PROJETS DE FIN D' ÉTUDES

Années 2006 et 2007

- Régulation Industrielle de Processus
- Système de Régulation de Niveau d' eau
- Interface à base de microprocesseur PIC 16F877
- Commande & Régulation avec LabVIEW

Sakli MOUADH – Génie Électrique & Automatique
Ingénieur diplômé de l' École Nationale d' Ingénieurs de GABÈS
TUNISIE

Vous pouvez charger la dernière mise à jour de ce document à l'adresse suivante :
http://www.bh-automation.fr/Download/Automaticiens/Projets_automatisme_2007_S_MOUADH.pdf

Pour plus d'informations sur Sakli MOUADH :
http://www.bh-automation.fr/Ressources/Automaticiens/#Sakli-MOUADH
http://www.bh-automation.fr/Download/Automaticiens/CV_Sakli_MOUADH.pdf

SOMMAIRE

TABLE DES FIGURES

Chapitre I : Commande numérique et notions de régulation de processus

I : Commande numérique et notions de régulation de processus

I.1- Objectif de régulation automatique

Réguler une grandeur, c'est obtenir d'elle un comportement donné, dans un environnement susceptible de présenter des variations.

On ne peut pas parler de principe de régulation sans parler des lois de commandes.

En faite, les grandeurs physiques commandées varient continûment dans le temps. Pour celles qui ne présentent que 2 états (système binaire ou « tout ou rien », tel les feux de signalisation, les commandes d'ascenseurs, de transfert de pièces par convoyeurs, etc..) en utilise une autre approche différente à la structure de boucle utilisé dans la plupart des systèmes y compris notre système (réservoir).

Les systèmes automatiques assurent en fait 2 types de fonctions :

- Maintenir la grandeur commandée, ou grandeur réglée, à une valeur de référence malgré les variations des conditions extérieurs ; on parle de la régulation en sens strict,
- Répondre à des changements d'objectif, ou à un objectif variable tel-que la poursuite de cible, on parle d'un fonctionnement d'asservissement.

I.2- Principe général de la régulation

Dans la plupart des appareils et installations industrielles, tertiaires et mêmes domestiques, il est nécessaire de maintenir des grandeurs physiques à des valeurs déterminées, en dépit des variations externes ou internes influant sur ces grandeurs. Le niveau d'un réservoir d'eau, la température d'une étuve, le débit d'une conduite de gaz, étant par nature variables, doivent donc être réglés par des actions convenables sur le processus considéré. Si les perturbations influant sur la grandeur à contrôler sont lentes ou négligeables, un simple réglage (dit en boucle ouverte) permet d'obtenir et de maintenir la valeur demandée (par exemple : action sur un robinet d'eau). Dans la majorité des cas, cependant, ce type de réglage n'est pas suffisant, parce que trop grossier ou instable. Il faut alors comparer, en permanence, la valeur mesurée de la grandeur réglée à celle que l'on souhaite obtenir et agir en conséquence sur la grandeur d'action, dite grandeur réglante. On a, dans ce cas, constitué une boucle de régulation et plus généralement une boucle d'asservissement.

Cette boucle nécessite la mise en ouvre d'un ensemble de moyens de mesure, de traitement de signal ou de calcul, d'amplification et de commande d'actionneur,

constituant une chaine d'éléments associés : la chaine de régulation (ou d'asservissement).

Toute chaîne de régulation (ou d'asservissement) comprend trois maillons indispensables : l'organe de mesure, l'organe de régulation et l'organe de contrôle.

Il faut donc commencer par mesurer les principales grandeurs servant à contrôler le processus. L'organe de régulation récupère ces mesures et les compare aux valeurs souhaitées, plus communément appelées valeurs de consigne. En cas de non-concordance des valeurs de mesure et des valeurs de consigne, l'organe de régulation envoie un signal de commande à l'organe de contrôle (vanne, moteur, etc.), afin que celui-ci agisse sur le processus. Les paramètres qui régissent le processus sont ainsi stabilisés en permanence à des niveaux souhaités.

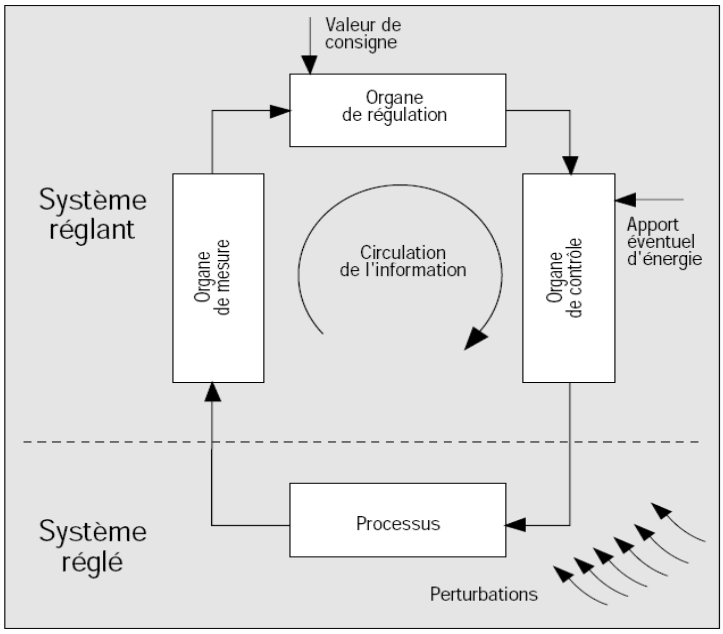


Figure I-0-1 Schéma de principe d'une chaine de régulation

Le choix des éléments de la chaîne de régulation est dicté par les caractéristiques du processus à contrôler, ce qui nécessite de bien connaître le processus en question et son comportement.

I.2.1- Comportement en régulation

La consigne est maintenue constante et il se produit sur le procédé une modification (ou une variation) d'une des entrées perturbatrices.

L'aspect régulation est considéré comme le plus important dans le milieu industriel, car les valeurs des consignes sont souvent fixes. Néanmoins, pour tester les performances et la qualité d'une boucle de régulation, on s'intéresse à l'aspect asservissement.

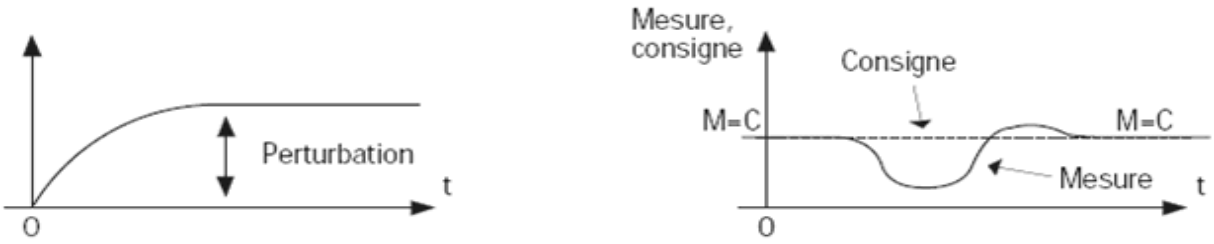


Figure 0-2Comportement en régulation

I.2.2- Comportement en asservissement

L'opérateur effectue un changement de la valeur de la consigne, ce qui correspond à une modification du point de fonctionnement du processus.

Si le comportement en asservissement est correct, on démontre que la boucle de régulation réagit bien, même lorsqu'une perturbation se produit.

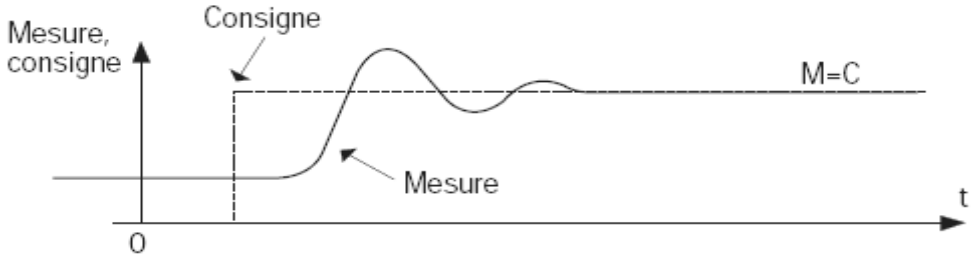


Figure I.3 : Comportement en asservissement

I.3- La régulation en boucle ouverte

Dans un asservissement en boucle ouverte, l'organe de contrôle ne réagit pas à travers le processus sur la grandeur mesurée (celle-ci n'est pas contrôlée). Une régulation en boucle ouverte ne peut être mise en œuvre que si l'on connaît la loi régissant le fonctionnement du processus (autrement dit, il faut connaître la corrélation entre la valeur mesurée et la grandeur réglante).

Contrairement à un asservissement en boucle fermée, un asservissement en boucle ouverte permet d'anticiper les phénomènes et d'obtenir des temps de réponse très courts. De plus, il n'y a pas d'oscillation à craindre (car il s'agit d'un système dynamiquement stable). Enfin, l'asservissement en boucle ouverte est la seule solution envisageable lorsqu'il n'y a pas de contrôle final possible.

Au niveau des inconvénients, la régulation en boucle ouverte impose de connaître la loi régissant le fonctionnement du processus, et il est très fréquent que l'on ne connaisse pas la loi en question. Autre inconvénient sérieux, il n'y a aucun moyen de contrôler, à plus forte raison de compenser, les erreurs, les dérives, les accidents qui peuvent intervenir à l'intérieur de la boucle ; autrement dit, il n'y a pas de

précision ni surtout de fidélité qui dépendent de la qualité intrinsèque des composants. Enfin, la régulation en boucle ouverte ne compense pas les facteurs perturbateurs.

I.4- La régulation en boucle fermée

Dans ce qui vient d'être dit, la variable de sortie (de la chaîne de régulation), ou grandeur réglante, exerce une influence sur la valeur de la variable d'entrée (de la chaîne de régulation) ou variable contrôlée, pour la maintenir dans des limites définies : il s'agit d'une régulation ou d'un asservissement en boucle fermée. L'action de la grandeur réglante sur la variable contrôlée s'opère à travers le "processus" qui boucle la chaîne.

Dans une régulation en boucle fermée, une bonne partie des facteurs perturbateurs sont automatiquement compensés par la contre-réaction à travers le procédé. Autre avantage, il n'est pas nécessaire de connaître avec précision les lois, le comportement des différents composants de la boucle, et notamment du processus, bien que la connaissance des allures statistiques et dynamiques des divers phénomènes rencontrés soit utile pour le choix des composants.

Parmi les inconvénients d'une régulation en boucle fermée, il faut citer le fait que la précision et la fidélité de la régulation dépend de la fidélité et de la précision sur les valeurs mesurées et sur la consigne.

Autre inconvénient, sans doute plus important, le comportement dynamique de la boucle dépend des caractéristiques des différents composants de la boucle, et notamment du processus, enfaîte un mauvais choix de certains composants peut amener la boucle à entrer en oscillation.

Enfin, la régulation en boucle fermée n'anticipe pas. Pour que la régulation envoie une commande à l'organe de contrôle, il faut que les perturbations ou les éventuelles variations de la valeur de consigne se manifestent sur la sortie du processus : ceci peut exiger un délai parfois gênant.

I.5- Les autres formes de régulation

Le but d'une chaîne de régulation est de contrôler un processus. Au niveau des organes de mesure et de contrôle, il n'a pas une très grande marge de manœuvre, car ces organes, dans une certaine mesure, s'imposent souvent d'eux-mêmes.

Il reste un domaine où son savoir-faire va s'exercer pleinement : c'est au niveau de

L'organe de régulation. Bien entendu, les caractéristiques de cet organe vont dépendre du processus à contrôler, des perturbations à prendre en compte, des caractéristiques des organes de mesure et de contrôle.

Bien souvent, les systèmes de régulation comportent, au lieu des chaînes linéaires ouvertes ou fermées, des ensembles de boucles imbriquées dont tout ou partie peut induire des contre-réactions à travers le processus.

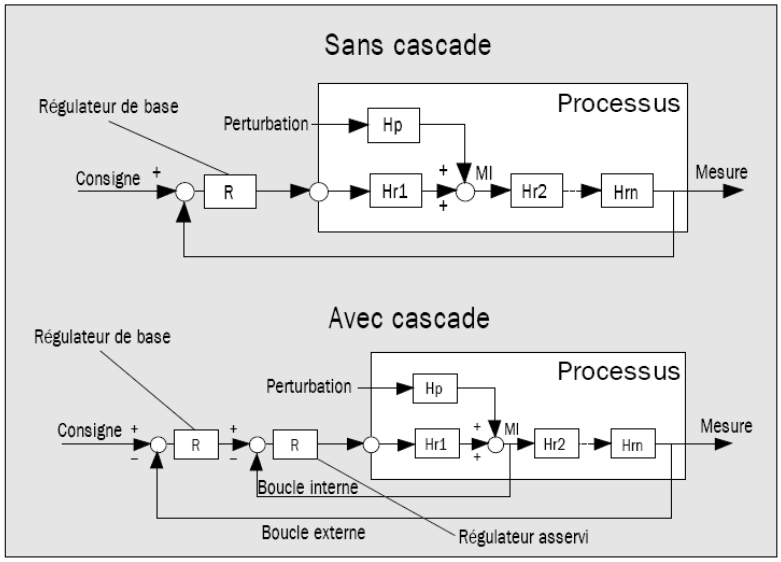


Figure I.4 : exemple de régulation en cascade et sans cascade

I.5.1- Régulation en cascade

L'objectif d'une régulation en cascade est de minimiser les effets d'une ou de plusieurs grandeurs perturbatrices qui agissent soit sur la variable réglante, soit sur une grandeur intermédiaire se trouvant en amont de la variable à régler. Ce type de régulation est intéressant lorsque l'on a affaire à des processus à longs temps de réponse. En effet, quand une perturbation se manifeste, il est nécessaire d'attendre que son influence se ressente au niveau de l'organe de mesure placé en sortie de chaîne. Si les temps de réponse sont longs, la correction n'intervient donc que tardivement, parfois avec la cause qui l'a produite et dont le sens s'est inversé, provoquant oscillations, instabilité, etc.

Bien évidemment, la régulation en cascade n'apporte aucune amélioration si la grandeur perturbatrice se produit en aval de la mesure intermédiaire. Pour que la cascade soit justifiée, il faut que la boucle interne soit beaucoup plus rapide que la boucle externe.

I.5.2- Régulation mixte

Ce type de régulation est l'association d'une régulation en boucle fermée et d'une régulation en boucle ouverte. Les deux boucles sont complémentaires et elles associent leurs actions pour améliorer la stabilité globale. Ce type de régulation est à mettre en œuvre lorsqu'une perturbation affecte directement la grandeur à régler.

I.5.3- Régulation de rapport

Ce type de régulation a par exemple pour objectif d'asservir un débit *Qa* à un autre débit libre *Q1* en imposant entre ces deux débits un facteur de proportionnalité *Kd*, fixé manuellement ou automatiquement.

I.6- Structure et comportement des processus :

Si un processus possède une grandeur réglante U et une grandeur réglée S, son comportement peut être représenté soit par une équation différentielle reliant les valeurs de S et de U en fonction du temps, soit par une représentation dite fonction de transfert déduite de la transformation de Laplace appliquée à cette équation différentielle.

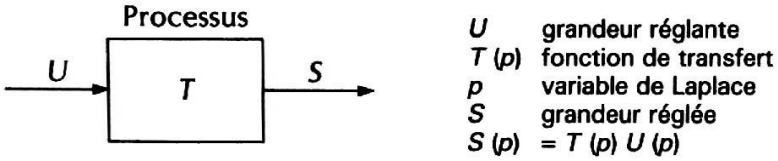


Figure I.5 : fonction de transfert

Dans ce dernier cas, le comportement du processus est décrit par la relation :

$$\boxed{S(p) = T(p)U(p)} \qquad \text{(I.1)}$$

dans laquelle $T(p)$ est la fonction de transfert considérée, p la variable de Laplace . Toutefois, ces relations s'appliquent seulement à des systèmes dont les variables varient de manière continue, et dans lesquels une grandeur réglante U n'exerce son action que sur une seule grandeur réglée S : il s'agit donc de processus mono variables continus.

Les processus industriels, en fait, ne sont pas toujours continus et peuvent être multi variables.

I.6.1- Comportement statique des processus

Lorsque le processus est dans un état stable, à chaque valeur U de la grandeur réglante correspond une valeur S de la grandeur réglée. Deux valeurs voisines U1 et U2 correspondent deux valeurs S1 et S2 de la grandeur réglée, telles que :

$$\frac{(S2 - S1)}{U2 - U1} = G0 \qquad \text{(I.2)}$$

G0 est appelé gain statique du processus. La valeur de celui-ci peut dépendre du point de fonctionnement considéré et n'est donc pas forcément constante dans tout le domaine de variation des grandeurs du processus. On dit alors que le processus est non linéaire. La courbe caractéristique $S(u)$ peut elle-même dépendre d'autres paramètres. Sur la figure (I.6) est représenté un réseau de courbes $S(u)$ dépendant d'un paramètre α. Il arrive aussi que le gain statique dépende du sens de l'action de la grandeur réglante : on a alors affaire à un phénomène d'hystérésis.

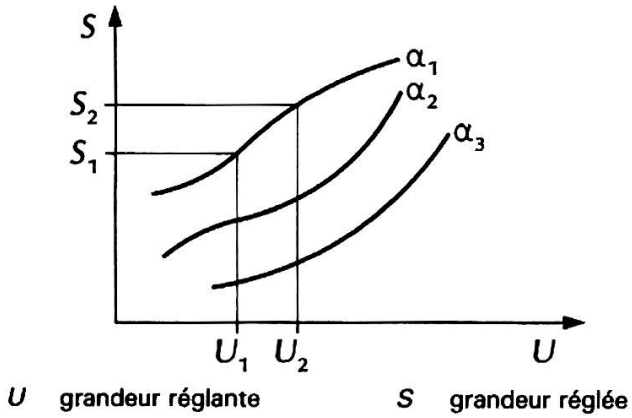


Figure I.6 : Comportement statique de processus

I.6.2- Comportement dynamique des processus

❖ Processus naturellement stables

La réponse naturelle d'un processus à une variation en échelon de sa grandeur réglante permet de savoir s'il est naturellement stable ou instable.

Dans le cas de **procédés stables** (tels que fours et étuves, par exemple), une excitation ΔU de la grandeur réglante produit, à la fin du régime transitoire correspondant, une variation limitée ΔS de la grandeur réglée

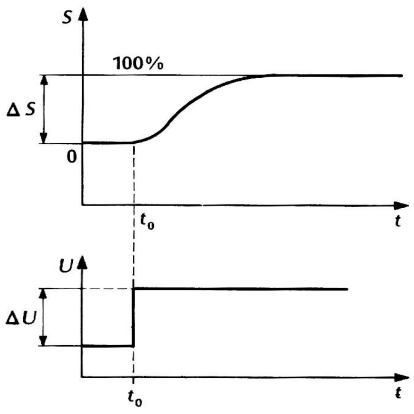


Figure I.7 : Comportement d'un processus naturellement stable

Dans la majorité des cas, les régimes transitoires associés à ce type de réponse ne comportent ni oscillation ni dépassement de la valeur finale ; ces processus sont dits **stables apériodiques**.

Il est possible de représenter ces processus par une relation mathématique générale de la forme :

$$G(p) = \frac{G_0}{(1+\theta_1 p)(1+\theta_2 p)...(1+\theta_n p)} \qquad \text{(I.3)}$$

❖ Processus naturellement instables

Dans un processus naturellement instable, une excitation ΔU de la grandeur réglante produit une variation ΔS de la grandeur réglée dépendant continûment du temps, donc illimitée.

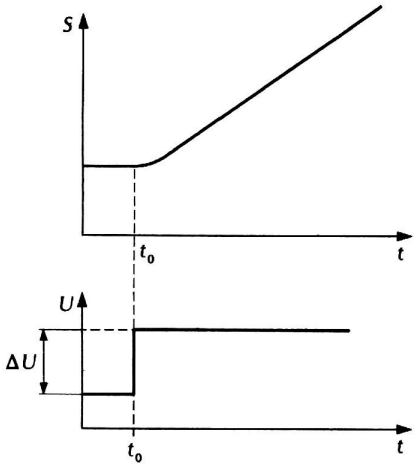


Figure I.8 : Comportement d'un processus naturellement in stable

Un exemple type de ces processus est la commande du niveau d'eau dans un réservoir, à partir d'un débit introduit ou soutiré. Si le débit entrant ou sortant est modifié d'une valeur constante, alors que le niveau était auparavant stabilisé, celui-ci augmentera ou diminuera jusqu'au débordement ou à la vidange du réservoir.

