 74 – Schéma du montage d'interfaçage d'un moteur pas à pas bipolaire

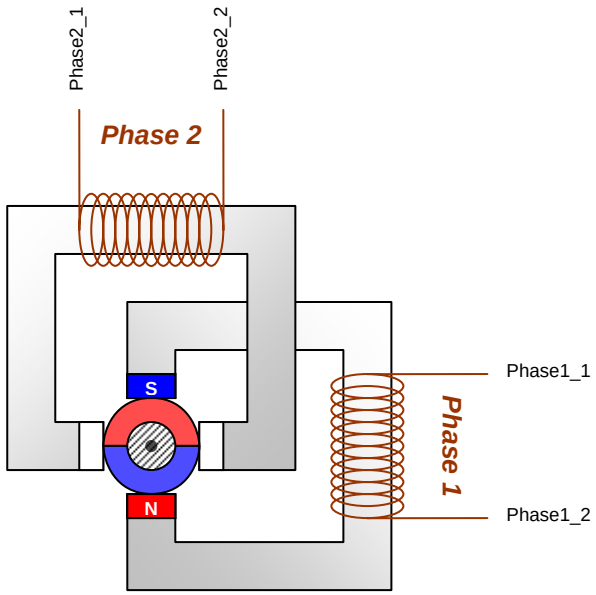


FIG. 75 – Principe de fonctionnement d'un moteur pas à pas bipolaire

Le principe de fonctionnement d'un moteur pas à pas bipolaire est rappelé figure 75. La figure 76 est une photo d'un tel moteur, où l'on peut voir les 4 fils d'alimentation des bobinages du moteur.

FIG. 76 – Photo d'un moteur pas à pas bipolaire

Parmi les signaux en entrée du montage, on a :
- **ClockWise** : commande le sens de rotation du moteur. A l'état logique 1, le moteur tourne dans le sens des aiguilles d'une montre ;
- **Clock** : horloge qui va imposer la vitesse de rotation. Sa fréquence peut aller jusqu'à quelques centaines de Hz selon les cas ;
- **HalfStep** : permet de faire tourner le moteur en demi-pas (à l'état logique 1), ce qui double donc le nombre de pas mais provoque des irrégularités dans le couple du moteur.
- **/Reset** : reset du composant, actif à l'état bas ;
- **Enable** : lorsque cette entrée est mise à 0, le L297 devient inactif (sorties à 0). La résistance de tirage permet de ne pas activer le composant tant que l'unité de contrôle n'est pas initialisée.

Quatrième partie
Réalisations

15 Conception d'un bus

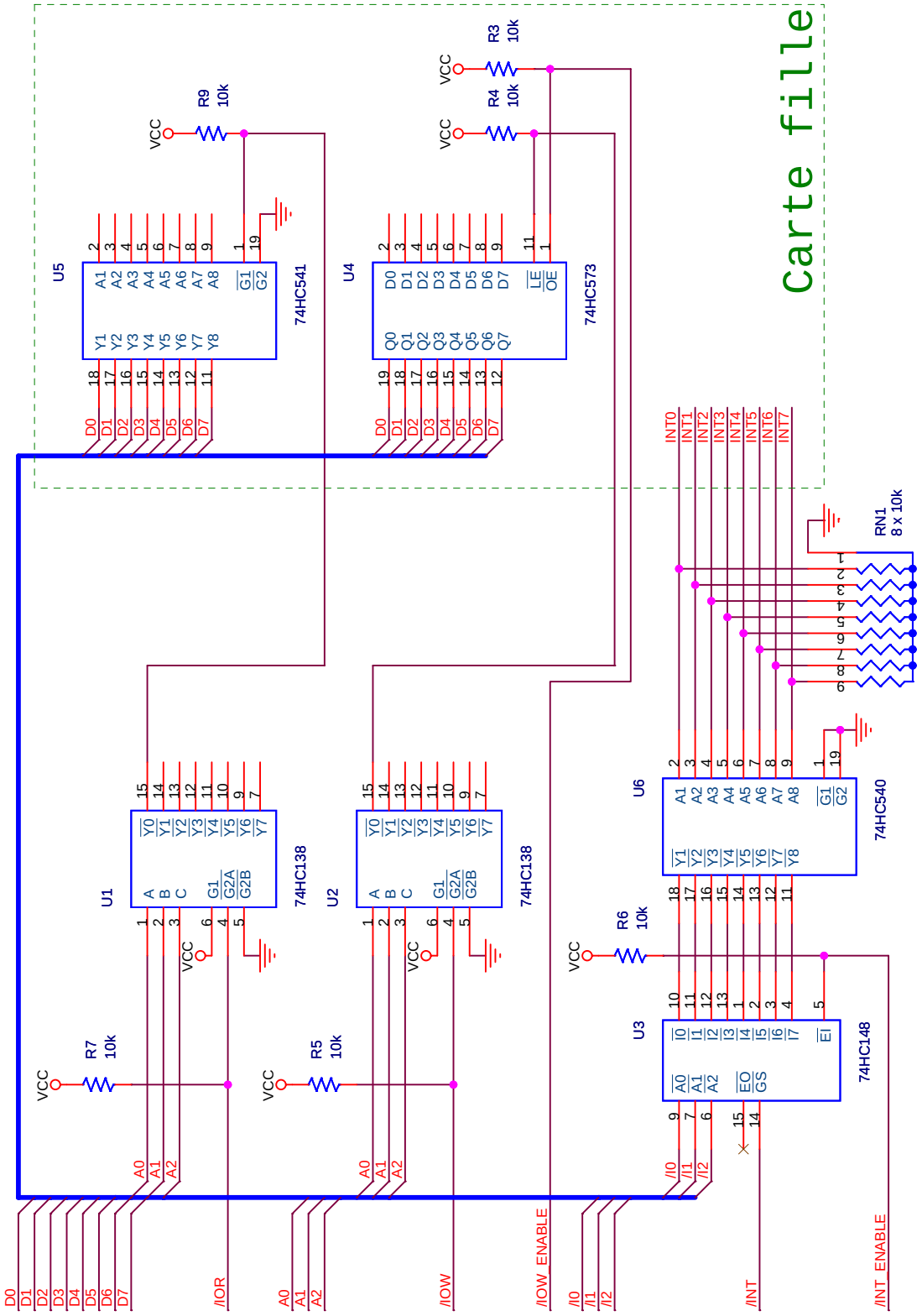


FIG. 77 – Schéma d'un bus avec gestion de la lecture, de l'écriture et des interruptions

Le schéma de la figure 77 représente un bus complet, avec une carte fille connectée dessus. Pour comprendre le fonctionnement du bus, nous allons le décomposer en 3 parties : la lecture des données par la carte mère, l'écriture des données par la carte mère et les interruptions générées par la carte fille.