Il est possible, dans la plupart des cas, de représenter ces processus par une relation de type :

$$G(p) = \frac{1}{p} G_1(p) \qquad \text{(I.4)}$$

avec $G_1(p)$ fonction de transfert quelconque du type apériodique. On dit alors

que le processus comporte naturellement un **intégrateur pur.**

I.7- Les caractéristiques du régulateur PID

Le régulateur standard le plus utilisé dans l'industrie est le régulateur PID (proportionnel intégral dérivé), car il permet de régler à l'aide de ses trois paramètres les performances (amortissement, temps de réponse, ...) d'un processus modélisé par un deuxième ordre. Nombreux sont les systèmes physiques qui, même en étant complexes, ont un comportement voisin de celui d'un deuxième ordre. Par conséquent, le régulateur PID est bien adapté à la plupart des processus de type industriel et est relativement robuste par rapport aux variations des paramètres du procédé, quand on n'est pas trop exigeant sur les performances de la boucle fermée par rapport à celles de la boucle ouverte (par exemple, accélération très importante de la réponse ou augmentation très importante de l'amortissement en boucle fermée).

Si la dynamique dominante du système est supérieure à un deuxième ordre, ou si le système contient un retard important ou plusieurs modes oscillants, le régulateur PID n'est plus adéquat et un régulateur plus complexe (avec plus de paramètres) doit être utilisé, au dépend de la sensibilité aux variations des paramètres du procédé.

I.7.1-Les actions PID

En pratique, à une catégorie donnée de systèmes à asservir correspond un type de correcteur adopté. Pour effectuer un choix judicieux, il faut connaître les effets des différentes actions : proportionnelle, intégrale et dérivée.

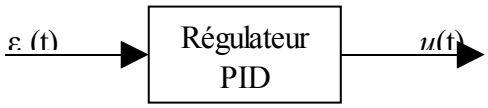


Figure I.9: Schéma synoptique d'un régulateur PID

Un régulateur PID est obtenu par l'association de ces trois actions et il remplit essentiellement les trois fonctions suivantes :

1. Il fournit un signal de commande en tenant compte de l'évolution du signal de sortie par rapport à la consigne
2. Il élimine l'erreur statique grâce au terme intégrateur
3. Il anticipe les variations de la sortie grâce au terme dérivateur.

La commande $U(t)$ donnée par le régulateur PID, dans sa forme

Classique est décrite par :

$$U(t) = K_p \left[\varepsilon(t) + \frac{1}{T_i} \int_0^t \varepsilon(\tau) d\tau + T_d \frac{d\varepsilon(t)}{dt} \right] \quad \text{(I.5)}$$

Elle est la somme de trois termes :

— le terme proportionnel $P = K_p \varepsilon(t)$ (I.6)

— le terme intégral $I = K_p \frac{1}{T_i} \int_0^t \varepsilon(\tau) d\tau$ (I.7)

— le terme dérivatif $D = K_p T_d \frac{d\varepsilon(t)}{dt}$ (I.8)

Les paramètres du régulateur PID sont le gain proportionnel K_p , le temps intégral T_i et le temps dérivatif T_d , les temps étant exprimés en secondes.

I.7.1.1- L'action proportionnelle

La sortie $U(t)$ du régulateur proportionnel est donnée en fonction de son entrée $\varepsilon(t)$ qui représente l'écart entre la consigne et la mesure par la relation :

$$u(t) = K_p \cdot \varepsilon(t) \quad \text{(I.9)}$$

Pour le cas discret, cette relation reste la même telle que :

$$u(k) = K_p \cdot \varepsilon(k) \quad \text{(I.10)}$$

Le rôle de l'action proportionnelle est de minimiser l'écart ε entre la consigne et la mesure et elle réduit le temps de monter et le temps de réponse. On constate qu'une augmentant du gain K_p du régulateur entraîne une diminution de l'erreur statique et permet d'accélérer le comportement global de la boucle fermée. On serait tenté de prendre des valeurs de gain élevées pour accélérer la réponse du procédé mais on est limité par la stabilité de la boucle fermée. En effet, une valeur trop élevée du gain augmente l'instabilité du système et donne lieu à des oscillations.

I.7.1.2- L'action dérivée

Elle est une action qui tient compte de la vitesse de variation de l'écart entre la consigne et la mesure, elle joue aussi un rôle stabilisateur, contrairement à l'action intégrale.

En effet, elle délivre une sortie variant proportionnellement à la vitesse de variation de l'écart ε :

$$u(t) = T_d \frac{d\varepsilon(t)}{dt} \tag{I.11}$$

avec T_d le dosage de l'action dérivée, exprimé en minutes ou en secondes.

L'action dérivée va ainsi intervenir uniquement sur la variation de l'erreur ce qui augmente la rapidité du système (diminution des temps de réponses). L'action dérivée permet aussi d'augmenter la stabilité du système par apport de phase ($+\frac{\pi}{2}$ ce qui augmente la marge de phase). L'annulation de cette action en régime statique impose donc de ne jamais l'utiliser seule : l'action dérivée n'exerce qu'un complément à l'action proportionnelle.

En pratique, il est souhaitable de limiter l'action dérivée afin de ne pas amplifier les bruits haute fréquence et de limiter l'amplitude des impulsions dues aux discontinuités de l'écart. Lorsque la période d'échantillonnage Te est petite, la différence vers l'arrière (Approximation d'Euler rétrograde) nous permet d'approcher la dérivée d'un signal à temps continu par :

$$\frac{df(t)}{dt} = \frac{f(t) - f(t - T_e)}{T_e} \tag{I.12}$$

L'opération de dérivation se traduisait par une multiplication par la variable de Laplace p en continu. Dans le cas discret, en appliquant la transformée en z à l'équation (I.12) on obtient :

$$Z\left\{\frac{f(t) - f(t - T_e)}{T_e}\right\} = \frac{1 - z^{-1}}{T_e} F(z) \tag{I.13}$$

Ceci conduit à établir l'équivalence linéaire entre la variable de Laplace p et la variable z :

$$p \longleftrightarrow \frac{1}{T_e} \cdot \left(1 - z^{-1}\right) \tag{I.14}$$

Donc en discret, le terme dérivé peut être remplacé par :

$$u(k) = \frac{T_d}{T_e} . (\varepsilon(k) - \varepsilon(k-1)) \tag{I.15}$$

I.7.1.3- L'action intégrale

L'action intégrale agit proportionnellement à la surface de l'écart entre la consigne et la mesure, et elle poursuit son action tant que cet écart n'est pas nul. On dit que l'action intégrale donne la précision statique (Elle annule l'erreur statique). L'action intégrale est conditionnée par le temps d'intégrale Ti.

$$u(t) = \frac{1}{T_i}\int_0^t \varepsilon(\tau)\cdot d\tau \qquad \text{(I.16)}$$

Comme dans le cas de l'action proportionnelle, un dosage trop important de l'action intégrale engendre une instabilité de la boucle de régulation. Pour son réglage, il faut là aussi trouver un compromis entre la stabilité et la rapidité.

L'ajout du terme intégral permet d'améliorer la précision mais en contrepartie, il introduit malheureusement un déphasage de $-\frac{\pi}{2}$ ce qui risque de rendre le système instable du fait de la diminution de la marge de phase.

Enfin, le correcteur intégral présente le défaut de saturer facilement si l'écart ne s'annule pas rapidement ce qui est le cas des systèmes lents. En effet, tout actionneur est limité : un moteur est limité en vitesse, une vanne ne peut pas être plus que totalement ouverte ou totalement fermée. Il se peut que la variable de commande amène l'actionneur à sa limite ce qui suspend la boucle de retour et le système aura une configuration assimilable à une boucle ouverte puisque l'actionneur demeurera saturé indépendamment de la sortie du système.

Quand l'erreur est réduite (action intégrale non saturée), il se peut qu'il faille un temps important pour que les valeurs des variables ne soient correctes de nouveau : on appelle ce phénomène l'emballement du terme intégral.

Pour l'éviter, on peut :

- soit suspendre l'action intégrale quand la commande est saturée ;
- soit appliquer une méthode d'anti-saturation, qui consiste à recalculer le terme intégral pour ne pas saturer.

Pour le cas discret, le terme intégral peut être remplacé par la somme des écarts et la différentielle dt par Te ce qui nous donne le résultat suivant :

$$u(n) = \frac{T_e}{T_i}\sum_{k=0}^{n}\varepsilon(k) = u(n-1) + \frac{T_e}{T_i}\varepsilon(n) \qquad \text{(I.17)}$$

Ti : Constante de temps intégrale et s'exprime en (s).

Le problème est que lorsque ε redevient nul, le signal U peut avoir atteint une valeur trop élevée pour qu'il y ait équilibre. Autrement, elle permet d'éliminer l'erreur résiduelle en régime permanent.

L'action intégrale est utilisée lorsqu'on désire avoir en régime permanent, une précision parfaite, en outre, elle permet de filtrer la variable à régler d'ou l'utilité pour le réglage des variables bruités telles que la commande.

I.7.2- Le régulateur PI

Le correcteur intégral est en général associé au correcteur proportionnel, il élabore alors une commande qui peut être donnée par la relation suivante :

$$u(t) = K_p\left(\varepsilon(t) + \frac{1}{T_i}\int_0^t \varepsilon(\tau)d\tau\right) \qquad \text{(I.18)}$$

La fonction de transfert du correcteur est alors donnée par :

$$C(p) = K_p\frac{1+T_i p}{T_i p} \qquad \text{(I.19)}$$

Pour un régulateur intégral pur, le régime dynamique est relativement long. D'un autre côté, le régulateur proportionnel réagit immédiatement aux écarts de réglage mais il n'est pas en mesure de supprimer totalement l'erreur statique. La combinaison des actions proportionnelle et intégrale permet d'associer l'avantage du régulateur P, c'est-à-dire la réaction rapide à un écart de réglage, à l'avantage du régulateur I qui est la compensation exacte de la grandeur pilote.

La transposition de correcteurs continus consiste à discrétiser un correcteur continu afin de l'utiliser dans une commande numérique.

En utilisant cette équivalence dans l'équation (I.11), on obtient le correcteur PI discret :

$$C(z) = \frac{U(z)}{\varepsilon(z)} = \frac{\left(K_p + K_i\right) - K_p \cdot z^{-1}}{1 - z^{-1}} \qquad \text{(I.20)}$$

D'où on obtient l'algorithme de commande du régulateur PI :

$$u(k) = u(k-1) + \left(K_p + K_i\right)\cdot\varepsilon(k) - K_p\cdot\varepsilon(k-1) \qquad \text{(I.21)}$$

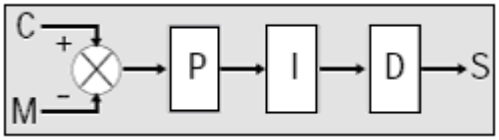
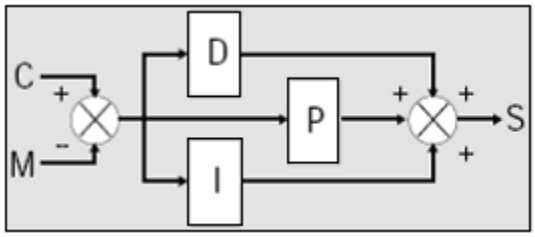
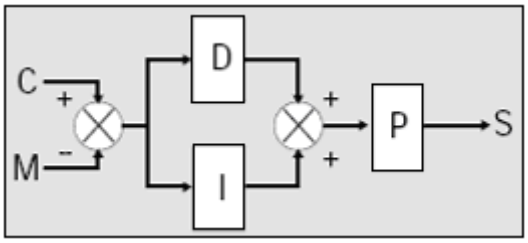
I.7.3. Le régulateur PID

L'action conjuguée PID permet une régulation optimale en associant les avantages de

chaque action : la composante P réagit à l'apparition d'un écart de réglage, la composante D s'oppose aux variations de la grandeur réglée et stabilise la boucle de régulation et la composante I élimine l'erreur statique. Et c'est pour cela que ce type de correcteur est le plus utilisé en milieu industriel.

Dans un régulateur PID, il existe plusieurs façons d'associer les paramètres P, I et D. en effet, le correcteur PID peut avoir une structure série, parallèle ou mixte :

Tableau I.1 : Différentes structures du régulateur PID

Structure du régulateur PID	Schéma et fonction de transfert
Série	 $K_P\left(\frac{T_i+T_d}{T_i}+\frac{1}{pT_i}+pT_d\right)$
Parallèle	 $K_P+\frac{1}{pT_i}+pT_d$
Mixte	 $K_P\left(1+\frac{1}{pT_i}+pT_d\right)$

La discrétisation du correcteur PID parallèle par l'approximation d'Euler rétrograde nous donne :

$$C(z)=\frac{U(z)}{\varepsilon(z)}=K_p+\frac{T_e}{T_i}\cdot\frac{1}{1-z^{-1}}+\frac{T_d}{T_e}\cdot\left(1-z^{-1}\right) \qquad \text{(I.22)}$$

ce qui nous donne l'algorithme de commande suivant :

$$u(k)=u(k-1)+\left(K_p+K_i+K_d\right)\cdot\varepsilon(k)-\left(K_p+2K_d\right)\cdot\varepsilon(k-1)+K_d\cdot\varepsilon(k-2) \qquad \text{(I.23)}$$

avec $K_i = \frac{T_e}{T_i}$ et $K_d = \frac{T_d}{T_e}$

I.8-Le régulateur FLOU

I.8.1-Historique :

Les prémisses de la logique floue sont apparues avant les années 1940, avec les premières approches, par des chercheurs américains, du concept d'incertitude. Il a fallu attendre 1965, pour que le concept de sous ensemble floue soit proposé par L. A. Zadeh, automaticien de réputation internationale, professeur à l'université de Berkeley en Californie, qui a contribué à la modélisation de phénomène sous forme floue, en vue de pallier les limitations dues aux incertitudes des modèles classiques à équation différentielle. En 1974, M. Mamdani expérimentait la théorie énoncée par Zadeh sur une chaudière à vapeur, matériel dont on connaît la complexité, introduisant ainsi la commande floue dans la régulation d'un processus industriel. Plusieurs applications ont alors vu le jour en Europe, pour des systèmes parfois très complexes, telle la régulation de fours de cimenterie réalisée par la société F.L.Smidt-Fuller.

Grâce au chercheur japonais M. Sugene, la logique floue était introduite au Japon dès 1985. Les sociétés japonaises comprirent l'avantage à la fois technique et commercial de la logique floue:

- facilité d'implantation;
- solution de problèmes multivariables complexes;
- robustesse vis à vis des incertitudes;
- possibilité d'intégration du savoir de l'expert

I.8.2- Les sous-ensembles flous

Un sous-ensemble flou F est défini sur un ensemble de valeur, le référentiel U. Il est caractérisé par une fonction d'appartenance :

$$\mu : x \in U \rightarrow \mu(x) \in [0,1] \qquad \text{(I.24)}$$

qui qualifie le degré d'appartenance de chaque élément de U à F.

Exemple : Evaluation de la température de l'eau d'un récipient par les mots

Froide : F Tiède : T Chaude : C

- En logique classique

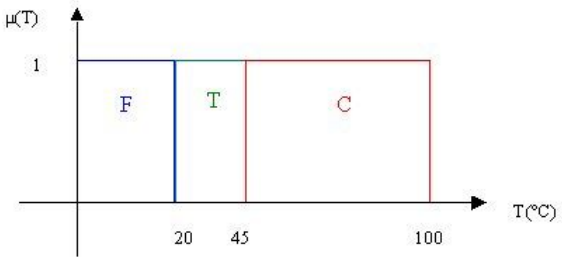


Figure .I.10 : Les fonctions d'appartenances en logique classique

- En logique floue

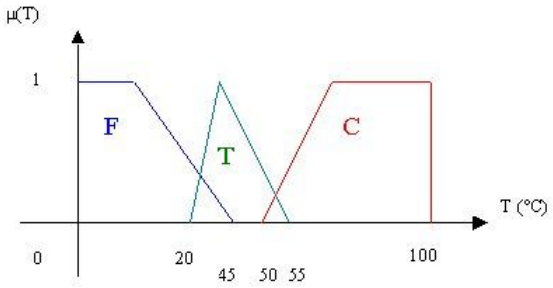


Figure .I.11 : Les fonctions d'appartenances en logique floue

On voit que la logique classique ne peut utiliser que le 0 et le 1 ainsi l'eau est d'abord totalement froide puis tiède et enfin chaude. En dessous nous pouvons observer la représentation graphique de trois fonctions d'appartenance Froid, Tiède et Chaud. Ces fonctions nous permettent de superposer sur des plages de température données les qualificatifs froid et tiède ainsi que tiède et chaud. On se rapproche donc du raisonnement humain.

I.8.3-Les bases de la commande floue

La théorie mathématique sur les sous-ensembles flous définit de nombreuses opérations sur ces sous-ensembles et sur les fonctions d'appartenances qui rendent ces notions utilisables. On ne présente ici que les opérations de base de cette théorie.

Si X, Y et Z sont des sous-ensembles flous et $\mu(x)$, $\mu(y)$ et $\mu(z)$ leur fonction d'appartenance, on définit :

- Le complémentaire de x par la fonction d'appartenance :

$$\mu(z) = 1 - \mu(x) \qquad \text{(I.25)}$$

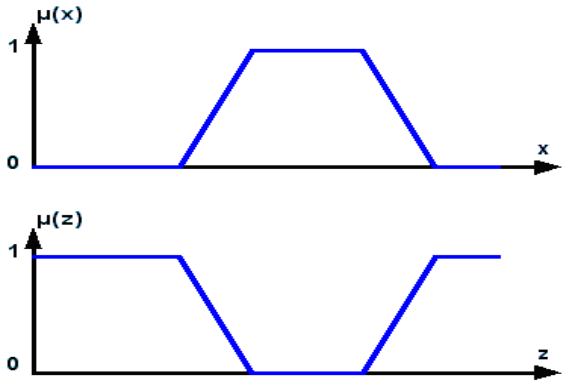


Figure I.12: Opérateur NON

- Le sous-ensemble X et Y, Z=A∩ B, par la fonction d'appartenance :

$$\mu \text{ (z)=min}(\mu(x), \mu(y)) \qquad \text{(I.26)}$$

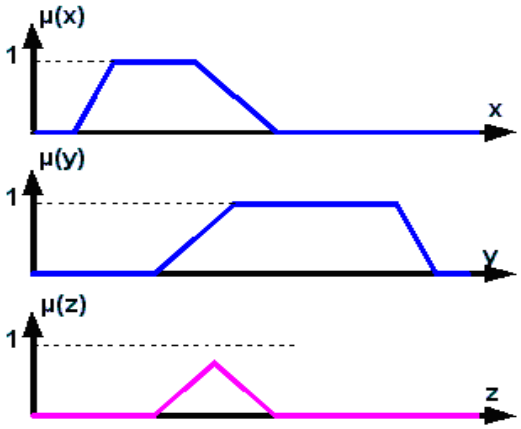


Figure I.13 : Opérateur ET

- Le sous-ensemble X ou Y, Z=A∪ B, par la fonction d'appartenance :

$$\mu \text{ (z)=max}(\mu(x), \mu(y)) \qquad \text{(I.27)}$$

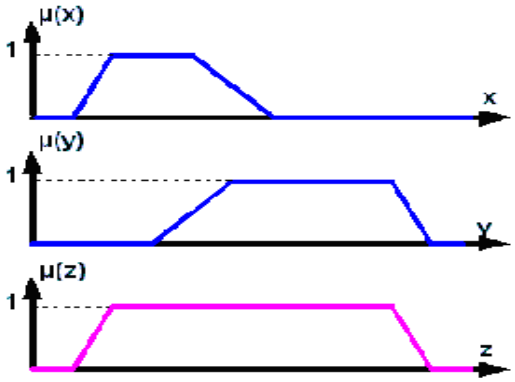


Figure I.14 : Opérateur OU

Ces définitions sont celles les plus communément utilisées mais parfois, pour certains cas, d'autres sont plus appropriées. Par exemple l'intersection peut être définie par le produit des fonctions d'appartenance et l'union par la moyenne arithmétique des fonctions d'appartenance. Ces différentes techniques de calcul engendrent une énorme capacité d'adaptation des raisonnements flous.

I.8.4- Structure d'une commande floue

La structure conventionnelle d'une commande floue est présentée par figure (II.7). Elle est composée de quatre blocs distincts dont les définitions son données ci-dessous.