15.1 Lecture des données

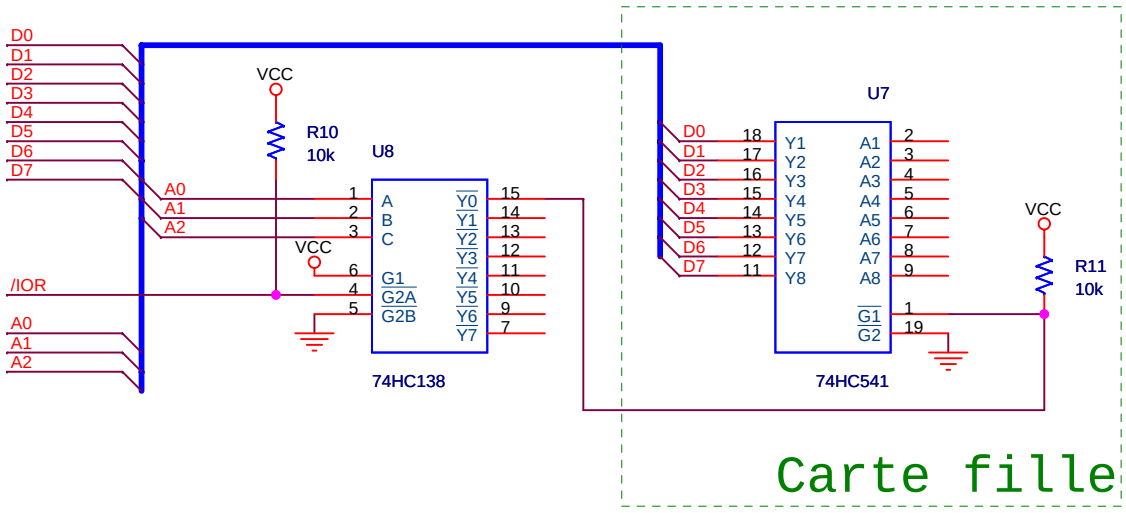


FIG. 78 – Schéma de la gestion de la lecture du bus

La lecture des données est relativement simple : le 138 est un décodeur, ou démultiplexeur, 3 vers 8 [9]. A partir d'une adresse i, codée sur 3 bits en binaire naturel (A0, A1 et A2), la sortie Yi passe de 1 à 0, lorsque l'entrée /IOR est mise à 0 (activation du composant). La sortie Yi va alors activer le buffer de la carte fille correspondante et connecter les entrées Yk aux sortie Ak du 541, et donc établir la liaison entre le bus interne de la carte fille et le bus de donnée D0 à D7. Le 541 est un buffer 8 bits à 3 états [7]. Les autres 541 n'étant alors pas activés, leurs sorties se trouvent en haute impédance, ce qui interdit tout conflit avec les autres cartes filles.

La séquence de lecture se déroule comme suit :

- Mettre l'adresse dans A0, A1, A2 ;
- Mettre /IOR à 0 (début de lecture) ;
- Lire les données sur D0, ..., D7 ;
- Mettre /IOR à 1 (fin de lecture).

Listing 3 – Implémentation en C pour la lecture sur le bus

```
1  // Programmé au départ pour le PIC18F452
2
3  #define IO_ADDR      PORTE    // Bus d'adresse
4  #define IO_DATA      PORTD    // Bus de donnée
5  #define IO_NIOR      PORTAbits.RA0
6
7  // Initialisation (à placer dans la fonction main ou dans une fonction d'initialisation
       du bus)
8  TRISE = 0b00000000;
9  TRISAbits.TRISA0 = 0;
10 IO_NIOR = 1;
11
12 // Lit le bit bitNb sur le bus
13 char readBusBit(char addr, char bitNb) {
14   char data;
15
16 //  TRISD = 0b11111111;   // Port D en entrée -- c'est le cas par défaut
17   IO_ADDR = addr;         // Ecriture de l'adresse
18   IO_NIOR = 0;            // Mise en lecture
19   data = (IO_DATA << bitNb) & 1;    // Lit les données
20   IO_NIOR = 1;            // Fin de lecture
21
22   return data;
```

```
23 }
24
25 // Lit le bus en entier (8 bits)
26 unsigned char readBus(char addr) {
27   unsigned char data;
28
29 //  TRISD = 0b11111111;   // Port D en entrée — c'est le cas par défaut
30   IO_ADDR = addr;         // Ecriture de l'adresse
31   IO_NIOR = 0;            // Mise en lecture
32   data = IO_DATA;         // Lit les données
33   IO_NIOR = 1;            // Fin de lecture
34
35   return data;
36 }
```

Le fait d'utiliser les ports E et D du PIC simplifie grandement la tâche, car l'on a pas besoin de travailler bit par bit, la taille des ports étant adaptée (3 bits et 8 bits respectivement).

Vous remarquerez sur la figure 78 la résistance de tirage au niveau de l'entrée /G2A du 138. Cela permet d'avoir par défaut le bus en haute impédance, tant que l'unité de contrôle n'a pas été initialisée.

15.2 Écriture des données

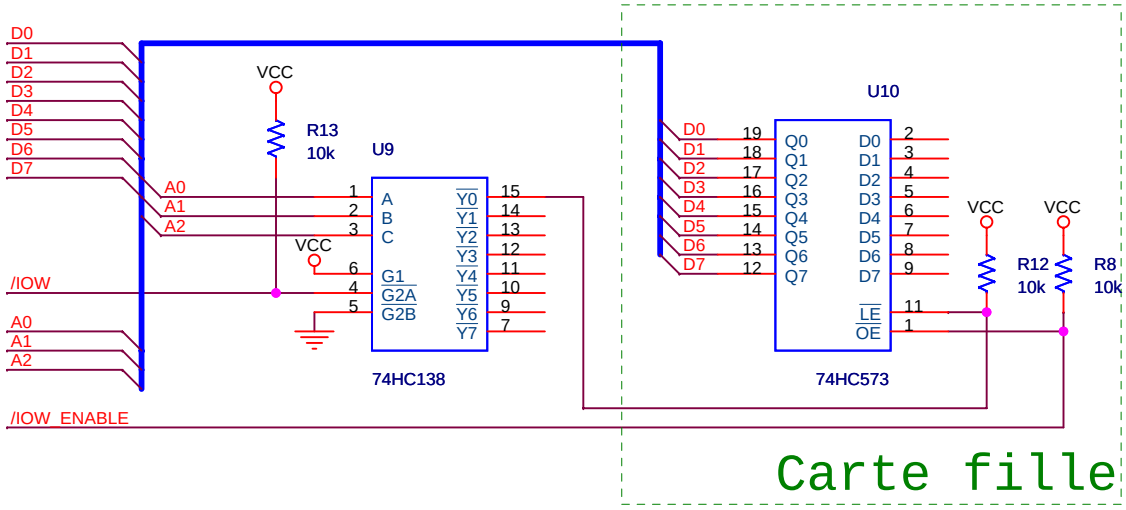


FIG. 79 – Schéma de la gestion de l'écriture du bus

Le principe de fonctionnement de la gestion de l'écriture sur le bus est très similaire à la lecture. Le 138 [9] joue le même rôle que dans le montage précédent, mais cette fois ci, on utilise un 573 sur les cartes filles. Le 573 est un latch de 8 bits, constitué de 8 bascules D [8]. Il mémorise donc la valeur logique des entrées, qui restent toujours disponibles en sortie quand bien même les entrées passent en haute impédance pour une opération d'écriture sur un autre 573.