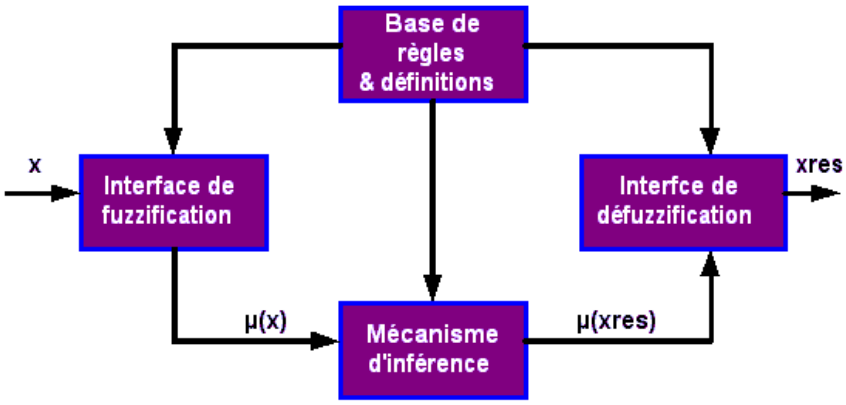


Figure I.15 : Structure d'une commande floue

Avec x représente le vecteur des entrées (variable d'entrée réelles), x_{RES} celui des commandes (variable de sortie réelle), μ(x) et μ(x_{RES}) les fonctions d'appartenances correspondantes (variable d'entrée floue et variable de sortie floue).

On procède tout d'abord à la partition en sous-ensembles flous des différents univers de discours que le système impose. Ensuite on détermine la base de règles qui va caractériser le fonctionnement désiré du système.

Puis il faut transformer les variables réelles, c'est à dire celles qui ont une réalité physique, en variables floues. On appelle cette étape la fuzzification On utilise ensuite ces variables floues dans un mécanisme d'inférence qui crée et détermine les variables floues de sortie en utilisant les opérations sur les fonctions d'appartenance.

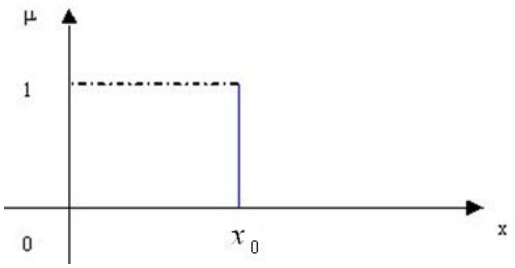
Enfin, on opère à la défuzzification qui consiste à extraire une valeur réelle de sortie à partir de la fonction d'appartenance du sous-ensemble flou de sortie établi par le mécanisme d'inférence.

I.8.5- Quelques techniques de fuzzification et défuzzification

Il existe de très nombreuses techniques de fuzzification et défuzzification. On ne présente ici que les plus basiques pour rendre un tant soit peu concret ces opérations qui constituent une des parts les plus importantes de la réalisation d'une commande floue.

- Technique de fuzzification :

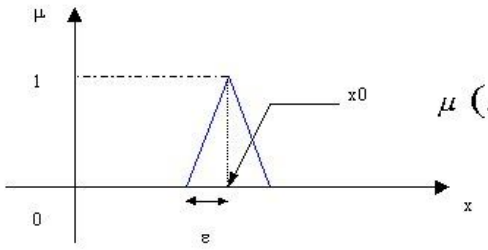
L'exemple trivial est la fuzzification d'une valeur exacte x_0. Le sous-ensemble flou lié à cette variable est alors caractérisé par la fonction d'appartenance suivante :



$$\mu : x \in U \setminus \{x_0\} \rightarrow 0$$
$$x_0 \rightarrow 1$$

Figure I.16 : Méthode de fuzzification pour une mesure exacte

L'autre technique de base est la fuzzification dune valeur x_0 entachée d'une incertitude ε . La fonction d'appartenance est alors :



$$\mu(x) = \max \left\{ 0 ; 1 - \frac{|x - x_0|}{\varepsilon} \right\}$$

Figure I.17 : Méthode de fuzzification pour une mesure incertaine

- Technique de défuzzification :

Le but de la défuzzification est d'extraire une valeur réelle y_0 à partir de la fonction d'appartenance $\mu(y)$ du sous-ensemble de sortie.

- Défuzzification par centre de gravité :

La méthode de défuzzification la plus utilisée est celle de la détermination du centre de gravité de la fonction d'appartenance résultante $\mu_{RES}(z)$. Dans ce contexte, il suffit de calculer l'abscisse z^*. La figure I.18 montre le principe de défuzzification.

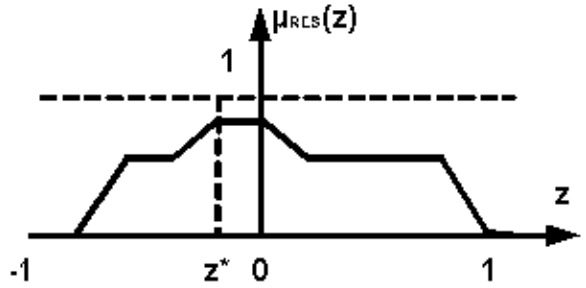
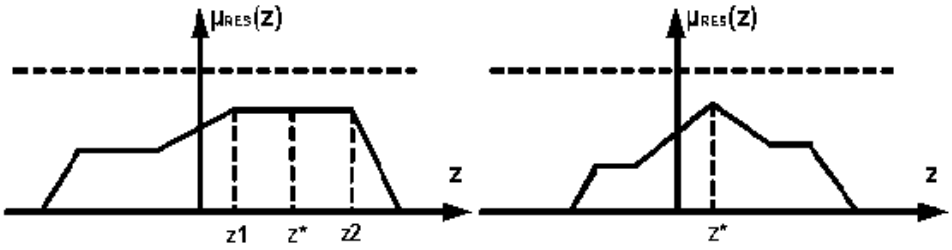


Figure I.18 : Défuzzification par centre de gravité

- ✓ Défuzzification par valeur maximale :

La défuzzification par centre de gravité exige en général une envergure de calcul assez importante. Par conséquent, il sera utile de disposer d'une méthode de défuzzification plus simple.

Comme signal de sortie z^*, on choisit l'abscisse de la valeur maximale de la fonction d'appartenance résultante μ_{RES} (z) comme le montre la figure I.19 (b).



(a) (b)

Figure I.19 : Défuzzification par valeur maximum.

Lorsque μ_{RES} (z) est écrêté, toute valeur entre z_1 et z_2 peut être utilisée comme l'indique la figure I.19 (a). Afin d'éviter cette indétermination, on prend la moyenne des abscisses du maximum. Cependant cette méthode présente un grand inconvénient : le signal de sortie z saute si la dominance change d'une fonction partielle à une autre.

I.8.6-. Les régulateurs flous de type Mandani

Dans les modèles linguistiques, appelés aussi modèle flous de type Mamdani, les antécédents et les conséquences des règles sont des propositions floues. La forme générale des règles de Mamdani est :

Un système flou de type Mandani est basé sur une collection de règles du type :

$$R^i : Si\ x_1\ est\ F_1^i\ et\ x_n\ est\ F_n^i\ ALORS\ U = h^i \qquad \text{(I.28)}$$

Où les x_i représentent les variables d'entrées du régulateurs et U est la variable de sortie du régulateurs, F_j^i étant les sous-ensembles flous.

Généralement les régulateurs flous de type Mandani, sont des régulateurs à deux entrées, l'erreur et sa variation. Et une sortie, qui représente la variation de la commande, régie par des règles de la forme suivante :

$$R^i : Si\ e\ est\ F_1^i\ et\ \Delta e\ est\ F_2^i\ ALORS\ \Delta U = h^i \qquad \text{(I.29)}$$

La structure du processus ainsi commandé est donnée par la figure (I.22) qui correspond à un régulateur PI-flou.

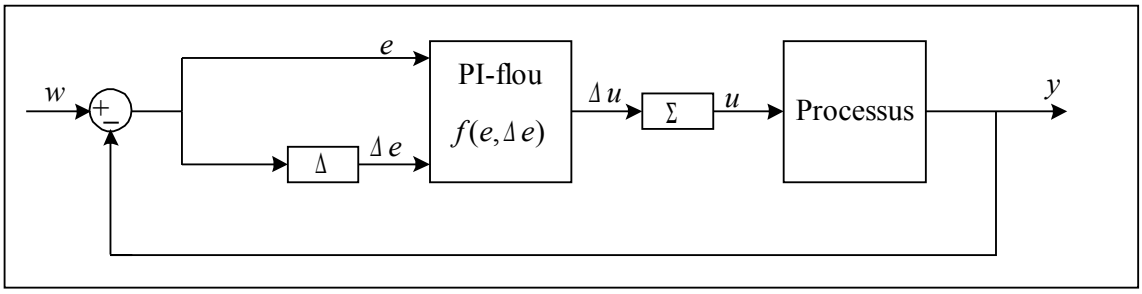


Figure I.20 : Structure d'un PI flou de type Mamdani

La loi de commande du régulateur PD-flou, est obtenu en suppriment l'intégration à la sortie. Les règles ont la forme suivante :

$$R^i : Si\ e\ est\ F_1^i\ et\ \Delta e\ est\ F_2^i\ ALORS\ U = h^i \qquad \text{(I.30)}$$

Pour le régulateur PID-floue, la loi de commande est obtenue grâce à l'erreur, sa variation et sa variation seconde..Les règles ont la forme suivante :

$$R^i : Si\ e\ est\ F_1^i\ et\ \Delta e\ est\ F_2^i\ et\ \Delta 2e\ est\ F_3^i\ ALORS\ \Delta U = h^i \qquad \text{(I.31)}$$

La défuzzification est généralement effectuée par la méthode du centre de gravité. Une variante de la méthode de Mamdani consiste à remplacé l'opérateur *min* de l'influence floue par l'opérateur *prod* (produit)

I.8.7- Les régulateurs flous de type Sugeno

Le modèle de Sugeno est connu sous le nom de TSK, car il a été proposé par Takagi, Sugeno et Kang en 1988.ce modèle a pour but de donner à la sortie de la règle des valeurs concrètes, et non pas une valeur floue, qui nécessite une étape de défuzzification.

Le système flou de type Takagi-Sugeno, utilise des règles écrites de la manière suivante :

$$R^i : Si\ x_1\ est\ F_1^i\ est \ldots\ldots x_n est\ F_n^i\ ALORS\ U = h^i(X) \qquad (I.32)$$

Chacune de ces règles représente un modèle local sur une région floue d'entrée, ou sur un sous-espace d'entrée. Dans chaque région, le modèle flou est défini par la fonction h^i qui relie les entrées à la sortie numérique. Le modèle global est constitué par l'interpolation des modèles locaux.

I.8.8-conclusion

Dans ce chapitre, nous avons présenté les différents méthodes de régulation en boucle ouverte et en boucle fermé,, une description générale des systèmes industriel ainsi que quelque méthode de régulation, qui se résument dans l'utilisation des régulateurs PID Enfin, nous avons touché la logique floue à travers les régulateurs de type Mandani et de Sugeno.

Nous présentons dans le chapitre suivant la maquette dont on doit réguler le niveau d'eau.

Chapitre II : Description du système de régulation de niveau d'eau

II : Description du système de régulation de niveau d'eau

II.1. Introduction

On se propose dans ce chapitre de présenter le système de régulation du niveau d'eau dans un réservoir. Ceci permettra de mettre en évidence les différents composants d'une boucle de régulation et de comprendre les actions PID du régulateur pour atteindre des performances désirées.

II.2. Présentation de la maquette

La maquette de régulation est constituée d'un réservoir haut de remplissage d'eau, d'un réservoir bas (source d'eau), d'un capteur de pression différentielle, d'une motopompe et de deux cartes électroniques l'une d'acquisition ou de mesure et l'autre est de puissance destinée à la commande. La régulation est assurée par un programme construit avec le logiciel LABVIEW, qu'on présentera après , la régulation est faite avec un PID industriel .

Le schéma synoptique de la structure de la maquette conçue est donné par la figure

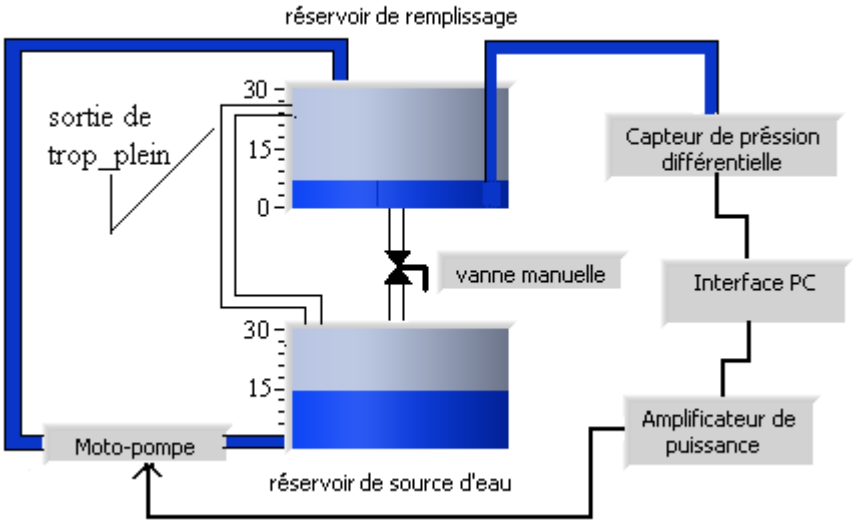


Figure II.1 : Structure de la maquette

- Le réservoir haut est en plexiglas transparent permettant ainsi l'observation de l'évolution du niveau d'eau. De plus le Plexiglas présente beaucoup moins de dangers que le verre en cas de bris. Le réservoir haut est de forme parallélépipédique, de 30 cm de hauteur et de largeur, et de 4 cm de profondeur, une règle graduée permet de lire directement le niveau d'eau. Ce réservoir présente une entrée de remplissage d'eau, une sortie d'évacuation et une sortie de trop-plein.

- Le réservoir bas représente la source d'eau, il est naturellement plus volumineux que le réservoir haut. Il est aussi de forme parallélépipédique, de 30 cm de hauteur et de largeur, et de 18 cm de profondeur. Il a une capacité de remplissage de 5 litres. Ce réservoir est aussi en plexiglas mais opaque et non transparent.

- La motopompe permet le remplissage du réservoir haut, elle est composée d'une entrée d'aspiration de 10 mm de diamètre, d'une sortie de refoulement de 3 mm de diamètre, d'une pompe hermétique et d'un cordon d'alimentation. La motopompe est alimentée par une tension continue variable de 0 à 12 V, elle consomme en fonctionnement nominal un courant de 2.6 A.

- Le capteur, référencé **HONEYWELL 26PCAFA6D** (voir Annexe A)**,** mesure la pression du fond du réservoir haut par rapport à la pression atmosphérique. Il est alimenté par 12 V et délivre une sortie variant de 0 à 17 mV. Pour une colonne d'eau de 70 cm, le capteur délivre une tension de mesure égale à 17 mV. La sortie du capteur étant en millivolts, la réalisation d'un amplificateur d'instrumentation s'avère donc indispensable pour permettre au régulateur de lire l'image du niveau d'eau. Un tube en verre relié à une conduite en plastique permet de transmettre hermétiquement la pression du fond du réservoir au capteur.

- La vanne manuelle placée sur le retour d'évacuation permet de varier le débit de sortie du réservoir haut et d'introduire des perturbations en fonctionnement statique.

- Le trop-plein a un diamètre suffisamment grand pour garantir deux fonctions : prévenir le débordement du réservoir haut, en cas de disfonctionnement du système ou en cas d'un dépassement important, et maintenir le réservoir bas à la pression atmosphérique empêchant ainsi la création d'une dépression lorsque la pompe aspire de l'eau.

II.3. Carte de mesure

Pour pouvoir réguler le processus, le régulateur doit avoir une tension image bien adaptée. En effet, le capteur délivre une sortie 0-17 mV et le microcontrôleur a une pleine échelle égale à 5 V. Le capteur fait correspondre une tension nulle à une

hauteur nulle et une tension de 17 mV à une hauteur de 70 cm or la hauteur maximale de la colonne d'eau dans le réservoir ne dépasse pas les 28 cm ce qui nous impose d'en tenir compte.

Pour une colonne d'eau de hauteur égale à 28 cm, on a relevé une tension de sortie du capteur valant 5,6 mV d'où on peut calculer le gain d'amplification :

$$G = \frac{5}{5.6\ 10^{-3}} = 893$$

II.3.1. Montage de l'amplificateur d'instrumentation

Le schéma de la figure II.2 représente le montage de l'amplificateur d'instrumentation, constituant le conditionneur du capteur de pression différentielle. Les entrées $e+$ et $e-$ représentent respectivement la sortie (+) et la sortie (−) du capteur.

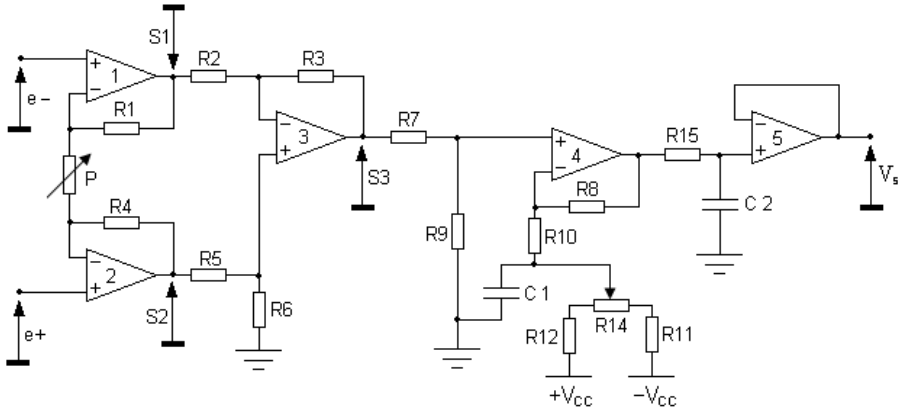


Figure II.2 : Schéma du conditionneur

Le bloc formé par les amplificateurs 1, 2 et 3 représente le conditionneur. Le potentiomètre P permet d'avoir un gain variable pour bien étalonner la mesure.

Le bloc formé par l'amplificateur 4 et le montage potentiométrique permet un réglage de l'offset. En effet, une tension de décalage apparaît à la sortie de l'amplificateur 3 ce qui fausse la lecture du niveau d'eau. Grâce à ce montage soustracteur, on parvient à éliminer la tension de décalage.

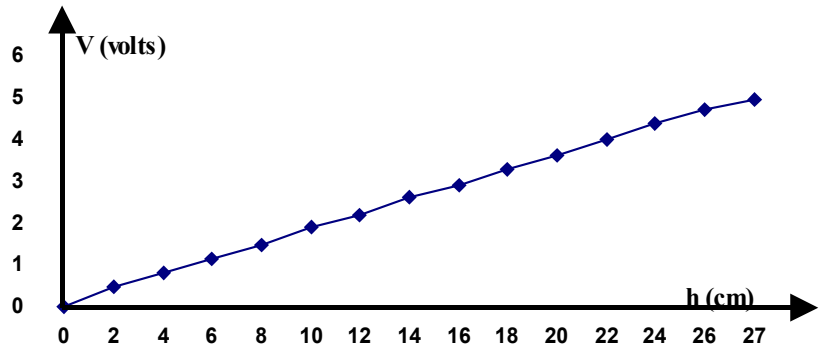
Le dernier bloc du montage représente un filtre passe bas qui contribue au lissage de la mesure par sa fréquence de coupure qui a été choisie égale à 100Hz .

II. 3.2- Etalonnage du capteur

L'étalonnage nous permet de se renseigner sur la sensibilité du capteur et sur la linéarité de sa réponse ou non par rapport à la variation de la pression. En variant le niveau d'eau, on a relevé les valeurs suivantes de la tension de sortie du capteur :

Tableau II.1 : Variation de la mesure en fonction du niveau

h(cm)	0	2	4	6	8	10	12	14	16	18	20	22	24	26	27
V(volts)	0,05	0,5	0,8	1,16	1,5	1,9	2,2	2,6	2,9	3,3	3,6	4	4,4	4,7	4,94



La courbe d'étalonnage est linéaire, on peut donc varier la commande proportionnellement au niveau d'eau choisit sans avoir à compenser la mesure délivrée par le capteur.

II.4. Carte de commande

Il est impossible de commander directement un moteur de courant nominal dépassant les deux ampères. L'amplificateur de puissance doit donc délivrer la puissance nécessaire au fonctionnement du moteur. La commande délivrée par le régulateur est une commande en tension ce qui veut dire que l'amplificateur doit avoir un gain unitaire en tension et un gain en courant permettant l'alimentation du moteur sans surcharge excessive.

Le moteur arrive à pomper de l'eau dans le réservoir haut à partir d'une tension de commande égale à 4,2 V, il est donc nécessaire de le commander par une tension dépassant les 5 volts pour pouvoir aisément remplir le réservoir.

Pour avoir le bon fonctionnement du montage, on applique une tension de 8V.

II.4.1. Montage de l'amplificateur de puissance :

Le montage de l'amplificateur de puissance utilisé est donné par le schéma de la figure II.4 :

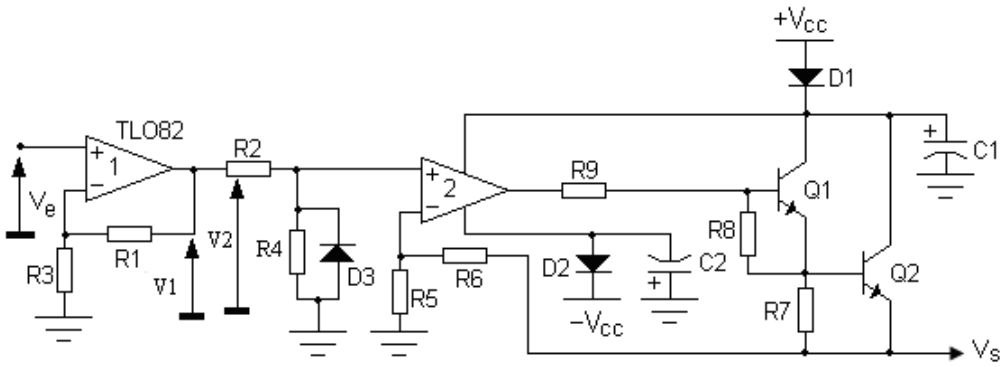


Figure II.3 : Schéma de l'amplificateur de puissance

II.4.2. Fonctionnement du montage

- Les diodes D1 et D2 sont des diodes de protection contre l'inversion des tensions d'alimentation.
- Les capacités C1 et C2 sont des capacités de filtrage.
- Soient V_1 la tension de sortie de l'amplificateur 1 et V_2 la tension d'entrée non inverseuse de l'amplificateur 2, on a :

$$V_1 = \left(1 + \frac{R_1}{R_3}\right) V_e \quad \text{et} \quad V_2 = \frac{R_4}{R_2 + R_4} V_1 = \frac{R_4}{R_2 + R_3}\left(1 + \frac{R_1}{R_3}\right) V_e$$

On doit tout d'abord avoir un gain en tension égal à 1.6 au niveau du premier amplificateur pour avoir une tension de commande égale à 8 V au lieu de 5 V, il faut donc avoir $R_1 = 0.6\,R_3$. En prenant $R_1 = 9.1\ k\Omega$ et $R_3 = 15\ k\Omega$, on obtient un rapport égal à 0.606.

On a aussi :

$$V_2 = \frac{R_5}{R_5 + R_6} V_S \text{ ou bien } V_S = \left(1 + \frac{R_6}{R_5}\right) \cdot V_2$$

Donc :

$$V_S = \left(1 + \frac{R_6}{R_5}\right) \cdot \left(\frac{R_4}{R_2 + R_4}\right) \cdot \left(1 + \frac{R_1}{R_3}\right) \cdot V_e$$

En choisissant $R_2 = R_4 = R_5 = R_6 = 10\ k\Omega$, on obtient

$$V_S = 1.6\, V_e$$

Pour une tension de commande égale à 5 V, le moteur sera alimenté par 8 V.

II.5-conclusion

Dans ce chapitre on a présenté la description électronique de la maquette de régulation de niveau d'eau, la carte de mesure et la carte de commande. On présentera ensuite dans le chapitre suivant la structure de la carte d'interfaçage réalisé et les principes de conversion A/N et N/A et le protocole de la transmission série

Chapitre III : Développement d'une carte d'interfaçage à base de PIC16F877

III. Développement d'une carte d'interfaçage à base de PIC16F877

III.1-principe d'interfaçage

L'interfaçage permet l'échange d'informations entre deux ou plusieurs périphériques, Pour notre cas on s'intéresse à l'interfaçage entre un PC et un réservoir remplit d'eau, pour cela on a réalisé une carte d'interfaçage à base de PIC 16F877.

III.2-descriptions des composants de la carte de commande réalisée

Pour pouvoir commander le procédé (dans notre cas c'est un niveau d'eau dans un réservoir) avec une loi de commande numérique à travers un PC, il est nécessaire d'utiliser une carte d'interfaçage qui a pour but de transformer les signaux numérique à des signaux analogique équivalente et les signaux analogiques en des signaux numérique équivalente.

La carte réalisée contient :

- Un PIC16F877
- Un DAC0808
- Un AOP LF351
- MAX232
- Port série

Nous décrivons si dessous les caractéristiques de ces composant un par un :

III.2.1-Microcontrôleur PIC 16F877

III.2.1.1-Définition d'un PIC

Un PIC est un microcontrôleur, c'est à dire une unité de traitement de l'information de type microprocesseur à laquelle on a ajouté des périphériques internes permettant de réaliser des montages sans nécessiter l'ajout de composants externes.les PICs sont des composant dits RISC (Reduced Instructions Set Computer), ou encore (composant à jeu d'instruction réduit).

III.2.1.2-Les différentes familles des PICs

La famille des pics est subdivisée à l'heure actuelle en 3 grandes familles :

- Base-line : c'est une famille qui utilise des mots d'instructions de 12 bits.

- Mid-range : c'est une famille qui utilise des mots de 14 bits(dont font partie les 16F84,16f876et 16f877

- High-end : c'est une famille qui utilise des mots de 16 bits.

III.2.1.3-Identification d'un Pic

Pour identifier un PIC, on utilise simplement son numéro :

16 : indique la catégorie du PIC, c'est un Mid-range.

L : indique qu'il fonctionne avec une plage de tension beaucoup plus tolérante.

C : indique que la mémoire programme est un EPROM ou une EEPROM.

CR ou F : indique le type de mémoire ; CR(ROM) ou F (FLASH).

XX : représente la fréquence d'horloge maximale que le PIC peut recevoir.

Une dernière indication qu'on le trouve est le type de boîtier.

Exemple :

Un 16f877-20 est un PIC MID-RANGE(16) ou la mémoire programme est de type FLASH (F) et réinscriptible de type 877 et capable d 'accepter une fréquence d'horloge de 20MHz

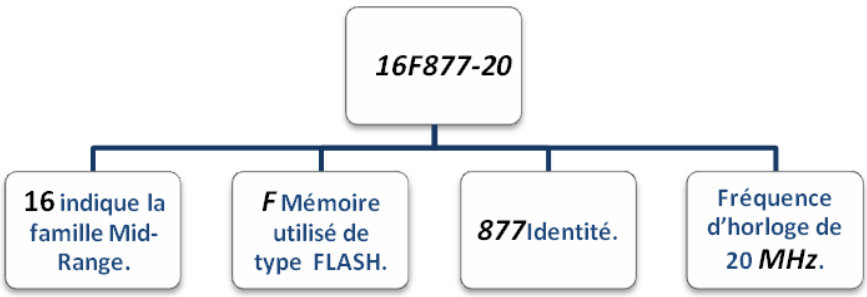


Figure III.1: description de la configuration du PIC 16F877

III.2.1.4-Les caractéristiques principales d'un microcontrôleur

On cite :

- De nombreux périphériques d'E/S
- Une mémoire de programme
- Une mémoire vive (en général de type SRAM)
- Eventuellement une mémoire EEPROM destinée à la sauvegarde par programme de données à la coupure de l'alimentation
- Un processeur 8 ou 16 bits
- Faible consommation électrique

Les tailles mémoire sont en général réduites, de l'ordre de :

- 16 ko pour la mémoire programme
- Quelques octets à 16 ko pour la RAM

La puissance de calcul est aussi limitée :

- 0.012 MIPS pour les Basic Stamp
- 1 à 5 MIPS pour les PIC

III.2.1.5-Caractéristique générale de la famille 16F87x

La famille 16F87x comprend toute une série de composants, voici les différents types existants et qui sont entrain d'évolution :

Tableau .III-1: Evolution de la famille 16F87x

PIC	FLASH	RAM	EEPROM	I /O	A /D	Port //	Port série
16F870	2K*14	128	64	22	5	NON	USART
16F871	2K*14	128	64	33	8	PSP	USART
16F872	2K*14	128	64	22	5	NON	MSSP
16F873	4K*14	192	128	22	5	NON	USART/MSSP
16F874	4K*14	192	128	33	8	PSP	USART/ MSSP
16F876	8K*14	368	256	22	5	NON	USART/ MSSP
16F877	8K*14	368	256	33	8	PSP	USART/ MSSP

Tous ces composants sont identiques, aux exceptions citées dans le tableau précédent. Les différences fondamentales entre ces PICs sont donc les quantités de mémoires disponibles, le nombre d'entrées/sorties, le nombre de convertisseurs de type « analogique/numérique », et le nombre et le type des ports intégrés.

Tous les PICs Mid-Range, dont il appartient notre PIC 16F877, ont un jeu de 35 instructions, stockent chaque instruction dans un seul mot de programme, et exécutent chaque instruction (sauf les sauts) en 1 cycle ; on atteint donc des très grandes vitesses ; et les instructions sont de plus très rapidement assimilées.

L'horloge fournie au PIC est prédivisée par 4 au niveau de celle-ci ; c'est cette base de temps qui donne le temps d'un cycle : si on utilise par exemple un quartz de 4MHz, on obtient donc 1000000 de cycles/seconde, or, comme le PIC exécute pratiquement 1 instruction par cycle, hormis les sauts, cela nous donne une puissance de l'ordre de 1 MIPS (1 Million d'Instructions Par Seconde).

III.2.1.6-Les principales caractéristiques du PIC 16F877

Le PIC 16F877 est caractérisé par :

- Une Fréquence de fonctionnement élevée, jusqu'à 20 MHz
- Une mémoire vive de 368 octets.
- Une mémoire EEPROM pour sauver des paramètres de 256 octets
- Une mémoire morte de type FLASH de 8 Kmots (1mot = 14 bits), elle est réinscriptible à volonté
- Chien de garde WDT
- 33 Entrées et sorties
- Chaque sortie peut sortir un courant maximum de 25 mA
- 3 Temporisateurs : TIMER0 (8 bits avec pré diviseur), TIMER1 (16 bits avec prédiviseur avec possibilité d'utiliser une horloge externe réseau RC ou QUARTZ) et TIMER2 (8 bits avec prédiviseur et postdiviseur)
- 2 entrées de captures et de comparaison avec PWM (Modulation de largeur d'impulsions)
- Convertisseur analogique numérique 10 bits avec 8 entrées multiplexées maximum
- Une interface de communication série asynchrone et synchrone (USART/SCI)
- Une interface de communication série synchrone (SSP/SPI et I2 C)
- Une tension d'alimentation entre 2 et 5.5 V

III.2.1.7-Les mémoires du PIC 16F877

Le PIC 16F877 dispose de trois types de mémoires :

- Mémoire vive RAM

C'est de la mémoire d'accès rapide, mais labile (c'est-à-dire qu'elle s'efface lorsqu'elle n'est plus sous tension); cette mémoire contient les registres de configuration du PIC ainsi que les différents registres de données. Elle contient également les variables utilisées par le programme.

Cette mémoire RAM disponible sur le 16F877 est de 368 octets, elle est répartie de la manière suivante:

- 80 octets en banque 0, adresses 0x20 à 0x6F
- 80 octets en banque 1, adresses 0xA0 à 0xEF
- 96 octets en banque 2, adresses 0x110 à 0x16F
- 96 octets en banque 3, adresses 0x190 à 0x1EF

- 16 octets communs aux 4 banques, soit 0x70 à 0x7F; 0xF0 à 0xFF; 0x170 à 0x17F; 0x1F0 à 0x1FF

❖ Mémoire morte FLASH

C'est la mémoire programme proprement dite. Chaque case mémoire unitaire fait 14 bits. La mémoire FLASH est un type de mémoire stable, réinscriptible à volonté.

C'est ce nouveau type de mémoire qui a fait le succès de microprocesseur PIC. Dans le cas du 16F877, cette mémoire FLASH fait 8 K. Lorsque l'on programme en assembleur, on écrit le programme directement dans cette mémoire.

❖ Mémoire EEPROM

Cette mémoire est de 256 octets, elle est Electriquement effaçable, réinscriptible et stable. Ce type de mémoire est d'accès plus lent, elle est utilisée pour sauver des paramètres.

L'adresse relative de l'accès EEPROM est comprise entre 0000et 00ff, ce qui nous permet d'utiliser qu'un registre de huit bits pour définir cette adresse.

III.2.1.8-Organisation externe du PIC 16F877

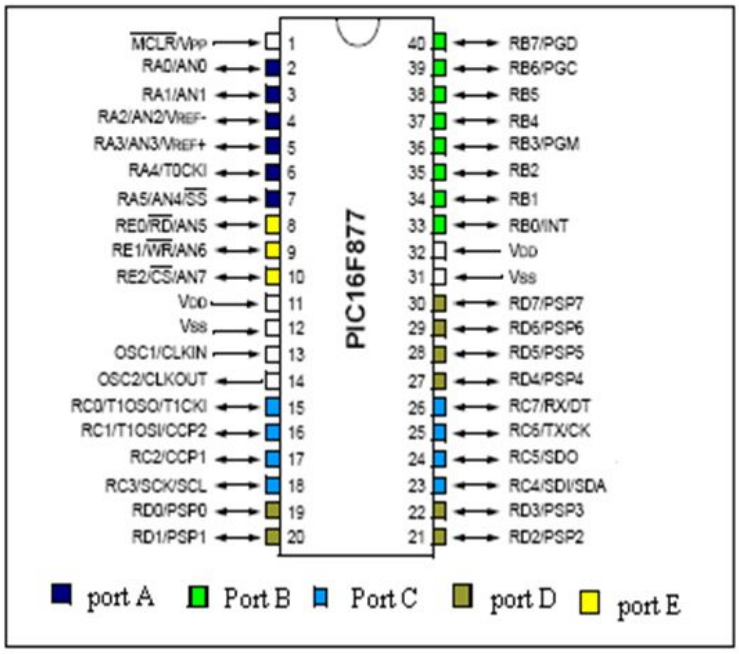


Figure III.2 : Brochage du PIC 16F877

Le boîtier du PIC 16F877 décrit par la figure III.2 comprend 40 pins : 33 pins d'entrées/sorties, 4 pins pour l'alimentation, 2 pins pour l'oscillateur et une pin pour le reset (MCLR).

La broche MCLR sert à initialiser le µC en cas de la mise sous tension, de remise à

zéro externe, de chien de garde et en cas de la baisse de tension d'alimentation.
Les broches VDD et VSS servent à alimenter le PIC.
On remarque qu'on a 2 connections « VDD » et 2 connections « VDD ».
La présence de ces 2 pins s'explique pour une raison de dissipation thermique. Les courants véhiculés dans le pic sont loin d'être négligeables du fait des nombreuses entrées/sorties disponibles.
Le constructeur a donc décidé de répartir les courants en plaçant 2 pins pour l'alimentation VSS, bien évidemment, pour les mêmes raisons, ces pins sont situées de part et d'autre du PIC, et en positions relativement centrales.
Les broches OSC1 et OSC2 ou CLKIN et CLOUT permettent de faire fonctionner l'oscillateur interne du PIC qui peut être un quartz, un résonateur céramique, un oscillateur externe ou un réseau RC. dont le rôle est de crées des impulsions de fréquences élevées.
Lors de la programmation, la broche MCLR doit être portée à un niveau compris entre 12 V et 14 V et le PIC16F877 commence à programmer en appliquant un signal d'horloge sur la broche RB6 (broche 39) et les informations binaires transitent en série sur la broche RB7 (broche 40). Chacune des informations qui transitent sur la broche RB7 est validée à la retombée du signal d'horloge sur la broche RB6 .

III.2.1.9 Les ports d'entrée/sortie

Les µc 16F877 contient les 5 ports suivants

- Port A : 6 pin I/O numérotées de RA0 à RA5
- Port B : 8 pin I/O numérotées de RB0 à RB7
- Port C : 8 pin I/O numérotées de RC0 à RC7
- Port D : 8 pin I/O numérotées de RD0 à RD7
- Port E : 3 pin I/O numérotées de RE0 à RE2

Tous ces ports se trouvent dans la banque 0, mais tous leurs registres se trouvent dans la banque1, pour déterminer les modes des ports (I/O), il faut sélectionner leurs registres TRISX :

- ❖ le positionnement d'un bit à « 1 » place la pin en entrée.
- ❖ Le positionnement de ce bit à « 0 » place la pin en sortie.

Notez, comme pour tous les ports, que la mise sous tension du PIC, et tout autre reset, force tous les bits utiles de TRISx à 1, ce qui place toutes les pins en entrée. En plus pour configurer TRISX avec notre compilateur il faut agir sur la valeur set_tris_x 0 ; Exemple : Set_tris_x (valeur).

- Le port A

Le port A est formé de six pins donc six entrées/sorties numérotées de RA0 à RA5 qui peuvent être utilisé comme des entrées pour le convertisseur analogique numérique ou utilisé pour le TIMER 0, dans ce dernier cas le pin RA4 sera utilisé comme entrée pour configurer TOCKI est de type drain ouvert.
On peut utiliser ce port, soit pour la conversion analogique /numérique, soit en mode (I/O), dans notre projet on a utilisé RA0 comme entrée pour le CAN.
Remarque : RA4 qui est toujours en collecteur ouvert (mise à 0) c'est à dire mise en sortie.

- Le port B

Rien de particulier à dire sur ce port, qui possède 8 pins d'entrée/sortie classique numérotées de RB0 à RB7.
On note que la pin RB0 qui, en configuration d'entrée, est de type « trigger de Schmitt » quand elle est utilisée en mode interruption « INT ». La lecture simple de RB0 se fait, de façon tout a fait classique, en entrée de type TTL.

- Le port C

Tout d'abord au niveau programmation, c'est un PORT tout ce qu'il y a de plus classique, comportant 8 pins de RC0 à RC7. On trouve donc un registre TRISC localisé dans la banque 1, qui permet de décider quelles sont les entrées et quelles sont les sorties. Le fonctionnement est identique à celui des autres TRIS, à savoir que le positionnement d'un bit à « 1 » place la pin en entrée, et que le positionnement de ce bit à « 0 » place la dite pin en sortie.
Au niveau électronique, on remarque que toutes les pins, lorsqu'elles sont configurées en entrée, sont des entrées de type « trigger de Schmitt » ; ce qui permet d'éviter les incertitudes de niveau sur les niveaux qui ne sont ni des 0V, ni des +5V, donc, en général, sur les signaux qui varient lentement d'un niveau à l'autre.

❖ Le port D :

Ce port n'est présent que sur les PIC16F877. il fonctionne de façon identique aux autre, dans son mode de fonctionnement générale .les 8 pin I/O, en mode entrée , sont de type « trigger de Schmitt »

Ce port est très utilisé en mode parallèle esclave (slave).

❖ Le port E :

Ce port n'est présent que sur les PICs de type 16F877.

Le port E possède trois pins donc trois entrées/sorties, RE0 à RE2, il est utilisé comme entrées au convertisseur analogique numérique et aussi il peut contrôler le port parallèle slave c'est-à-dire le port D.

On remarque que les pins REx peuvent également être utilisées comme pins d'entrées analogiques. C'est de nouveau le registre ADCON1 qui détermine si ce port sera utilisé comme port I/O ou comme port analogique.

III.2.1.10-L' interruption RB0/INT

Cette broche a une double fonction elle peut être utilisée comme une broche standard RBO ou comme une entrée d'interruption INT.

Si cette broche est utilisée comme une entrée d'interruption externe, elle doit être maintenue à un niveau haut par l'intermédiaire de résistances de 10 kΩ pour ne pas déclencher d'interruptions imprévues, cela permet aussi de relier plusieurs sources d'interruptions sur une même ligne.

III.2.1.11-Le convertisseur analogique numérique du PIC 16F877

La fonction conversion analogique-numérique consiste à transformer une grandeur électrique en une grandeur numérique exprimée sur N bits. Cette grandeur de sortie représente, dans le système de codage qui lui est affecté, un nombre proportionnel à la grandeur analogique d'entrée.