15.2.1 L'initialisation

S'agissant d'un bus d'écriture des données, son initialisation est nécessaire. Si l'on activait les 573 à la mise sous tension, leurs sorties seraient dans un état indéterminé, puisque les entrées seront au départ vraisemblablement dans un état de haute impédance (si elles sont reliées directement à un PIC par exemple, avant que celui-ci n'ait initialisé les entrées). C'est la raison d'être du signal /IOW_ENABLE. Une résistance de tirage assure que par défaut, son état logique soit à 1, et donc que les sorties des 573 soient en haute impédance. Le PIC va ensuite initialiser les entrées des différents 573 (qui rappelons-le sont mémorisées) et à la fin seulement activer leurs sorties en mettant /IOW_ENABLE à 0 (en activant les tampons 3 états).

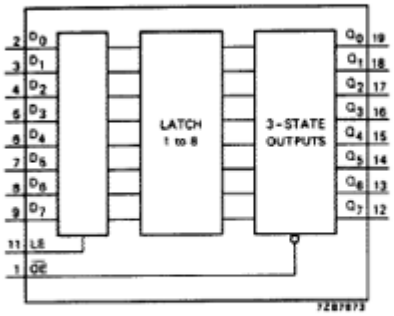


Fig.4 Functional diagram.

FUNCTION TABLE

OPERATING MODES	INPUTS			INTERNAL LATCHES	OUTPUTS
	$\overline{OE}$	LE	D_N		Q_0 to Q_7
enable and read register (transparent mode)	L	H	L	L	L
	L	H	H	H	H
latch and read register	L	L	l	L	L
	L	L	h	H	H
latch register and disable outputs	H	L	l	L	Z
	H	L	h	H	Z

Notes

1. H = HIGH voltage level
   h = HIGH voltage level one set-up time prior to the HIGH-to-LOW LE transition
   L = LOW voltage level
   l = LOW voltage level one set-up time prior to the HIGH-to-LOW LE transition
   Z = high impedance OFF-state

FIG. 80 – Extrait de la datasheet du 74HC573

Un extrait de la datasheet du 573 [8], figure 80, permet de bien saisir le fonctionnement du composant. Voici un exemple de procédure d'initialisation et le code C correspondant :

- Pour chaque adresse (A0, A1, A2) :
  - Mettre D0 à D7 en sortie ;
  - Écrire les données sur D0, ..., D7 ;
  - Mettre /IOW à 0 (début de l'écriture) ;
  - Remettre /IOW à 1 (fin de l'écriture) ;
  - Remettre D0 à D7 en entrée.
- Mettre /IOW_ENABLE à 0 (activation des sorties).

Listing 4 – Implémentation en C pour l'initialisation en écriture du bus

```
1  // Programmé au départ pour le PIC18F452
2
3  #define IO_ADDR           PORTE    // Bus d'adresse
4  #define IO_DATA           PORTD    // Bus de donnée
5  #define IO_NIOW           PORTAbits.RA1
6  #define IO_NIOW_ENABLE    PORTAbits.RA2
7
8  // Initialisation (à placer dans la fonction main ou dans une fonction d'initialisation
       du bus)
9  TRISE = 0b00000000;
10 TRISAbits.TRISA1 = 0;
11 IO_NIOW = 1;
12 TRISAbits.TRISA2 = 0;
13 IO_NIOW_ENABLE = 1;
14
15 // Variable stockant pour chaque adresse l'état du bus (nécessaire pour l'écriture bit
       par bit sur le bus, voir la fonction writeBusBit)
16 volatile unsigned char busData[8];
17
18 void initBus() {
19   char i;
20
21   for (i = 0; i < 8; i++) {
22     busData[i] = 0;
23
24     IO_ADDR = i;
25     TRISD = 0b00000000;    // Port D en sortie
26     IO_DATA = 0;           // Met le bus à 0
27     IO_NIOW = 0;           // Mise en écriture
28     IO_NIOW = 1;           // Fin d'écriture
```

```
29      TRISD = 0b11111111;    // Remet le port D en entrée
30    }
31
32    IO_NIOW_ENABLE = 0;
33  }
```

Il est important de savoir que sans le signal /IOW_ENABLE, l'initialisation des 573 peu être aléatoire lors de la mise sous tension. Donc si vous ne voulez pas que votre robot se mette à rouler tout seul lorsque vous le branchez, pensez-si !

15.2.2 L'écriture

La procédure d'écriture est donc la suivante :
- Mettre l'adresse dans A0, A1, A2 ;
- Écrire les données sur D0, ..., D7 ;
- Mettre /IOW à 0 (début de l'écriture) ;
- Mettre /IOW à 1 (fin de l'écriture).

Listing 5 – Implémentation en C pour l'écriture sur le bus

```
1  // La fonction writeBusBit permet de ne modifier qu'un seul bit sur le bus et demande
       pour cela de stocker en mémoire pour chaque adresse l'état du bus.
2  void writeBusBit(char addr, char bitNb, char bitValue) {
3    if (((busData[addr] >> bitNb) & 1) == 0 && bitValue == 1)
4      busData[addr] = busData[addr] | (1 << bitNb);
5    else if (((busData[addr] >> bitNb) & 1) == 1 && bitValue == 0)
6      busData[addr] = busData[addr] ^ (1 << bitNb);
7
8    IO_ADDR = addr;              // Ecriture de l'adresse
9    TRISD = 0b00000000;          // Port D en sortie
10   IO_DATA = busData[addr];     // Ecrit les données
11   IO_NIOW = 0;                 // Mise en écriture
12   IO_NIOW = 1;                 // Fin d'écriture
13   TRISD = 0b11111111;          // Remet le port D en entrée
14 }
15
16 void writeBus(char addr, unsigned char data) {
17   busData[addr] = data;
18
19   IO_ADDR = addr;          // Ecriture de l'adresse
20   TRISD = 0b00000000;      // Port D en sortie
21   IO_DATA = data;          // Ecrit les données
22   IO_NIOW = 0;             // Mise en écriture
23   IO_NIOW = 1;             // Fin d'écriture
24   TRISD = 0b11111111;      // Remet le port D en entrée
25 }
```