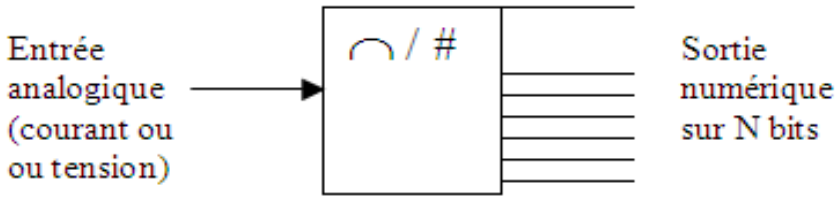


Figure III.3 : Principe de conversion analogique / numérique

Le CAN intégré dans le 16F877 est un CAN 10 bits qui donne une précision en 5V de 5mv environ, ce qui est une précision tout a fait intéressante. La résolution 10 bits du PIC permet d'attribuer 1024 valeurs numérique a notre signal d'entré.

D'une manière générale, le principe de conversion analogique/numérique nécessite deux opérations :

A- La quantification : opération qui consiste à associer une valeur analogique à la plus petite variation mesurable entre deux valeurs codées distinctes en sortie. Cette valeur est appelée **quantum.**

-

$$q = \frac{\Delta Ve\max}{2^n} = \frac{V_{ref+} - V_{ref-}}{2^n}$$

1 - q : quantum (V), aussi appelé résolution

2 - ΔVe_{MAX} : c'est l 'écart entre la valeur mini et maxi de Ve à numériser (V)

3 - n : nombre de bits en sortie du convertisseur

B- Le codage : opération qui assigne une valeur numérique à chacun de ces niveaux. Les codages les plus couramment utilisés sont :

- Le binaire naturel, pour les nombres non signés,
- Le complément à 2 pour les nombres signés,
- Le code binaire signé.

Il existe différent méthodes de conversion analogique numérique , tel que :

- La conversion a simple rampe
- La conversion à double rampe
- La conversion par approximation successive
- La conversion parallèle ou Falsh

Dans notre projet la conversion dans le PIC s'effectue par approximation successive dont le fonctionnement est détaillé ci dessous :

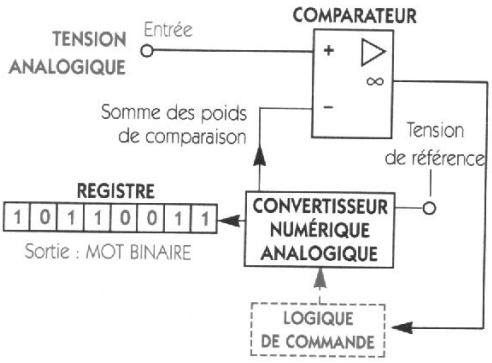


Figure III.4 : Principe de conversion d'un ADC de PIC 16F877

❖ L'organigramme d'acquisition de la mesure

Cet organigramme (Figure III.5) représente les étapes d'acquisition d'un signal analogique appliqué sur les pins du CAN.

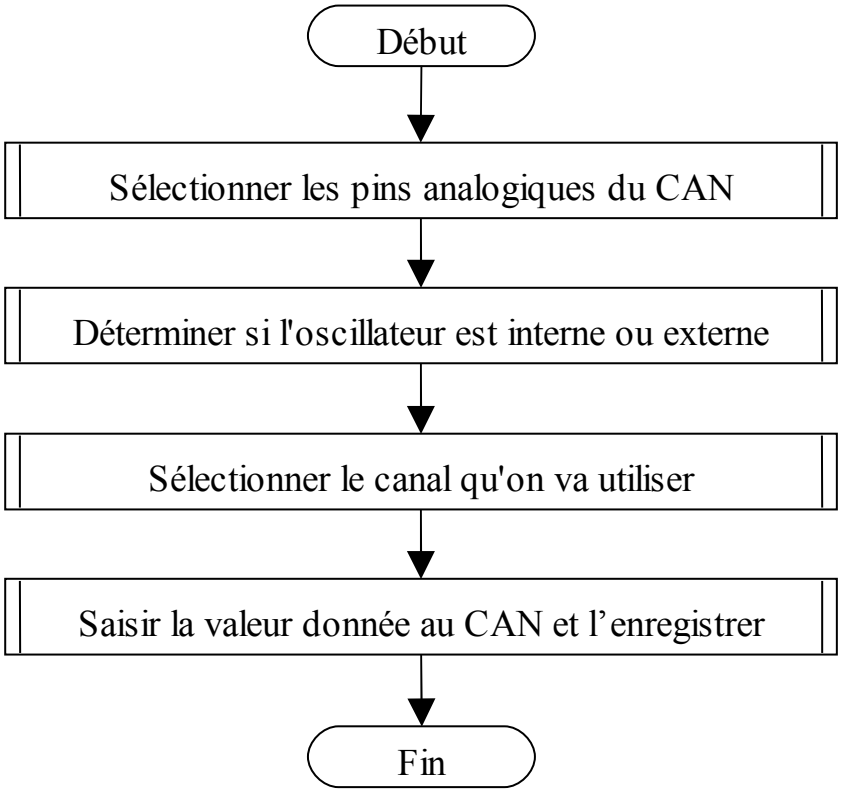


Figure III.5 : L'organigramme d'acquisition de la mesure

III.2.2- Le convertisseur numérique analogique

On dispose d'un mot numérique de n bits, que l'on voudrait convertir en une tension analogique, en considérant un code binaire :

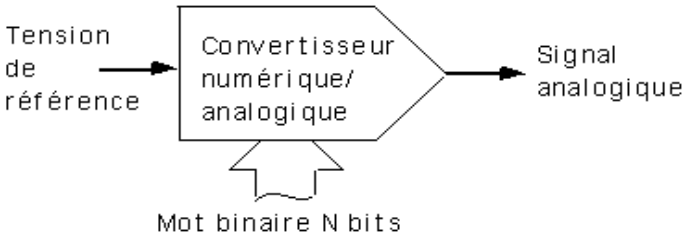


Figure III.6 : Schéma d'un convertisseur numérique analogique

- Le convertisseur de courant :

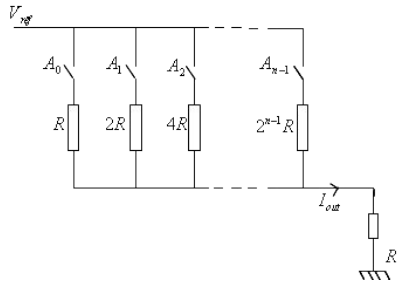


Figure III.7 : Schéma de principe d'un CNA de courant

Ce convertisseur fournit un courant qui dépend de la tension référence $V_{réf}$ ainsi que les états des interrupteurs suivant la formule suivante :

$$I_{out} = I_{réf}\left(A_0 + \frac{A_1}{2} + \frac{A_2}{4} + \cdots + \frac{A_{n-1}}{2^{n-1}} \right)$$

Avec : $I_{réf} = \dfrac{V_{réf}}{R}$

Si on veut obtenir un CNA de tension, on est amené à ajouter un amplificateur [10].

Dans notre projet nous avons utilisé un convertisseur analogique numérique de courant "le DAC0808" suivi d'un convertisseur courant/tension à base d'un amplificateur opérationnel.

- Le DAC 0808

Les DAC0808 sont des circuits monolithiques convertisseurs D/A 8 bits fournissant un courant pleine échelle de 150 mA et ne dissipant que 33 mW avec une alimentation égale à 5 V. L'ajustement du courant de référence (Iréf) n'est pas nécessaire dans la plupart des applications puisque le courant pleine échelle est de 1 LSB de $\dfrac{255}{256}$ Iréf .

Les DAC0808 s'interfacent directement avec la logique TTL, DTL ou CMOS.

Concernant le brochage du DAC0808, il est donné par le schéma suivant :

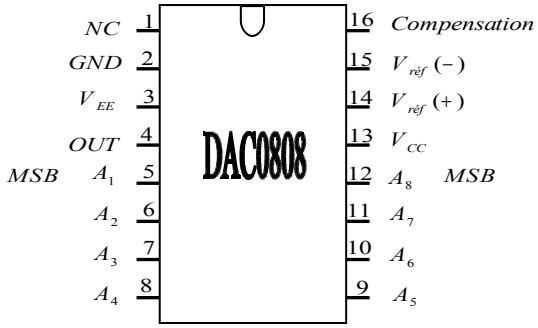


Figure III.8 : Brochage du DAC0808

le tableau ci-dessous donne un aperçu sur les valeurs limites d'un CNA de type DAC0808 :

V_{CC}	+18V DC
V_{EE}	-18V DC
Tension d'une entrée digitale, V5-V12	-10 VDC à +18 VDC
Tension de sortie, Vo	-11 VDC à +18 VDC
Courant de référence, I14	5 mA
Tension de référence V14, V15	V_{CC} et V_{EE}
Puissance dissipée	1000 mW
Décote au-dessus de 25 °C	6.7 mW/°C
T° de fonctionnement DAC0808L	-55°C ≤ TA≤ +125°C
T° de fonctionnement DAC0808LC séries	0≤ TA ≤+75°C
Plage de températures de stockage	-65°C à +150°C

Figure III.9 : Caractéristiques du DAC0808

L'application typique du DAC0808 utilisée pour notre carte est de convertir le courant issu du convertisseur en une tension proportionnelle en utilisant le circuit LF351 qui est un amplificateur. Le schéma IV.12 illustre ce fonctionnement :

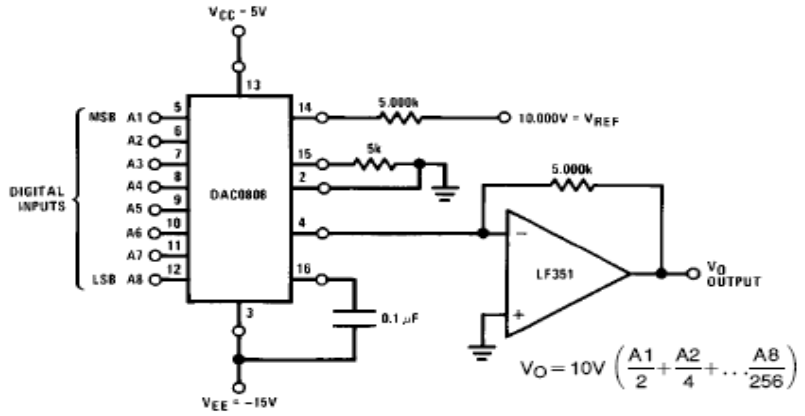


Figure III.10 : Schéma du câblage du DAC0808 avec l'amplificateur LF351

III.2.3. communication série asynchrone a travers le port série RS232

Les liaisons séries permettent la communication entre deux systèmes numériques en limitant le nombre de fils de transmission.

La liaison série aux normes de RS232 est utilisée dans tous les domaines de l'informatique. Elle est de type asynchrone, c'est-à-dire qu'elle ne transmet pas le signal de l'horloge.

Le schéma fonctionnel est le suivant :

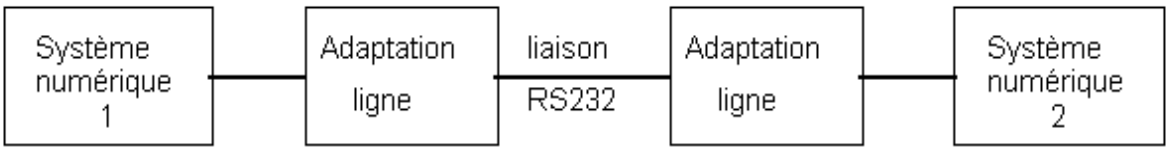


Figure III.11 schéma fonctionnel d'une liaison série asynchrone de la norme RS232

La transmission série nécessite au moins 2 fils de communication, l'un pour la transmission (Tx) et l'autre pour la réception (Rx)et un fil de masse.

III.2.3.1- Protocole de transmission

Afin que les éléments communicants puissent se comprendre, il est nécessaire d'établir un protocole de transmission. Ce protocole devra être le même pour les deux éléments afin que la transmission fonctionne correctement.

Paramètres rentrant en jeu :

- Longueur des mots : 7 bits (ex : caractère ascii) ou 8 bits

- La vitesse de transmission : les différentes vitesses de transmission son réglables à partir de 110 bauds (bits par seconde) de la façon suivante : 110 bds, 150 bds, 300 bds, 600 bds, 1200 bds, 2400 bds, 4800 bds, 9600 bds.
- Parité : le mot transmis peut être suivi ou non d'un bit de parité qui sert à détecter les erreurs éventuelles de transmission. Il existe deux types de parité.
    - ✓ parité paire : le bit ajouté à la donnée est positionné de telle façon que le nombre des états 1 soit paire sur l'ensemble donné + bit de parité
      ex : soit la donnée 11001011 contenant 5 état 1, le bit de parité paire est positionné à 1, ramenant ainsi le nombre de 1 à 6.
    - ✓ parité impaire : le bit ajouté à la donnée est positionné de telle façon que le nombre des états 1 soit impaire sur l'ensemble donné + bit de parité
      ex : soit la donnée 11001011 contenant 5 état 1, le bit de parité paire est positionné à 0, laissant ainsi un nombre de 1 impaire..
- Bit de start : la ligne au repos est à l'état logique 1 pour indiquer qu'un mot va être transmis la ligne passe à l'état bas avant de commencer le transfert. Ce bit permet de synchroniser l'horloge du récepteur.
- Bit de stop : après la transmission, la ligne est positionnée au repos pendant 1, 2 ou 1,5 périodes d'horloge selon le nombre de bits de stop.

Le bit de start apparait en premier dans la trame puis les données (poids faible en premier), la parité éventuelle et le (les) bit(s) de stop.

exemple :

Soit à transmettre en parité paire, avec 2 bits de stop, le caractère B dont le codage ascii est 1000010(2) la trame sera la suivante :

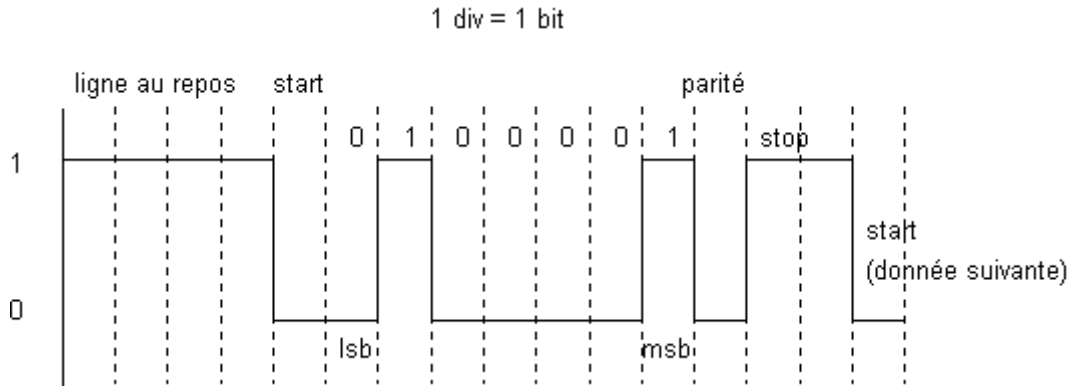
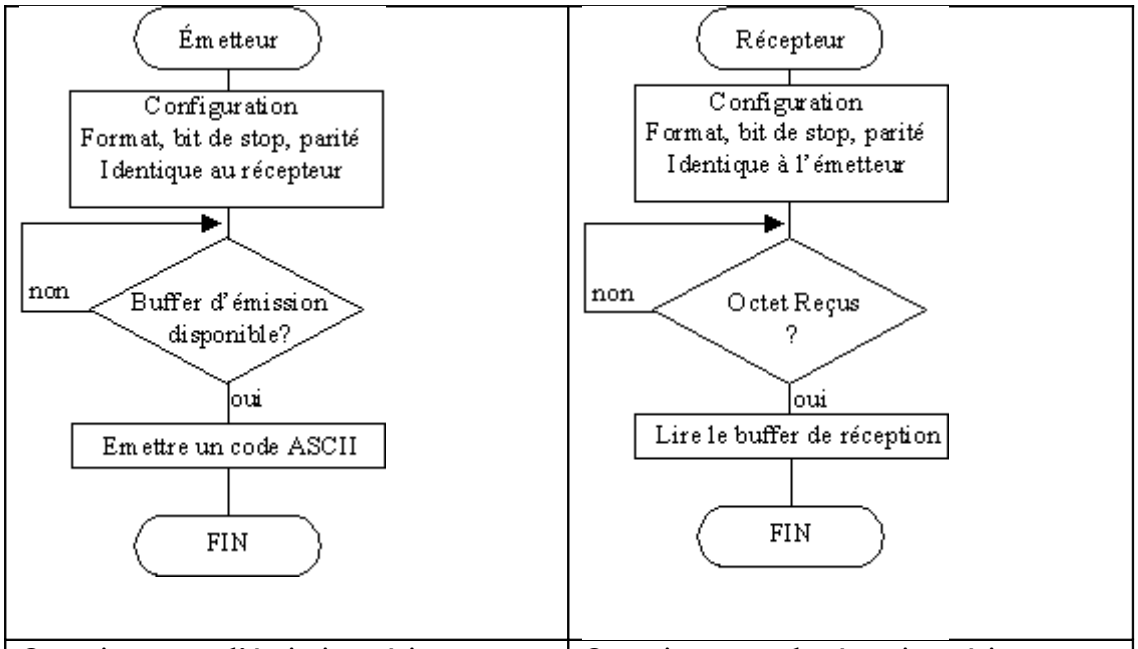


Figure III.12 Exemple de transmission série

Les deux organigrammes ci-dessous présenteront le principe de la transmission et la réception série que nous avons utilisé pour la programmation de notre PIC, évidemment le même principe sera utilisé pour le programme implanté sur LabVIEW.



Organigramme d'émission série	Organigramme de réception série

Figure III.13 Organigramme de la transmission série

III.2.3-2- Principe d'adaptation PIC- RS232

Passons maintenant au principe d'adaptation entre le PIC et le port série Rs232.

Le PIC utilise les niveaux 0V et 5V pour définir respectivement des signaux « 0 » et « 1 ». La norme RS232 définit des niveaux de +12V et –12V pour établir ces mêmes niveaux.

Nous aurons donc besoin d'un circuit chargé de convertir les niveaux des signaux entre PIC et PC. La pin TX du PIC émettra en 0V/5V et sera convertie en +12V/-12V vers notre PC. La ligne RX du PIC recevra les signaux en provenance du PC, signaux qui seront converti du +12V/-12V en 0V/5V par notre circuit de pilotage du bus.

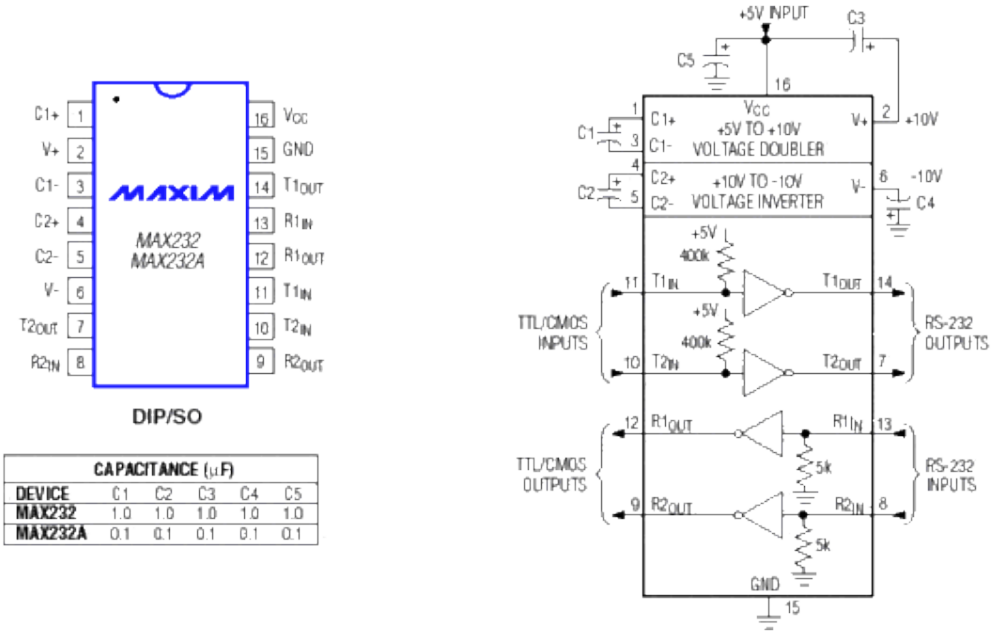
Notons que la liaison étant full-duplex, émission et réception sont croisées, chaque fil ne transitant l'information que dans un seul sens.

Nous utiliserons le célèbre circuit MAX232 pour effectuer cette adaptation de niveaux. Ce circuit contient un double convertisseur à double direction. Autrement dit, il dispose de :

- 2 blocs, dénommés T1 et T2, qui convertissent les niveaux entrés en 0V/5V en signaux sortis sous +12V/-12V. En réalité, on n'a pas tout à fait +12V et -12V,

mais plutôt de l'ordre de +8,5V/-8,5V ce qui reste dans la norme RS232.

- 2 blocs, dénommés R1 et R2, qui convertissent les niveaux entrés en +12V/-12V en signaux sortis sous 0V/5V.



CAPACITANCE (µF)					
DEVICE	C1	C2	C3	C4	C5
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1

Figure III.14 : Circuit intégré MAX232

III.3-descriptions de la carte d'Alimentation stabilisé

Notre carte doit être alimentée par une alimentation stabilisée qui fournit par 5V à la sortie pour alimenter notre PIC et ±12V pour l'alimentation du circuit LF351N.

La carte réalisée contient :

- ❖ Un transformateur abaisseur, qui fournit sur son secondaire une tension alternative très inférieure à celle du secteur (220V/17V)
- ❖ Un pont redresseur (diodes en pont de Graëtz), qui fournit en sortie une tension non plus alternative mais redressée,
- ❖ cinq capacités de filtrage, qui réduisent l'ondulation de la tension issue du pont redresseur,
- ❖ trois régulateurs de tension, dont le rôle est de stabiliser le potentiel de sortie à une certaine valeur 5V, 12V ou -12V. enfaîte Pour avoir une tension de 5V on a choisi l'un des régulateurs de tension les plus utilisé le 7805, également pour les tensions 12V et -12V, on a choisi respectivement le 7812 et le 7912.

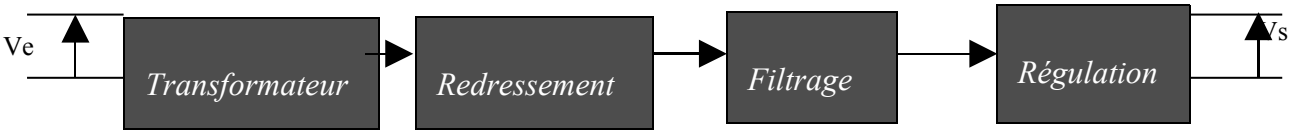


Figure III.15 : Schéma synoptique

III.3.1 Choix du transformateur

Notre Carte de développement nécessite une tension d'alimentation de (5V, ±12v) et un courant de 0.5A, ainsi nous devons choisir un transformateur a point milieu dont la tension maximale au secondaire est supérieure à 15.5V

III.3.2 Choix de pont de redressement

Le choix de pont de diode est basée essentiellement sur :

- La tension inverse maximale de diode.
- Le courant moyen direct.

III.3.3 Choix des condensateurs

- Choix des condensateurs de filtrage:

Pour obtenir une tension presque constante il faut brancher un ou plusieurs condensateurs en parallèle juste à la sortie de pont redresseur, plus la valeur de la capacité est élevée plus le filtrage sera meilleur. Les deux principaux critères à considérer dans le choix d'un condensateur sont :

1. sa capacité
2. sa tension de service

- Choix des condensateurs de découplage :

On les choisi de tel sorte qu'ils servent à améliorer la stabilité du régulateur

III.3.4 Choix de régulateur de tension

Un régulateur de tension intégré est un composant à semi-conducteur dont le rôle consiste à rendre quasi continue une tension qui présente une ondulation issue d'un pont redresseur et à stabiliser sa valeur.

La tension de sortie V_{out} est le principal critère de choix, puisqu'elle correspond à la tension désirée. Ainsi, pour une tension de 5V, on choisira un LM705 qui possède les caractéristiques suivantes :

Courant de sortie 1A.

Protection thermique interne contre les surcharges.

Aucun composant externe nécessaire.

Plage de sécurité pour le transistor de sortie.

Limitation interne du courant de court-circuit.

III.4 conclusion

On a vue donc les différents éléments constituant la carte d'interfaçage réalisé, leurs fonctionnements, ainsi que le principe de la transmission série et celui de l'adaptation en ligne.

Nous verrons dans le chapitre suivant les principales caractéristiques du logiciel LabVIEW, ainsi que les résultats obtenus après la régulation.

Chapitre IV : Commande et régulation dans LabVIEW

IV. Commande et régulation dans LabVIEW

IV.1- Introduction à la programmation graphique

LAbVIEW (Laboratory Virtual Instrument Engineering Workbench) est un langage de programmation dédié au contrôle d'instruments et l'analyse de données. Contrairement à la nature séquentielle des langages textuels, LabVIEW est basé sur un environnement de programmation graphique utilisant la notion flot de données pour ordonnancer les opérations.

Enfaite il existe deux formes de programmation graphiques:

Une programmation dite flot de contrôle et une autre dite flot de donnés.

VI.1.1-programmation flot de contrôle

Ils ont longtemps été utilisés pour décrire les algorithmes, ces représentations décrivent les programmes comme étant de nœud de calcul connecté par des arcs spécifiant quel calcul doit être effectué ensuite

Ex : Grafcet et Réseau de petri.

IV.1.2-Programmation flot de donné

C'est une fonction analogue à la propagation du signal à travers un circuit électrique. Le digramme flot de données est un graphe acyclique qui peut être composé de 3 éléments suivants :

- Des terminaux : qui sont les liens avec l'extérieur qui représente la production où la consommation de données.
- Des Nœud qui sont les traitements à effectué et qui sont représenté par une figure géométrique pouvant contenir une image illustrant leur fonctionnalité
- Les arcs orienté : qui relie les nœuds et les terminaux et permettent d'indiqué le passage de données d'un nœud vers un autre.

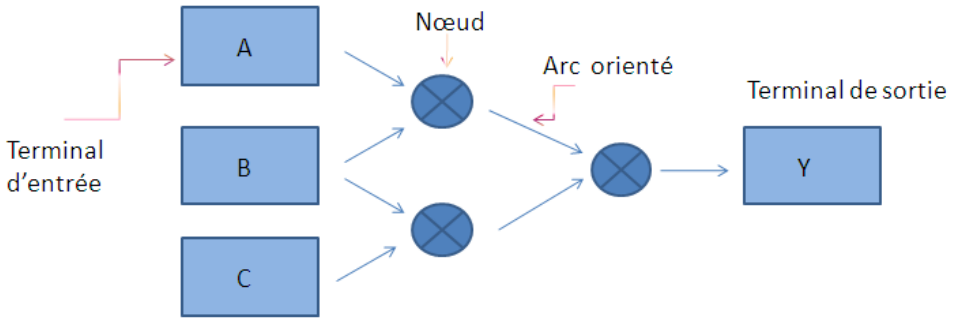


Figure IV.1 : Exemple de diagramme de flot de donnés

IV.2- L'environnement LabVIEW

LabVIEW est centré autour du principe d'instrument virtuel (Virtual Instrument ou encore VI). IL se décomposer en deux parties :

- la première partie (partie cachée ou interne) : elle contient l'algorithme du programme décrit sous la forme d'un diagramme flot de données en langage graphique.
- la seconde partie (partie visible) est constituée de l'interface utilisateur.

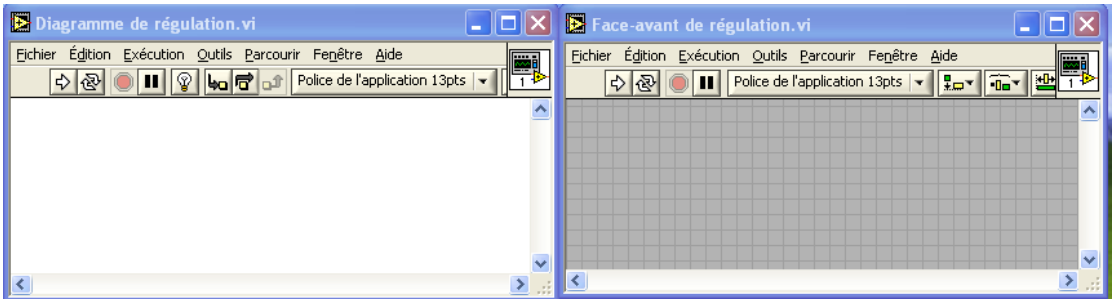


Figure IV.2 : Fenêtre de l'environnement de développement sur LabVIEW
Face avant (à droite) et Diagramme (à gauche)

Pour écrire un programme sur LabVIEW, on a besoins des « Palettes » qui nous offre la possibilité de modifier la face avant et le digramme de LabVIEW, on trouve trois palettes :

- **Palette d'outils**

Elle est disponible sur la face-avant et sur le diagramme, elle contient les outils nécessaires pour faire fonctionner et modifier la face avant et les objets du diagramme.

Figure IV. 3 :palette d'outils

- **Palette de commandes**

Elle est disponible uniquement sur la face-avant, elle contient les commandes et les indicateurs de la face-avant nécessaire pour créer l'interface utilisateur.

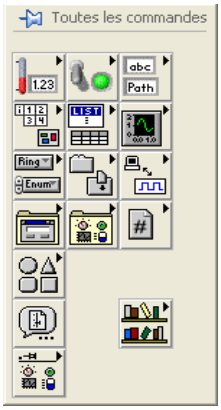


Figure IV. 4 : palette de commandes

- **Palette de fonctions**

Elle est disponible uniquement sur le diagramme. Elle contient les objets nécessaire pour la programmation graphique comme les opérations d'arithmétique, d'E/S d'instrument, d'E/S de fichier et d'acquisition de données.

Figure IV. 5 : palette de fonctions

IV.3- Structure de données dans LabVIEW

LabVIEW utilise un langage fortement typé et toutes données ou structure de données ne peuvent être manipulées qu'avec des fonctions admettant ce type, en faite dans LabVIEW on trouve les types de base scalaire, les types entiers (signés ou non, codés sur 8, 16 ou 32 bits), le type réel (codé sur 16, 32 ou 64 bits), le type booléen et le type chaîne de caractères (figure IV.5). Il est important de noter que les éléments représentant ces données, ainsi que les liaisons issues de ces éléments, sont de forme et de couleur différente.

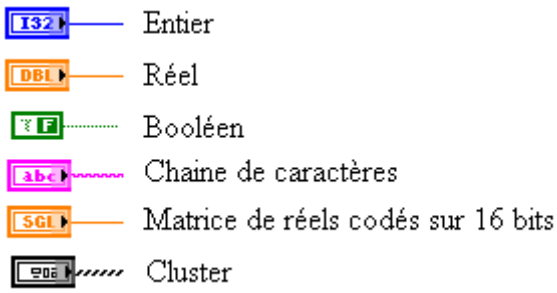


Figure IV. 6 :différents types de structures de données dans LabVIEW

Le langage permet aussi de créer des structures de données plus élaborées.

IV.3.1- le type tableau (structure de données homogènes)

Comme tous les langages de programmations la présence de notion de tableau est obligatoire, puisqu'elle définit un outil de base pour le stockage et les traitements des informations.

La bibliothèque tableau dans LabVIEW est riche en fonctions prédéfinit, tel que, l'initialisation, l'indexation, l'inversement d'un tableau, la concaténation de deux tableau, et plusieurs autre fonctions, bien sur, la notion des tableaux peut être élargie pour contenir les matrices. Et comme toutes les autres bibliothèques, on peut l'enrichir avec des fonctions ou des procédures créent à partir des langages de bas niveau comme le langage C.

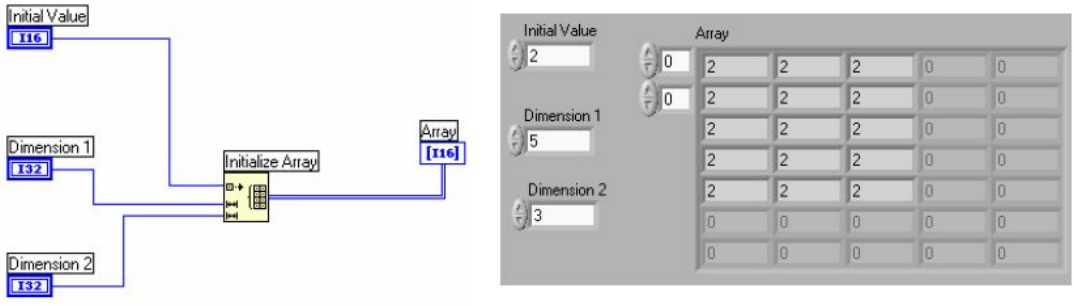


Figure IV. 7 : Exemple d'initialisation d'un tableau a 2 dimensions (matrice)

IV.3.2-le type « cluster » (structure de données hétérogènes)

Ce second constructeur de type est l'équivalent du « record » en Ada ou du « struct » en C. Les différents composants ou champs de ce groupe de données (cluster) peuvent être assemblés ou récupérés par des fonctions spécifiques.

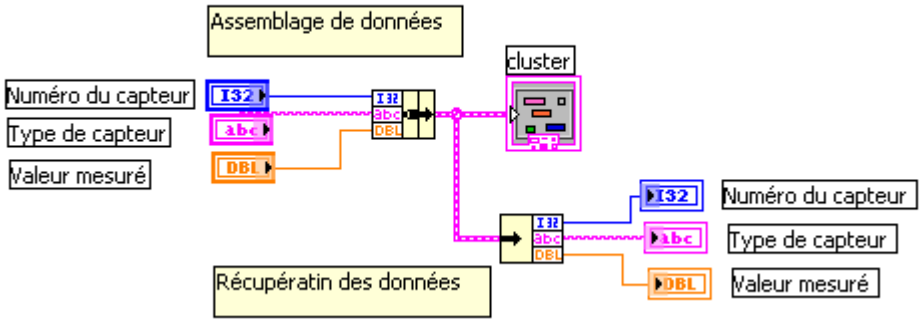


Figure IV. 8 : Exemple de manipulation des données avec le type cluster

IV.4 Structures de programmation

LabVIEW utilise un langage flot de données pur qui a été enrichi de quatre types de structures : la séquence, deux structures d'itération (la boucle« Pour » avec un nombre d'itérations fixé et la boucle « Tant Que » avec un nombre d'itérations soumis à condition) et la structure de choix.

IV.4.1- structure de séquence

La structure de « séquence » permet de spécifier l'ordre d'exécution de flots de données. Cette structure se présente sous la forme d'un cadre et a le statut d'un nœud

- **Exemple :**

Sur la figure (IV.9), l'utilisation de la séquence s'impose : il s'agit d'initialiser un port série puis d'envoyer une chaîne de caractères. Le premier flot de données risque de provoquer des erreurs car rien n'empêche le nœud d'écriture sur le port série de s'exécuter avant le nœud d'initialisation de ce port : l'utilisation de la séquence permet d'ordonner correctement les opérations.

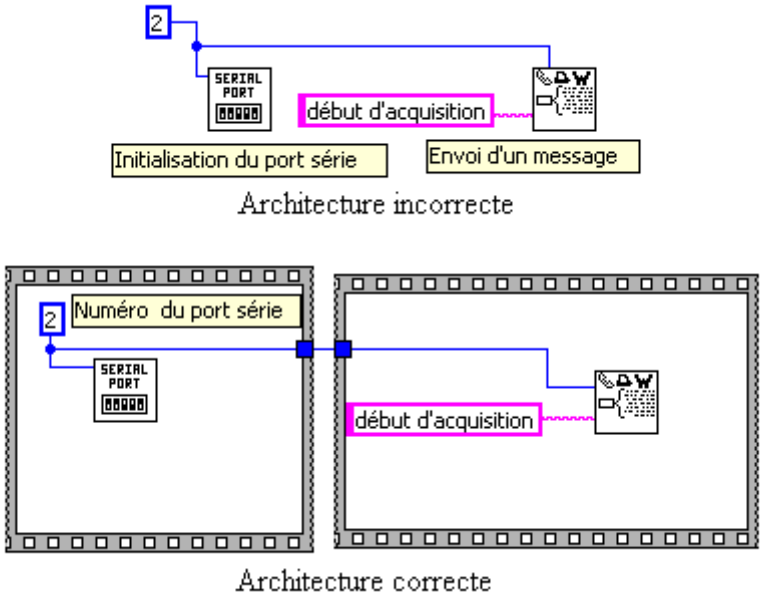


Figure IV.9 : Exemple d'utilisation de la structure de "séquence"

Figure IV. 9 : Exemple d'utilisation de la structure de séquence

IV.4.2- les structures itératives

Les deux structures itératives, la boucle « Pour » et la boucle « Tant que », ont aussi le statut d'un nœud ordinaire. La boucle « Pour » permet d'exprimer la répétition (ou itération) pour un nombre de fois prédéterminé défini par une connexion d'entrée obligatoire: le nombre d'itérations à effectuer N. À l'intérieur de la boucle « Pour » se trouve un terminal d'entrée local générant l'entier indiquant l'indice d'itération de la boucle (i varie de 0 à N–1).

- **Exemple :**

L'utilisation de cette structure, présentée sur la figure (IV.10), concerne l'acquisition de N mesures avec un temps fixe entre chacune, défini par un nœud de temporisation Les N données de sortie sont assemblées pour former un vecteur avec une auto-indexation.

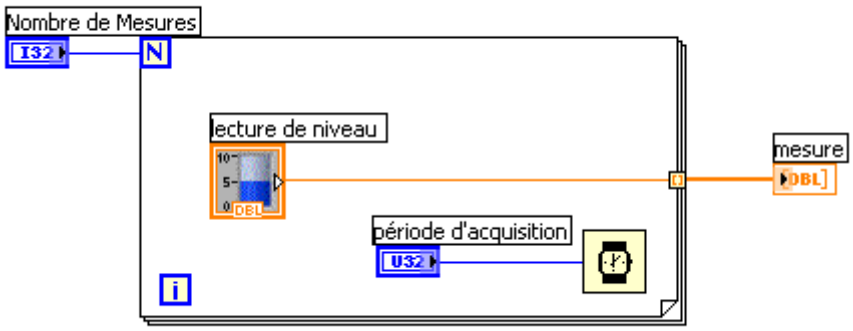


Figure IV.10 : Exemple d'utilisation de la structure itérative "pour"

La boucle « Tant Que » permet d'exprimer la répétition pour un nombre de fois non connu à l'avance. À l'intérieur de la boucle « Tant Que » se trouve un terminal d'entrée local générant l'entier indiquant l'indice d'itération de la boucle. Un terminal de sortie de type booléen permet d'arrêter la boucle lorsque la valeur « False » lui est envoyée.

- **Exemple :**

L'utilisation précédente décrite sur la figure 9, est reprise avec cette structure « Tant Que » en arrêtant l'acquisition pour une valeur trop grande de la mesure (figure IV.11).

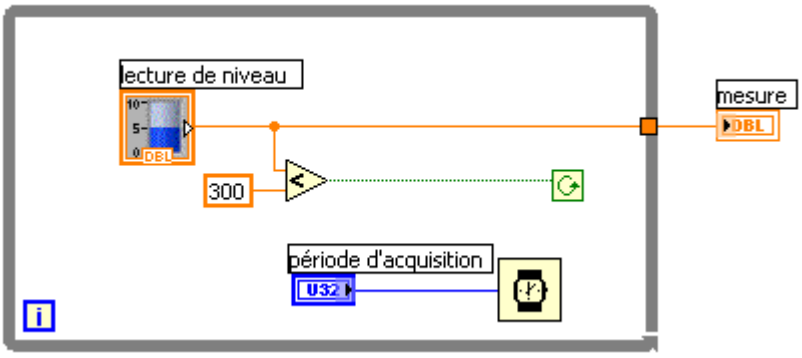


Figure IV.11 Exemple d'utilisation de la structure itérative "tant que"

Dans les deux cas de structure itérative lorsque des tableaux de données arrivent ou partent des tunnels, ceux-ci ont la possibilité d'être automatiquement indexés : récupération un par un des éléments du tableau ou concaténation pour créer un tableau. Dans le cas des deux structures, il est possible de mémoriser des résultats produits lors d'itérations antérieures et de les récupérer ensuite. Ces registres à décalage sont des variables locales à une boucle.

- **Exemple :**

La figure (IV.12) présente une utilisation de ces registres à décalage dans le cas de la boucle « Pour ». Il est important de remarquer que ces registres à décalage constituent un des effets de bord du langage. Il est de plus nécessaire d'initialiser ces mémoires locales pour réaliser une utilisation contrôlée et cohérente.