15.3 Les interruptions

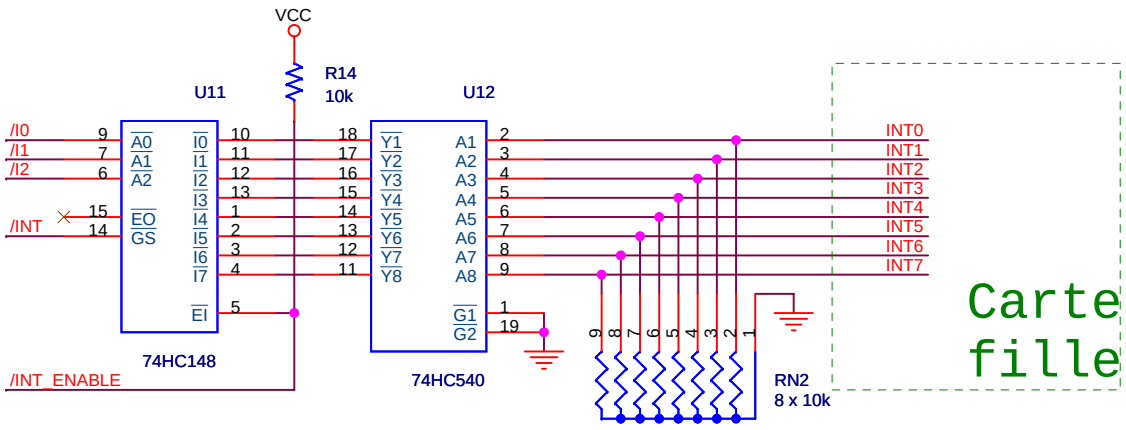


FIG. 81 – Schéma de la gestion des interruptions du bus

La gestion des interruptions est relativement simple : lors d'un front montant sur une des broches INTi, provoqué par une carte fille, le décodeur, ou démultiplexeur 148 [2] va générer un front descendant sur /INT et l'adresse de l'interruption sera donnée par les 3 bits /I0, /I1 et /I2, tout cela à condition que le 148 soit actif, c'est-à-dire à condition que /INT_ENABLE soit mis à 0.

15.3.1 Activation des interruptions

Listing 6 – Activation des interruptions pour le PIC18F452

```
#define IO_INT_NENABLE  PORTBbits.RB1
#define IO_INT_ADDR     ((~PORTCbits.RC1 & 1) | ((~PORTCbits.RC2 & 1) << 1) | ((~
    PORTCbits.RC3 & 1) << 2))
// La définition de IO_INT_ADDR est un peu compliquée, mais rien n'est superflu, sinon ça
     ne marche pas...

void main (void) {
  // ... début du programme ...

  TRISBbits.TRISB0 = 1;      // RB0 en entrée (/INT)
  TRISBbits.TRISB1 = 0;      // RB1 en sortie (/INT_ENABLE)

  // On autorise toutes les interruptions
  INTCONbits.GIE = 1;
  INTCONbits.PEIE = 1;
  INTCONbits.INT0IE = 1;     // Active l'interruption sur RB0 (/INT)
  INTCON2bits.INTEDG0 = 0;   // Interruption sur front descendant

  IO_INT_NENABLE = 0;        // Activation des interruptions

  // ... suite du programme ...
}
```

15.3.2 Gestion des interruptions

Listing 7 – Gestion des interruptions pour le PIC18F452

```
void highISR(void) {
  unsigned char sProdL;
  unsigned char sProdH;

  // Sauvegarde du contenu des registres de calcul
  sProdL = PRODL;
  sProdH = PRODH;

  // Interruption provenant du 148
  if (INTCONbits.INT0IF) {
    if (IO_INT_ADDR == 0) {
      // ... instructions ...
    }
    else if (IO_INT_ADDR == 1) {
      // ... instructions ...
    }
    // else if ...

    // On réautorise l'interruption
    INTCONbits.INT0IF = 0;
  }

  // Restauration des registres de calcul
  PRODL = sProdL;
  PRODH = sProdH;
}
```

16 Système de positionnement

Ce système permet de calculer la position absolue x y d'un robot, ainsi que son cap. Pour cela, le système présenté utilise 3 balises émettrices et une tourelle de réception rotative sur le robot. Chaque

balise émet un signal avec un codage différent, qui est réceptionné par la tourelle, celle-ci recevant le signal quand elle est parfaitement alignée avec une balise, ce qui permet de mesurer l'angle γ_i entre la balise i et la référence du robot. Connaissant les 3 angles des balises, il est alors possible de trianguliariser la position du robot.

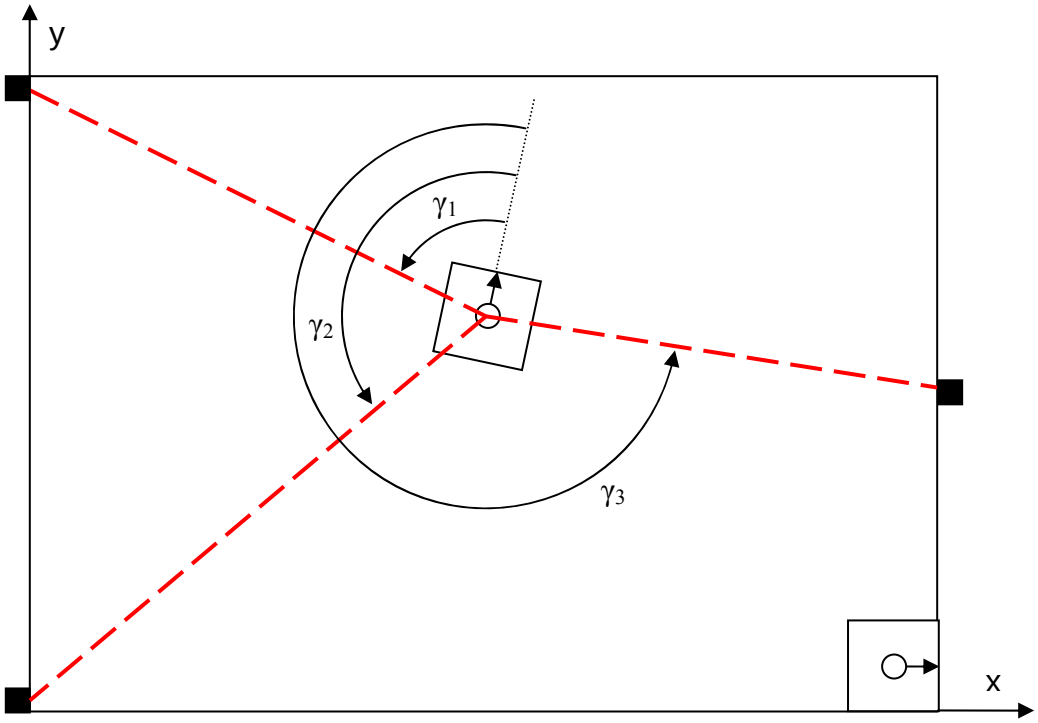


FIG. 82 – Système à 3 balises émettrices

16.1 Principe de triangularisation

Calcul de la position La figure 83 montre une paramétrisation possible du problème, qui sera utilisée pour les calculs.