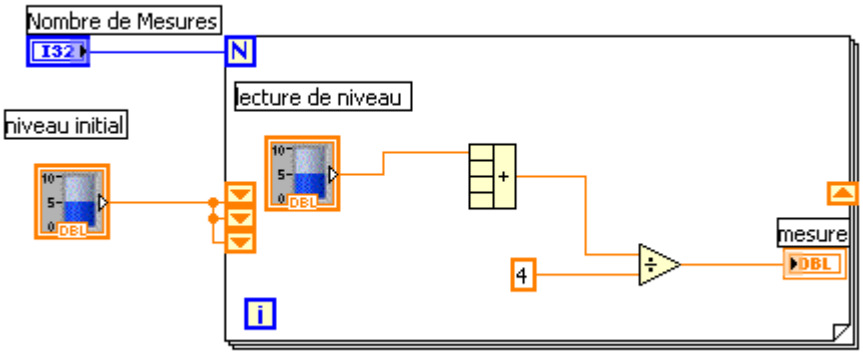


Figure IV.12 : Exemple d'utilisation des registres à décalage dans une boucle "pour".

IV.4.3- la structure de choix

La dernière structure nécessaire est celle qui va permettre d'exprimer l'alternative (le choix). La sélection du cas exécuté est faite par la valeur de la variable connectée à l'entrée représentée par un point d'interrogation (figure IV.13). L'identifiant du cas représenté est indiqué en haut de la structure.

- Exemple :

La figure (IV.13) présente une utilisation de la structure de choix. Si nous avons une action « vrais » alors il' y a une acquisition de donné sur le port série, et si l'action est « faux », alors il' y aura pas d'acquisition.

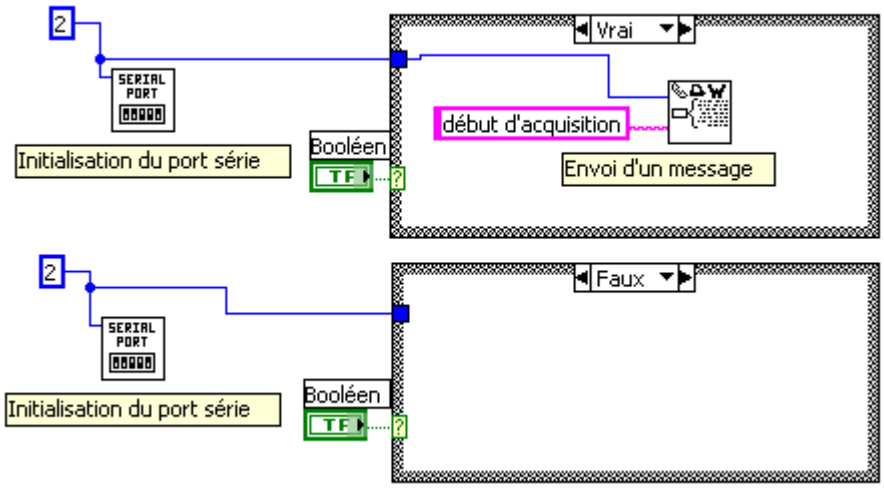


Figure IV.13 : Exemple d'utilisation de structure de choix.

IV.5-Traitement numérique

VI.5.1 : les fonctions prédéfinies

LabVIEW possède les instructions de base d'un langage de programmation permettant de traiter les différents types de données. Ainsi, nous avons des fonctions liées aux variables numériques (entiers, réels et complexes), aux variables booléennes, aux chaines de caractères, et aux tableaux.

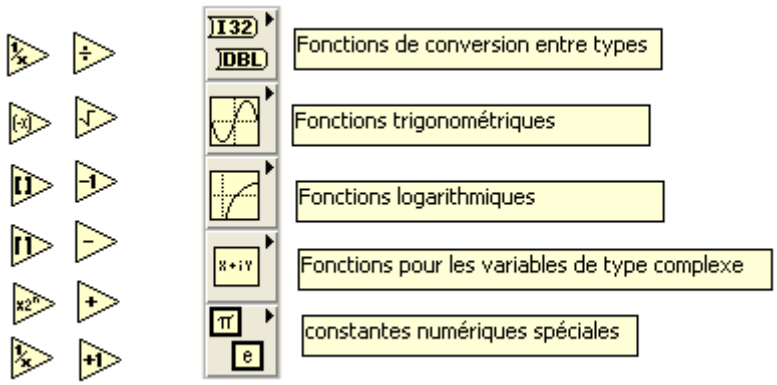


Figure IV.14 : Quelques instructions de traitement de données numérique dans LabVIEW

Nous trouvons aussi les opérateurs de test et de comparaisons liées à ces différents types de données. À partir des structures de contrôle, des fonctions et des opérateurs, de base, il est alors possible de traduire un algorithme quelconque et d'enrichir la bibliothèque des fonctions en utilisant le mécanisme d'encapsulation. Un diagramme complet est alors réduit à un nœud qui peut être ensuite réutilisé.

VI.5.2 : Boites à outils mathématique

LabVIEW contient aussi des boites de calcul mathématiques qui servent à introduire des commandes complexes tel que (linspace , ode45 ,sin, cos , etc..)

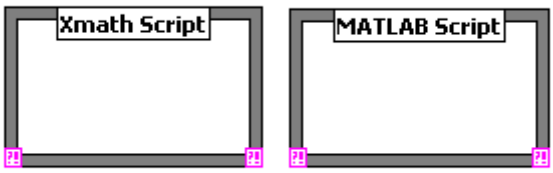


Figure IV.15 : Exemple de boite de calcul mathématique dans LabVIEW

IV.6- Bibliothèques de commande

LabVIEW possède des bibliothèques de fonctions spécialisées dans le domaine de la mesure, du test et du contrôle-commande. Ces bibliothèques de fonctions peuvent être réalisées soit à partir des fonctions et opérateurs de base vus précédemment, soit directement par des programmes écrits en langage de haut niveau, compilé et ensuite intégré sous la forme d'un nœud graphique.

Cette dernière possibilité est aussi offerte aux utilisateurs de cet environnement LabVIEW par l'intermédiaire de la construction de nœuds particuliers appelés « nœud d'interface vers code » (CIN : Code Interface Node). L'utilisation de ces fonctions qui peuvent être assez complexes est facilitée par une aide en ligne qui donne pour chaque fonction une description sommaire, souvent suffisante de ses spécifications.

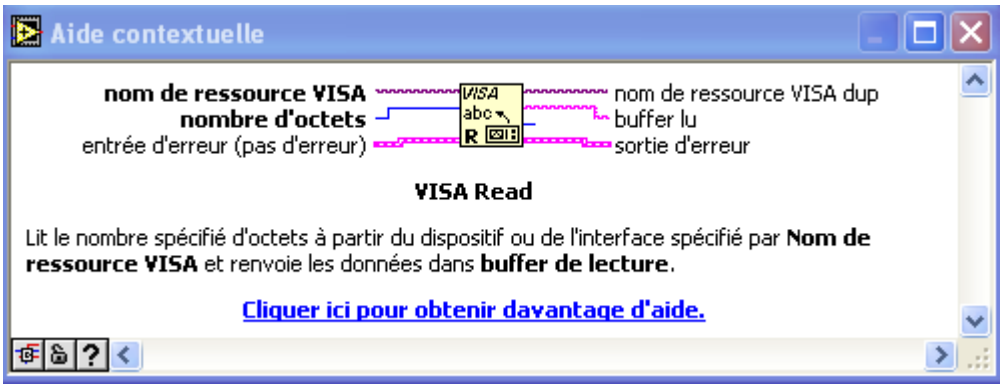


Figure IV.16 : Exemple de fenêtre d'aide offrant la description d'une fonction fournie dans la bibliothèque de LabVIEW

Dans notre projet on a besoin des bibliothèques spécifiques qui n'existe pas dans les versions standard de LabVIEW, mais dans des versions nommée (LabVIEW RT) c.-à-d. real time comme les bibliothèques de commande et de contrôle, enfaite nous avons utilisé des bibliothèques nommées « PID control toolset » et « fuzzy control toolset » pour la commande et la régulation de notre système. Il existe d'autre bibliothèque de commande telle que « DSP control toolset » et quelques autres commandes avancées.

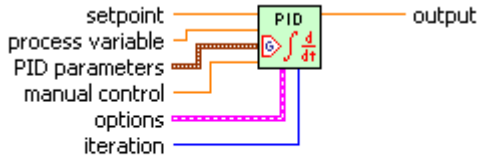
On va présenter d'abord les deux bibliothèques de commande que nous avons utilisée dans notre projet :

IV.6.1-La bibliothèque de contrôle PID

Elle ajoute aux fonctions standards de LabVIEW un ensemble d'algorithmes de régulation P, PI, PD et PID, avec toutes les options que l'on trouves généralement dans les systèmes de régulation complets.

Tous les régulateurs dans cette bibliothèque ont une structure parallèle de la forme

$$C(p) = \frac{U(p)}{\varepsilon(p)} = K_P + \frac{1}{pT_i} + pT_d$$



PID.vi

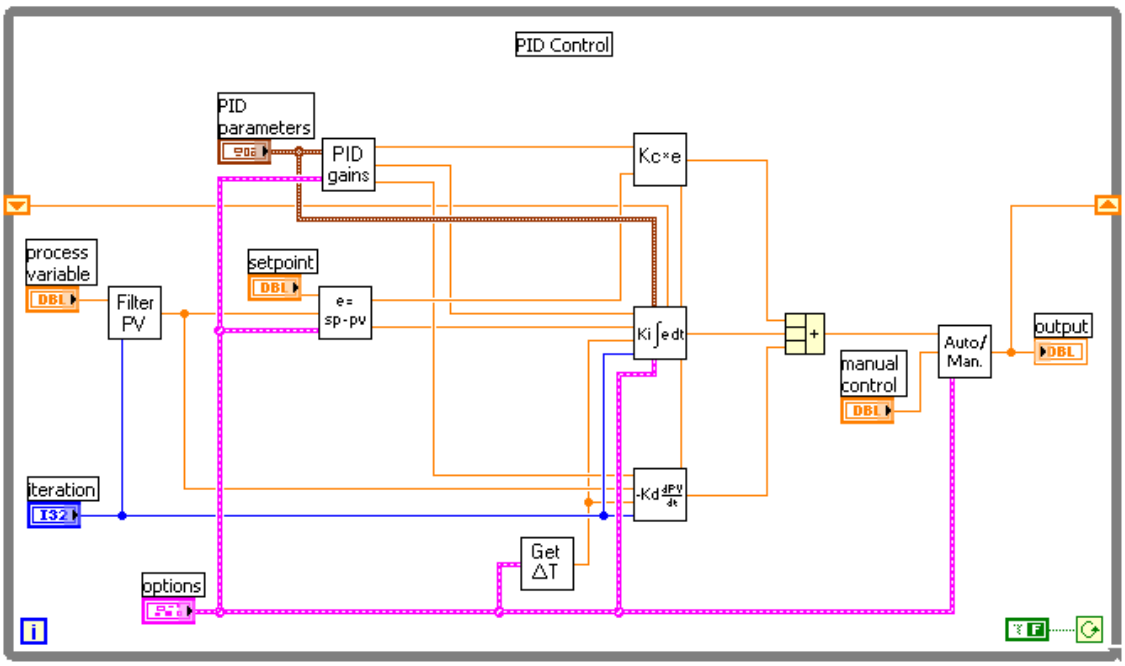


Figure IV.17 : Exemple de bibliothèque de régulation PID
Figure IV.18 : structure interne de PID

IV.6.2-La bibliothèque de logique floue

Elle est destinée à accélérer le développement d'applications de contrôle pour les systèmes non linéaires ou complexes. Elle propose une interface graphique prête-à-l'emploi pour la conception de contrôleurs de logique floue et des VI pour créer ces contrôleurs dans LabVIEW.

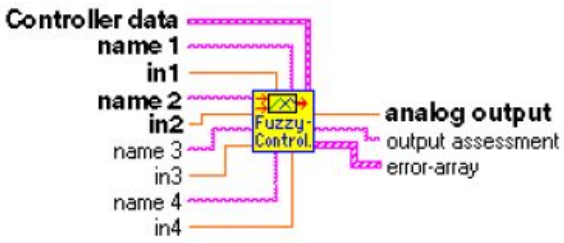


Figure IV.19 : Exemple de Bibliothèque de la commande floue

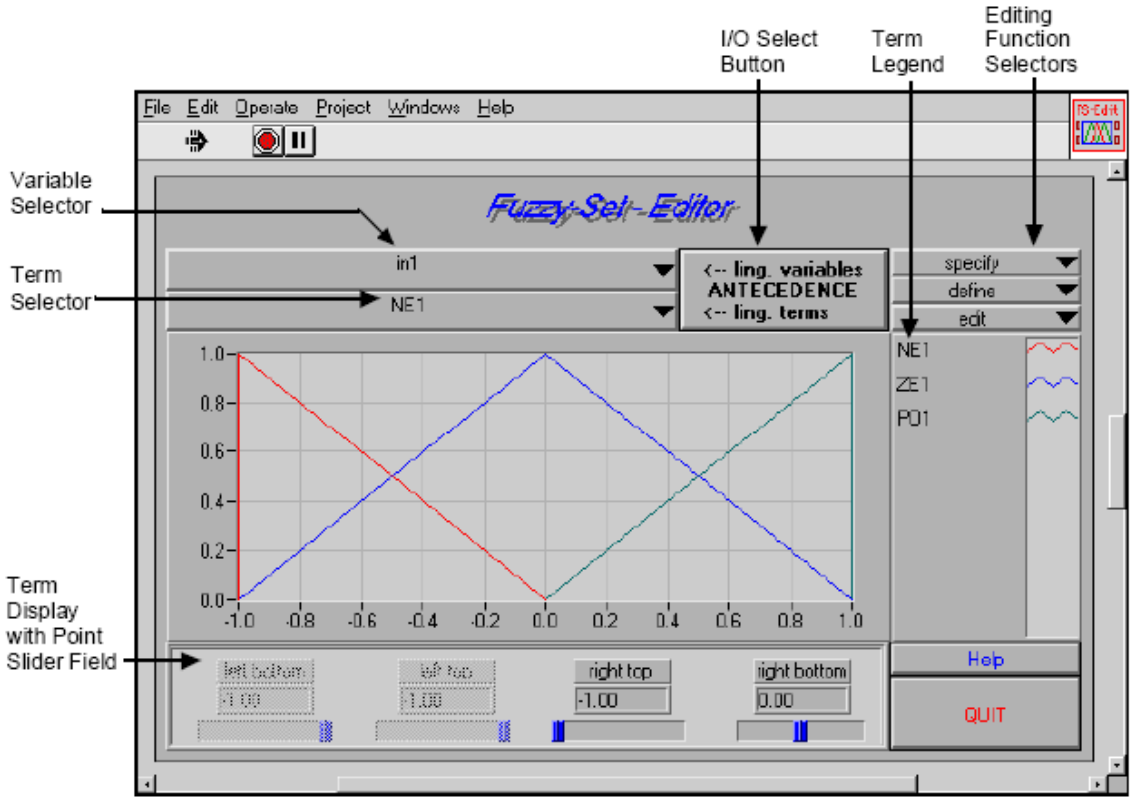


Figure IV.20 : Interface graphique pour la conception d'une commande floue

IV.7- Transmission série dans LabVIEW

IV.7.1- Bibliothèques RS232

La liaison série utilise un émetteur pour envoyer les données les unes derrière les autres, bit par bit, sur une ligne de communication unique, à destination d'un récepteur. Cette technique de communication est classique pour les transferts à faible vitesse ou sur de longues distances.

Étant donné la simplicité de la liaison série, la bibliothèque pour le port série se résume à cinq instruments virtuels (VI): Serial Port Init, Serial Port Write,

Serial Port Read, Bytes at Serial Port et Serial Port close. Un programme de gestion de port série, réalisé en langage G dans l'environnement LabVIEW est donc très simple puisqu'il consiste à mettre en séquence les différents VI de

base.

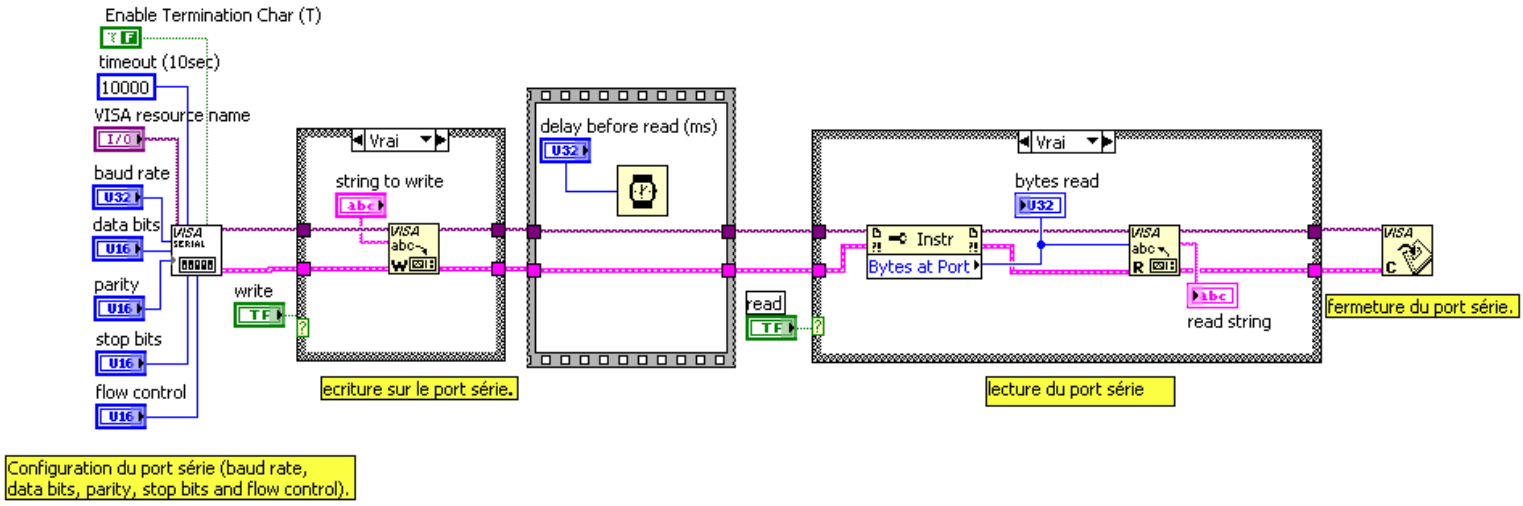


Figure IV.21 Bibliothèque du port série sur LabVIEW

IV.8- application de la régulation PID dans LabVIEW

IV.8.1-boucle de régulation :