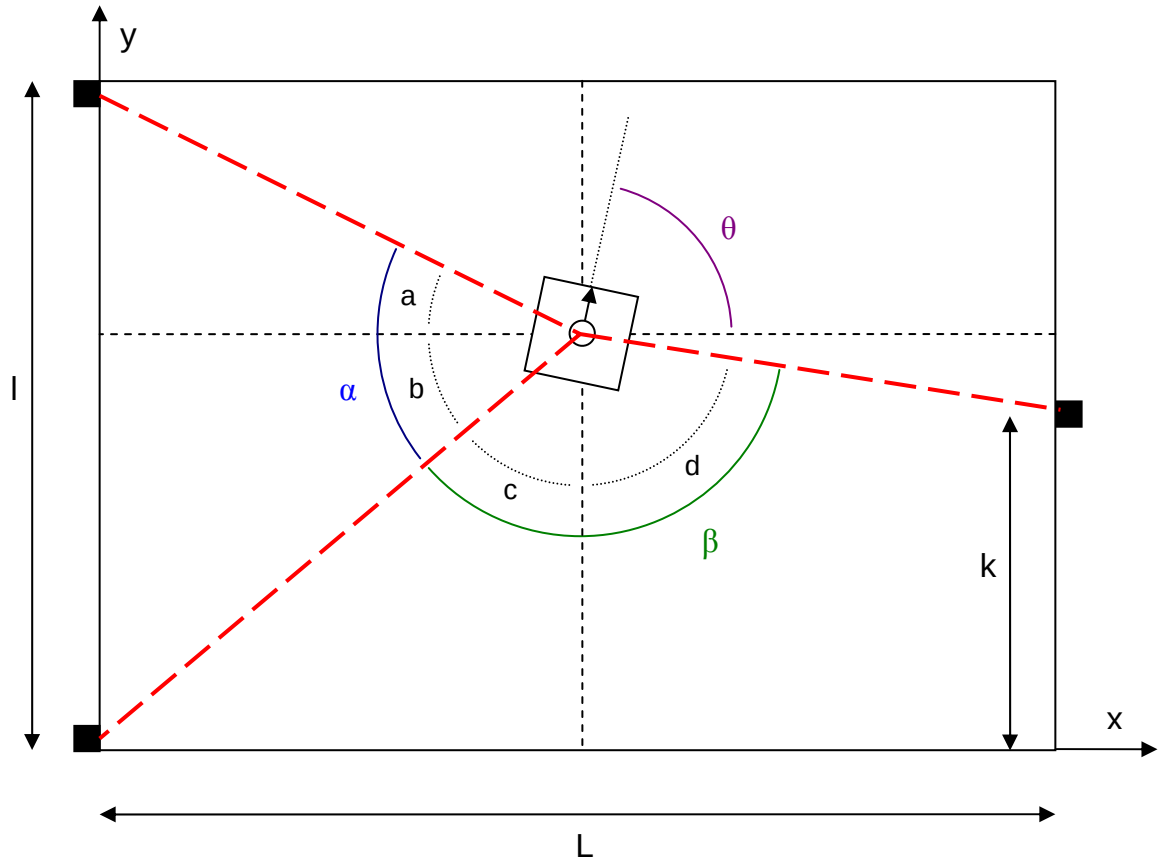


FIG. 83 – Paramétrisation du système à 3 balises émettrices

4 premières équations sont données par de la trigonométrie élémentaire :

$$tan(a) = \frac{l-y}{x} \; ; \; tan(b) = \frac{y}{x} \; ; \; tan(c) = \frac{x}{y} \; ; \; tan(d - \frac{\pi}{2}) = \frac{k-y}{L-x}$$

En les combinant, on obtient un système de deux équations à deux inconnues :

$$\alpha = arctan\left(\frac{l-y}{x}\right) + arctan\left(\frac{y}{x}\right)$$

$$\beta - \frac{\pi}{2} = arctan\left(\frac{x}{y}\right) + arctan\left(\frac{k-y}{L-x}\right)$$

La relation $tan(a+b) = \frac{tan(a)+tan(b)}{1-tan(a)tan(b)}$ permet de se débarrasser des tangentes et on trouve après simplification (avec $tan(\beta - \frac{\pi}{2}) = -\frac{1}{tan(\beta)}$) :

$$A = tan(\alpha) = \frac{lx}{x^2+y^2-ly}$$

$$B = tan(\beta) = \frac{Ly-kx}{x^2+y^2-ky-Lx}$$

Il est possible de résoudre ce système à la main, mais c'est (très) long... Un logiciel de calcul formel y arrive sans difficulté.

$$x = \frac{(lB-Bk-L)(-Ak+ABL-Bk-L)Al}{B^2l^2+A^2l^2B^2+A^2B^2k^2+A^2L^2+A^2k^2+B^2A^2L^2-2A^2lB^2k-2A^2lBL+2BlAk-2B^2lAL}$$

$$y = \frac{l(-Ak+ABL-Bk-L)(-lB-Ak+ABL)}{B^2l^2+A^2l^2B^2+A^2B^2k^2+A^2L^2+A^2k^2+B^2A^2L^2-2A^2lB^2k-2A^2lBL+2BlAk-2B^2lAL}$$

Avec $A = tan(\gamma_2 - \gamma_1)$ et $B = tan(\gamma_3 - \gamma_2)$.

Listing 8 – Résolution du système avec Maple 9

```
> sols:= factor(solve({A=l*x/(x^2+y^2-l*y), B=(L*y-k*x)/(x^2+y^2-k*y-L*x)}, {x, y}));
```

L'avantage de cette méthode est que le calcul de x et y est ramené à une fraction polynomiale et ne nécessite aucun appel à des fonctions trigonométriques, mise à part pour le calcul des tangentes de α et β. Il est possible de traiter le problème de façon plus géométrique, en faisant appel aux fonctions trigos, mais le résultat est alors plus lourd pour le calcul numérique.

Il est avantageux d'utiliser le rapport y/x, qui est simple, pour optimiser les calculs :

$$r = \frac{y}{x} = \frac{-lB - Ak + ABL}{(lB - Bk - L)A}$$

Calcul du cap Un fois que l'on a la position du robot, il devient très facile de calculer son cap, de différentes manières. En voici une qui exploite le rapport y/x :

$$\theta = \pi + b - \gamma_2 = \pi + arctan\left(\frac{y}{x}\right) - \gamma_2 = \pi + arctan(r) - \gamma_2$$

Implémentation en C La simplicité du programme en C ne doit pas masquer le fait qu'il demande pas mal de ressources au PIC, qui ne peut pas effectuer nativement d'opération en 32 bits.

Listing 9 – Implémentation du calcul de la position et du cap en C

```
#include <p18f452.h>  // déclarations pour le PIC18F452
#include <math.h>     // fonctions de math, pour tan(x) et atan(x)

volatile int posX;    // x en mm
volatile int posY;    // y en mm
volatile int cap;     // cap en °

void calcPos(double g1, double g2, double g3) {
  double A, B, C, R;

  A = tan(g2-g1);
  B = tan(g3-g2);

  // Les valeurs numériques sont pour l = 2.1, L = 3 et k = l/2 = 1.05
  // Factoriser au maximum les polynômes permet de gagner en temps de calcul
  C = A*(1.05*B-3);    // Optimisation (évite un calcul redondant)
  R = (A*(3*B-1.05)-2.1*B)/C;
  posX = 1000*C*(A*(-2.205+6.3*B)-2.205*B-6.3)/(A*(B*(A*(10.1025*B-12.6)+4.41-12.6*B)
      +10.1025*A)+4.41*B*B);
  posY = R*posX;
  cap = 180+(atan(R)-g2)*57.30;
}
```

Précision des mesures Il est possible de majorer l'incertitude en (x_0, y_0), $\Delta x_{(x_0,y_0)}$, connaissant l'incertitude sur la mesure des angles γ_i, $\Delta\gamma_i$:

$$\Delta x_{(x_0,y_0)} = \left|\frac{\partial x}{\partial \gamma_1}_{(\gamma_1(x_0,y_0),\gamma_2(x_0,y_0),\gamma_3(x_0,y_0))}\right| \Delta\gamma_1 + \left|\frac{\partial x}{\partial \gamma_2}_{(\ldots)}\right| \Delta\gamma_2 + \left|\frac{\partial x}{\partial \gamma_3}_{(\ldots)}\right| \Delta\gamma_3$$

En pratique, l'incertitude dépendant de la position sur le terrain, on ne peut que calculer l'incertitude par rapport à la position mesurée et non la position exacte (inconnue) du robot. L'incertitude ainsi calculée n'est donc qu'une estimation de l'incertitude réelle, qui est d'autant meilleure que celle-ci varie peu à cet endroit du terrain !

La figure 84 montre l'allure de l'incertitude sur x suivant la position du robot sur le terrain, pour une incertitude sur les angles γ_i de 1°. Il faut garder à l'esprit que ceci n'est qu'une majoration de l'incertitude.

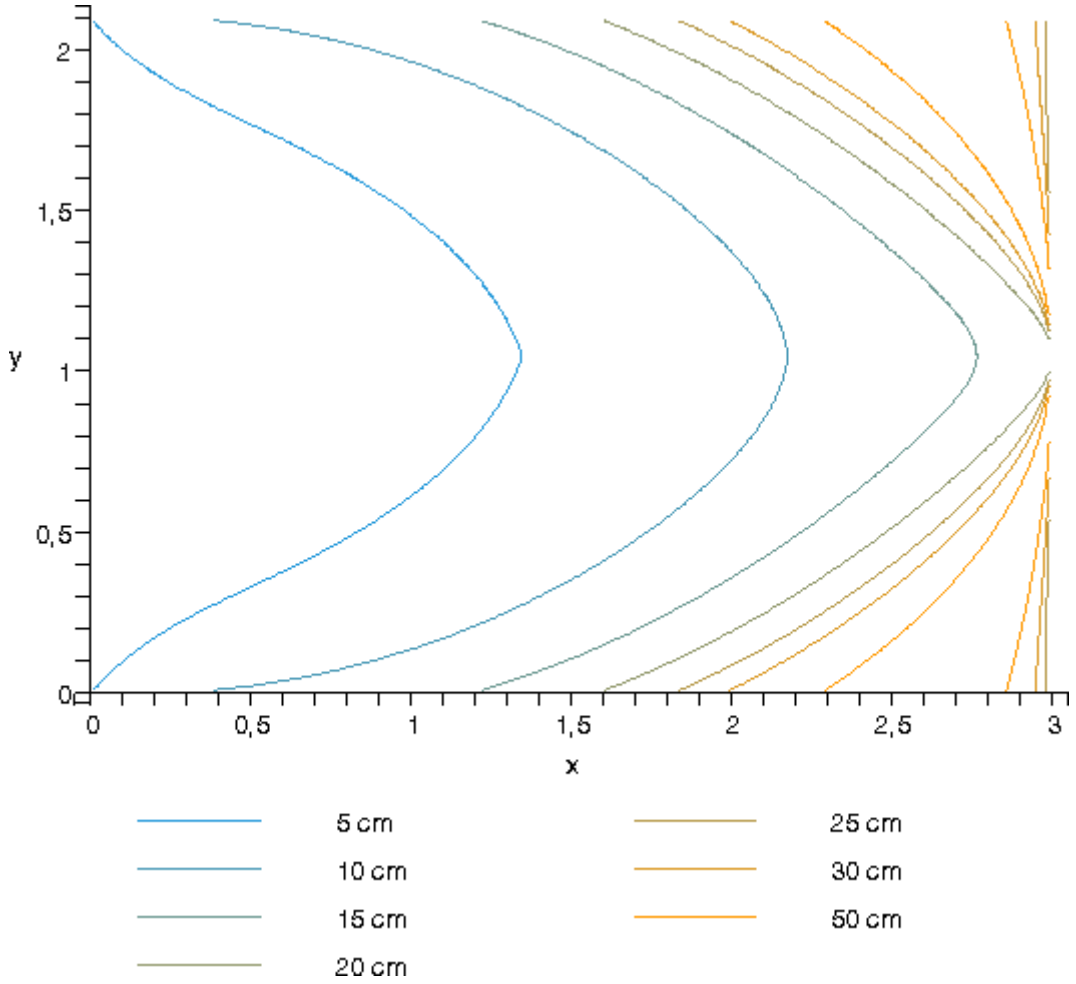


FIG. 84 – Incertitude sur x suivant la position sur le terrain

L'incertitude peut être très importante et ceci est largement dû à un placement non optimum des balises (qui était, cela dit, celui de la Coupe de France de robotique de 2005).

Le programme 10 permet de calculer l'incertitude sur x, y et θ.

Listing 10 – Calcul de l'incertitude sous Maple 9

```
> restart;
  with(plots):
> L:= 3;
  l:= 2.1;
  k:= l/2;
> A:= tan(g2-g1):
  B:= tan(g3-g2):
> x:= unapply((l*B-L-B*k)*(A*B*L-A*k-B*k-L)*A*l/(B^2*l^2+B^2*A^2*l^2+A^2*L^2-2*B*A^2*l*L
    -2*B^2*A^2*l*k+A^2*B^2*k^2+B^2*A^2*L^2+A^2*k^2-2*B^2*l*A*L+2*B*l*A*k), g1, g2, g3):
  y:= unapply(l*(A*B*L-A*k-B*k-L)*(-l*B+A*B*L-A*k)/(B^2*l^2+B^2*A^2*l^2+A^2*L^2-2*B*A^2*l
      *L-2*B^2*A^2*l*k+A^2*B^2*k^2+B^2*A^2*L^2+A^2*k^2-2*B^2*l*A*L+2*B*l*A*k), g1, g2, g3
      ):
  cap:= unapply(Pi+arctan(y(g1, g2, g3)/x(g1, g2, g3))-g2, g1, g2, g3):
> dx:= unapply(abs(diff(x(g1, g2, g3), g1))*dg1 + abs(diff(x(g1, g2, g3), g2))*dg2 + abs(
    diff(x(g1, g2, g3), g3))*dg3, g1, g2, g3, dg1, dg2, dg3):
  dy:= unapply(abs(diff(y(g1, g2, g3), g1))*dg1 + abs(diff(y(g1, g2, g3), g2))*dg2 + abs(
      diff(y(g1, g2, g3), g3))*dg3, g1, g2, g3, dg1, dg2, dg3):
  dcap:= unapply(abs(diff(cap(g1, g2, g3), g1))*dg1 + abs(diff(cap(g1, g2, g3), g2))*dg2
      + abs(diff(cap(g1, g2, g3), g3))*dg3, g1, g2, g3, dg1, dg2, dg3):
> alpha:= unapply(arctan((l-y)/x) + arctan(y/x), x, y):
  beta:= unapply(arctan(x/y) + Pi/2 + arctan((k-y)/(L-x)), x, y):
> g1:= unapply(0, x, y):
  g2:= g1 + alpha:
  g3:= g1 + alpha + beta:
> viewX:= 0.01..L-0.01:
```

```
20   viewY:= 0.01..l-0.01:
21   contXY:= [5, 10, 15, 20, 25, 30, 50]:
22   contCap:= [1.5, 2, 5, 10, 15, 20]:
23 > precX:= [seq(implicitplot(dx(g1(x, y), g2(x, y), g3(x, y), 0.018, 0.018, 0.018)=contXY[
       i]/100, x=viewX, y=viewY, numpoints=10000, labels=["x", "y"], color=COLOR(RGB, i/7,
       0.6, 1-i/7), legend=cat(convert(contXY[i], string), " cm")), i=1..7)]:
24   display(precX);
25 > precY:= [seq(implicitplot(dy(g1(x, y), g2(x, y), g3(x, y), 0.018, 0.018, 0.018)=contXY[
       i]/100, x=viewX, y=viewY, numpoints=10000, labels=["x", "y"], color=COLOR(RGB, i/7,
       0.6, 1-i/7), legend=cat(convert(contXY[i], string), " cm")), i=1..7)]:
26   display(precY);
27 > precCap:= [seq(implicitplot(dcap(g1(x, y), g2(x, y), g3(x, y), 0.018, 0.018, 0.018)=
       contCap[i]*Pi/180, x=viewX, y=viewY, numpoints=10000, labels=["x", "y"], color=COLOR(
       RGB, i/6, 0.6, 1-i/6), legend=cat(convert(contCap[i], string), "°")), i=1..6)]:
28   display(precCap);
```

Cinquième partie
Mécanique

17 La motorisation

17.1 Dimensionner ses moteurs

Voici quelques bases pour dimensionner correctement un moteur et choisir notre modèle en fonction des données constructeurs, à savoir le couple, la vitesse, le rapport de réduction... etc.

Mais avant de commencer, il faut déjà connaitre certains paramètres de notre robot :

- Le diamètre des roues D (en m) ;
- La vitesse maximale du robot V_{max} (en m/s) ;
- Sa masse m (en kg). C'est sans doute le paramètre le plus difficile à estimer, dans la mesure où l'on ne sait pas précisément à l'avance combien pèsera le robot une fois terminé. Les batteries, les motoréducteurs ainsi que l'armature constituent la plus grande partie de la masse du robot ;
- L'angle de la plus grande pente à franchir, θ_{max} (en °).

17.1.1 Vitesse de rotation et rapport de réduction du moteur

Calculons la vitesse de rotation maximale des roues Vr_{max} (en tr/mn) en fonction de la vitesse maximale du robot :

$$Vr_{max} = \frac{60 \times V_{max}}{\pi \times D}$$

Vr_{max} correspond donc à la vitesse de rotation en sortie du réducteur. La vitesse de rotation du moteur Vm (en tr/mn) est alors donnée par $Vm = Vr_{max}/R$, où R est le rapport de réduction (inférieur à 1).

Exemple : on veut une vitesse maximale de 1 m/s, pour des roues de 10 cm de diamètre et l'on dispose d'un réducteur de $1/20^{e}$. La vitesse de rotation du moteur doit alors être de $Vm = \frac{60\times 1}{\pi\times 0{,}1}/(1/20) = 3820$ tr/mn.

Fig. 85 – Motoréducteur (12 ou 24 V)

Moteur et réducteur ne forment généralement qu'un seul bloc appelé motoréducteur, comme sur la figure 85. S'il faut choisir indépendamment le moteur et le réducteur, la vitesse de rotation du moteur se déterminera donc en fonction des rapports de réduction des blocs réducteur disponibles.

17.1.2 Couple minimum du moteur

Le calcul du couple minimum que doit pouvoir fournir le moteur fait intervenir la masse du robot et l'angle de la plus grande pente à franchir. Pour simplifier le calcul, on supposera qu'il n'y a pas de frottement entre la roue et le sol. On suppose de plus que l'ensemble du poids du robot est équitablement réparti sur les 2 roues motrices.

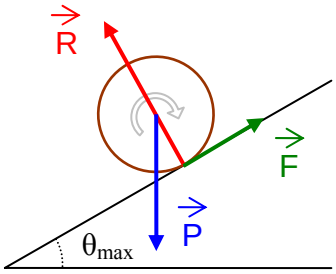


FIG. 86 – Inventaire des forces au niveau d'une roue du robot

La figure 86 fait l'inventaire des forces sur une roue motrice. Une projection suivant la direction de la force $\vec{F}$ nous donne :

$$F = \frac{m}{2} \times g \times sin(\theta_{max}) + \frac{m}{2} \times a$$

Où a est l'accélération du robot, que nous supposerons constante, pour atteindre la vitesse V_{max} en un temps donné, Δt (le robot étant initialement à l'arrêt).

On a $V = a \times t$, ce qui nous donne $a = V_{max}/\Delta t$. On en déduit alors le couple minimum nécessaire en sortie du réducteur Cr_{min}, qui permettra au robot de répondre au cahier des charges :

$$Cr_{min} = F \times \frac{D}{2} = (g \times sin(\theta_{max}) + \frac{V_{max}}{\Delta t}) \times \frac{m}{2} \times \frac{D}{2}$$

Le couple minimum en sortie du moteur Cm_{min} est donné par (η étant le rendement du réducteur) :

$$Cm_{min} = \frac{R \times Cr_{min}}{\eta}$$

Exemple : le robot pèse 1 kg, a des roues de 10 cm de diamètre et doit pouvoir gravir des pentes de 20°. Il doit pouvoir atteindre la vitesse V_{max} en 2 s. On dispose d'un réducteur de rapport $1/20^{e}$ et de rendement 0,8. Le couple minimum à la sortie du réducteur est alors $Cr_{min} = (9,81 \times sin(20°) + \frac{1}{2}) \times \frac{1}{2} \times \frac{0,1}{2}$ = 96 mNm et $Cm_{min} = \frac{(1/20) \times 0,096}{0,8}$ = 6 mNm.

17.1.3 Puissance du moteur

Après avoir déterminé la vitesse angulaire et le couple du moteur, on en déduit sa puissance :

$$Pm = Cm \times Vm$$

En reprennant les données des exemples précédents, on trouve $Pm = 2,4$ W.

Sixième partie
Ressources

18 Adresses utiles

18.1 Électronique

- Kelsey Park School Electronics Club (http://www.kpsec.freeuk.com/)
  De nombreuses explications sur les composants courant de l'électronique, qui ont pour certaines été reprises dans ce guide ;
- Electronics in Meccano (http://www.eleinmec.com/)
  Vous découvrirez avec ce site comment intégrer de l'électronique dans les Meccanos, avec de nombreux articles d'électronique ;
- ALL DATASHEET (http://www.alldatasheet.com/)
  Si vous cherchez une datasheet.

18.2 Robotique

- The Fribotte HomePage (http://fribotte.free.fr/)
  Club d'anciens participants à la coupe de France de robotique, avec un base de donnée technique assez fournie ;
- VieArtificielle.com (http://www.vieartificielle.com/)
  Site sur la robotique en général.

18.3 Coupe de robotique

- Planète Sciences, Secteur Robotique (http://www.planete-sciences.org/robot/)
  Le site officiel de la coupe de France de robotique.

18.4 Fournisseurs

- Radiospares (http://www.radiospares.fr/)
  Détaillant sur internet, composants électroniques, électromécaniques, fournitures industrielles... A noter que pour un particulier, le montant d'une commande doit être au minimum de 150 €[4] ! Radiospares étant un partenaire de Planète Sciences, il est possible de bénéficier de 10% de réduction sur toutes les commandes en participant à la coupe de France de robotique[5].
- Selectronic (http://www.selectronic.fr/)
  Composants, outillages, matériel électronique... Il existe aussi des boutiques Selectronic (place de la Nation à Paris par exemple) ;
- Conrad (http://www.conrad.fr/)
  Vente en ligne de produits électroniques et techniques. Plus orienté pour les particuliers (il existe aussi Conrad Pro pour les professionnels - http://www.conradpro.fr/) ;
- MDP Motor (http://www.mdpmotor.com/)
  Cette entreprise propose une gamme complète de moteurs, motoréducteurs et cartes de contrôle. C'est un partenaire de Planète Sciences et à ce titre, il est possible de bénéficier de 20% de réduction sur tous les produits du catalogue MDP en participant à la coupe de France de robotique (voir la note 5), ce qui est loin d'être négligeable compte tenu du prix des moteurs.

[4] Valable en juin 2006.
[5] Valable pour l'édition 2006. Plus de détails à la page de présentation de la coupe sur le site de Planète Sciences (http://www.planete-sciences.org/robot/).

Glossaire

Gravure	La gravure d'une carte est l'action de réaliser les pistes et les empreintes des composants sur celle-ci, par abrasion mécanique ou chimique de la couche de cuivre recouvrant l'une ou les deux faces de la carte.
OrCAD Capture	Logiciel de dessin et de simulation (avec PSpice) de circuits électroniques.
OrCAD Layout	Logiciel permettant de créer le typon d'une carte électronique.
Routage	Le routage d'une carte consiste à placer les pistes reliant les composants sur les différentes couches composant le circuit imprimé.
Typon	En anglais PCB, désigne le dessin d'un circuit imprimé. Pour réaliser un typon, il faut créer et placer les empreintes des composants sur la carte et réaliser le routage de celle-ci.

Références

[1] Semiconductor Components Industries. 2N3055, MJ2955 Complementary Silicon Power Transistors. Datasheet, 2004.

[2] Texas Instruments. SN54HC148, SN74HC148 8-LINE TO 3-LINE PRIORITY ENCODERS. Datasheet, 1997.

[3] Texas Instruments. MAX232, MAX232I DUAL EIA-232 DRIVER/RECEIVER. Datasheet, 1998.

[4] Texas Instruments. TUSB3410 USB To Serial Port Controller. Datasheet, 2002.

[5] Motorola. Amplifier Transistors NPN Silicon BC337,-16,-25,-40 BC338,-16,-25,-40. Datasheet, 1996.

[6] Fairchild Semiconductor. KA78XX/KA78XXA 3-Terminal 1A Positive Voltage Regulator. Datasheet, 2001.

[7] Philips Semiconductors. 74HC/HCT541 Octal buffer/line driver; 3-state. Datasheet, 1990.

[8] Philips Semiconductors. 74HC/HCT573 Octal D-type transparent latch; 3-state. Datasheet, 1990.

[9] Philips Semiconductors. 74HC/HCT138 3-to-8 line decoder/demultiplexer; inverting. Datasheet, 1993.

[10] Microchip Technology. PIC18FXX2 Datasheet. Datasheet, 2002.

[11] CIF (http://www.cif.fr/). F.A.Q. Circuit imprimé. 2006.