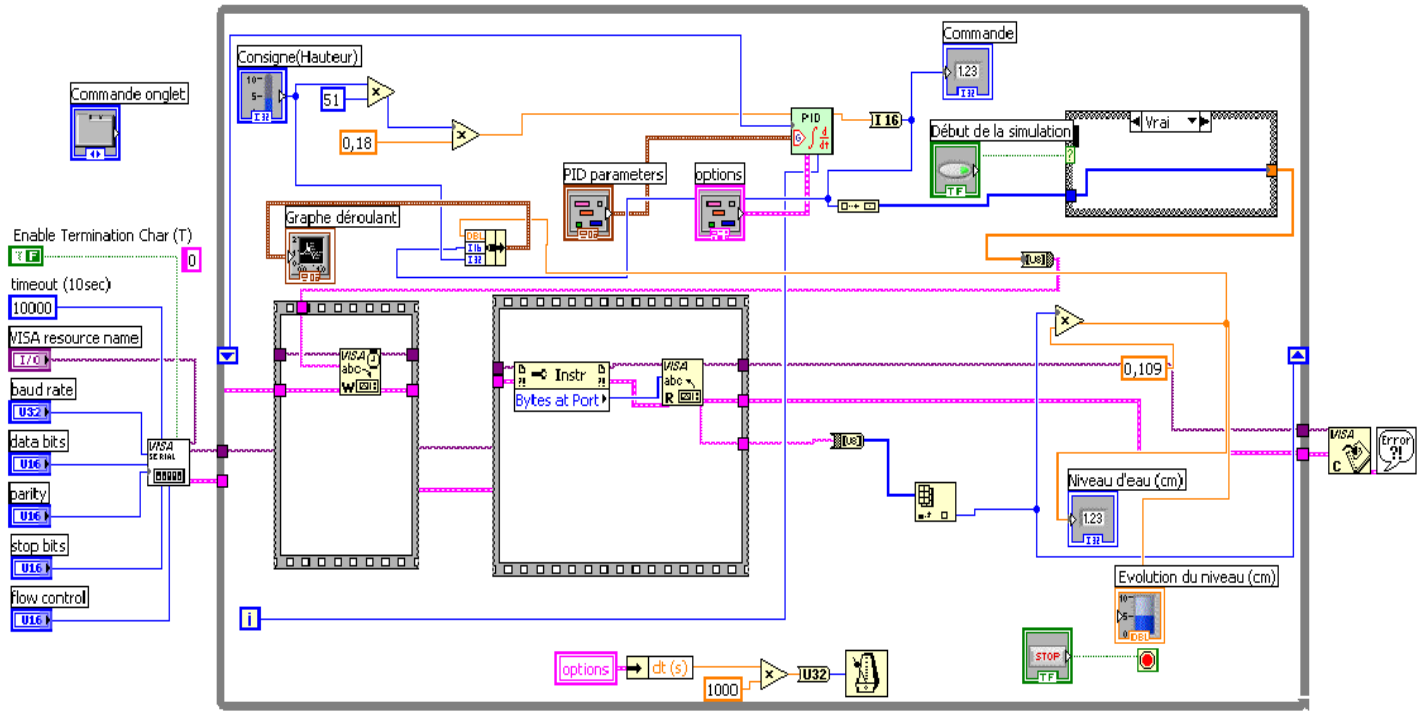


Figure IV.22 : Diagramme de commande PID

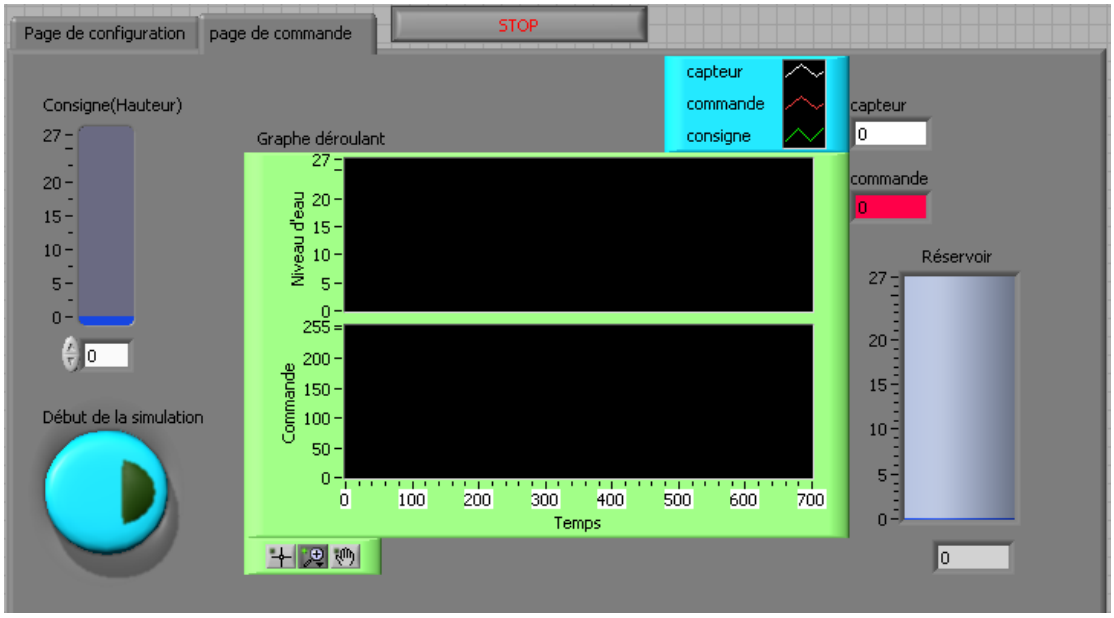


Figure IV.23 : page de commande PID

IV.8.2- configuration des paramètres

- On fixe la consigne désiré et les paramètres de PID, dans notre application on a choisie Kp=10, Ti=0.3 min Td=0 min, et la période d'échantionnage minimale Te=0.3s.

La détermination de ces paramètres est faite grâce à la méthode des approximations successives.

Cette méthode consiste à faire des testes sur chacune des actions du régulateur indépendamment des autres actions

- Nous avons configuré le port série comme suit :
  - port de communication : port2
  - vitesse de transmission : 9600 baud
  - nombre de bit à transmettre : un octet
  - sans bit de parité
  - un seul bit de stop
  - sans contrôle de flux

Figure IV.24 : page de configuration PID

IV.8.3-résultats expérimentaux

IV.8.3.1-En asservissement

Cas A (voir aussi annexe)	Cas B (voir aussi annexe)

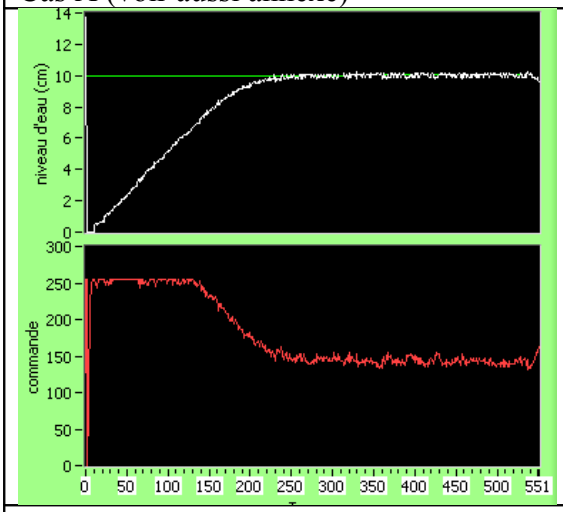
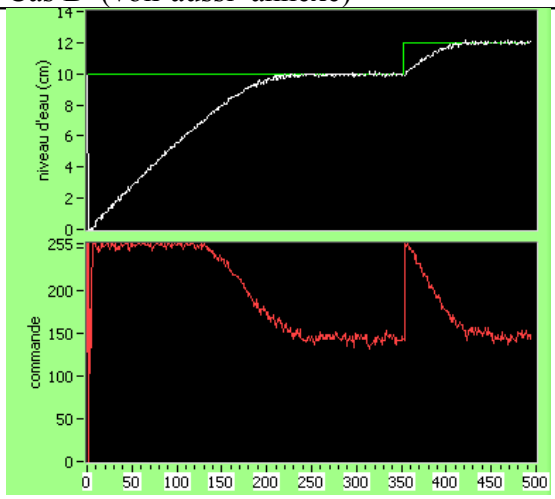

	

Figure IV.27 : réponse du système à une consigne de 10 cm	***Figure IV.28 : réponse du système après un changement de consigne de 10 cm à 12 cm***

- Le cas A présente un mode d'asservissement normale (suivie de la consigne initiale).

- Le cas B présente un changement de consigne de 10 cm au 12 cm, on remarque que la commande change au même temps de changement de la consigne

IV.8.3.2-En régulation

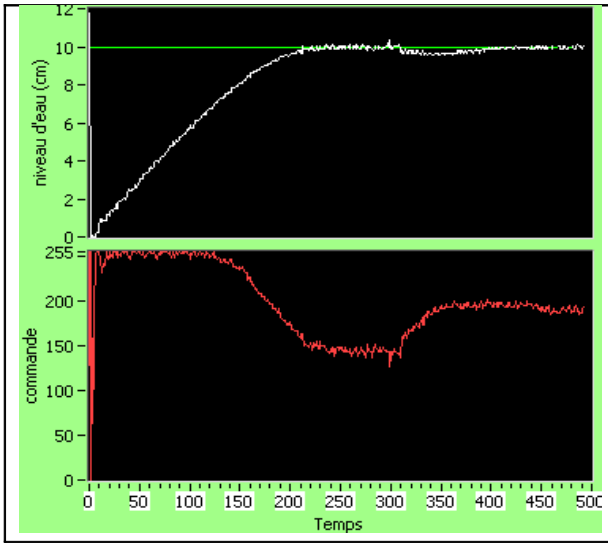


Figure IV.29 : réponse du système après une perturbation sur la vanne

- Lorsque on aboutie à un état stable on change l'ouverture de la vanne de 50% à 100%, on effectue donc une perturbation sur le système.
- On remarque que la commande augmente pour corriger cette perturbation et garder le même état initial

IV.9- Application de la régulation Floue dans LabVIEW

IV.9.1-boucle de régulation

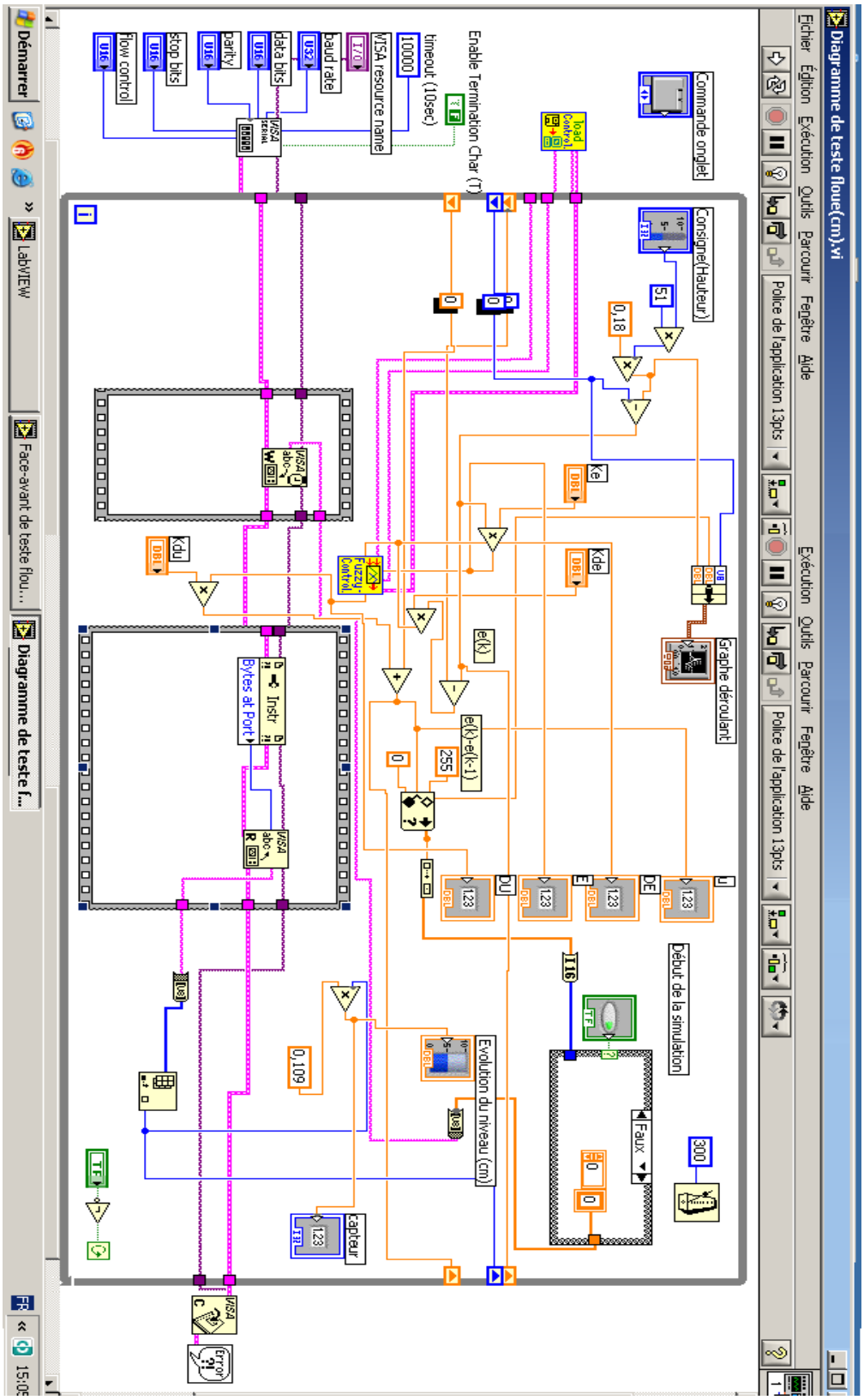


Figure IV.30 : Diagramme de la commande floue sur LabVIEW

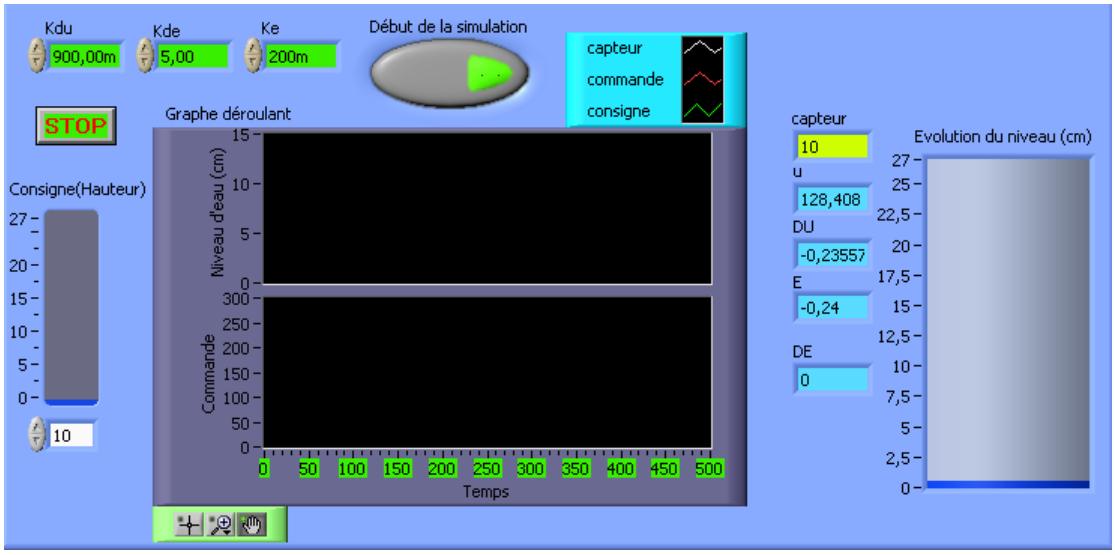


Figure IV.31 : Page de la commande floue sur LabVIEW

IV.9.2- configuration des paramètres floue

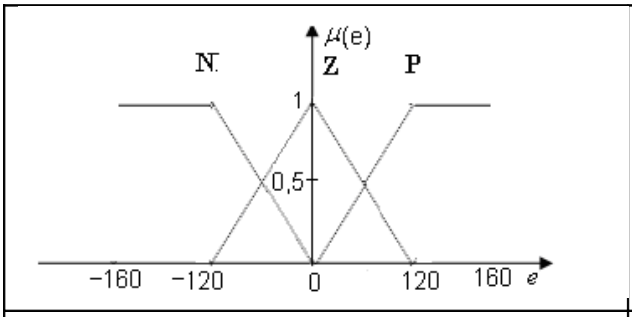


Figure IV.32 : Fonction d'appartenance pour l'erreur

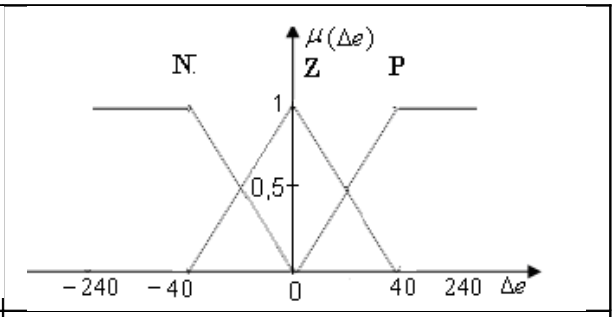


Figure IV.33 : Fonction d'appartenance pour la variation de l'erreur

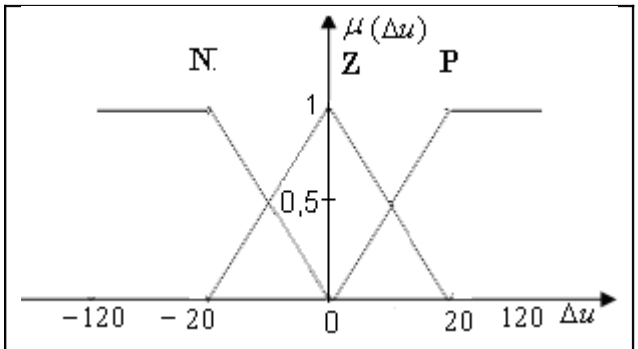


Figure IV.34 : Fonction d'appartenance pour la variation de la commande

Tableau IV.1 : Base de règles

$e / \Delta e$	N	Z	P

N	N	N	Z
Z	N	Z	P
P	Z	P	P

❖ Configuration du port série :

- port de communication : port2
- vitesse de transmission : 9600 baud
- nombre de bit à transmettre : un octet
- sans bit de parité
- un seul bit de stop
- sans contrôle de flux

IV.9.3-résultats expérimentaux

Cas A mode régulation

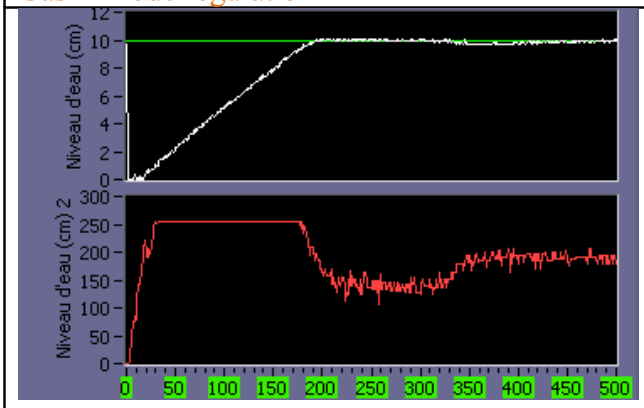


Figure IV.32: réponse du système après une perturbation sur la vanne

Cas B mode asservissement

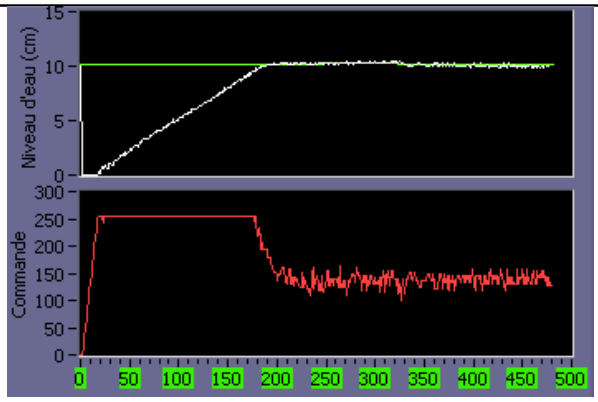


Figure IV.33: réponse du système à une consigne de 10 cm

➢ Conclusion :

Si on fait une comparaison entre les deux commandes (PI floue et PI classique), en remarque que la commande floue corrige mieux la perturbation et en plus elle répond plus rapidement que la commande PI.

Conclusion générale

Au cours de ce projet de fin d'études, la conception et la réalisation d'une carte d'interfaçage pour la régulation du niveau d'eau dans un réservoir nous a permis d'étudier la régulation et les commandes PI classique et PI floue.

Ce travail nous a permis donc d'enrichir nos connaissances dans plusieurs domaines et notamment dans le domaine de la régulation numérique et de l'électronique analogique et numérique.

Bibliographie

[1] : Cours d'Automatique 2005-2006
Bernard Bayle
Ecole Nationale supérieur de Physique de Strasbourg

[2] BELKACEM OULD-BOUAMAMA
Régulation automatique (le 15 juin 1998)
www.eudil.fr\eudil\belk\sc00a.htm

[3] Régulation industrielle –Notion de Base
Article PDF

[4] R.RHODE ENSEMBLE SCOLAIRE PRADEAU LA SEDE
Régulation analogique
Article PDF

[5] ALINA BESAÇON- VODA et SYLVIANNE GENTIL
Régulateurs PID analogique et numérique
Technique de l'ingénieur : R7416 (03/1999)

[6] M.ROBERT
Introduction au microcontrôleur
http://freelektronik.free.fr/LEKTRONIK/C9A.htm

[7] BIGONOFF
Cours: La programmation des pics par BIGONOFF
Première partie – PIC16F84 – Révision 6

[8] BIGONOFF
Cours: La programmation des pics par BIGONOFF
Second partie – La gamme Mid- Range par l'étude des 16F87X (16F876-16F877)

[9] Data sheet Microchip
PIC16F87X28/40-pin 8-Bit CMOS FLASH Microcontrollers

[10] Présentation de la structure interne du 16F877
Cours : fichier WORD

[11] Les convertisseurs
http://www.etronics.free.fr\dossiers\analog\analog11.htm

[12] Data sheet Analog Device
DAC0808/DAC0807/DAC0806 8-Bit D/A Converters

[13] Programmation en C des PIC.

[14] Guide d'utilisateur du logiciel PIC C

[15] Guide d'utilisateur du logiciel LabVIEW

[16] PFE. BEN HASSINE NAJDI et TURKI MOURAD
Conception et réalisation d'un système de commande numérique du niveau d'eau dans un réservoir
Mémoire PFE-ENIM2006

[17] Support de cours de techniques d'interfaçage
Elaboré par M GHORBEL MOHAMED 2006-2007.

ANNEXE

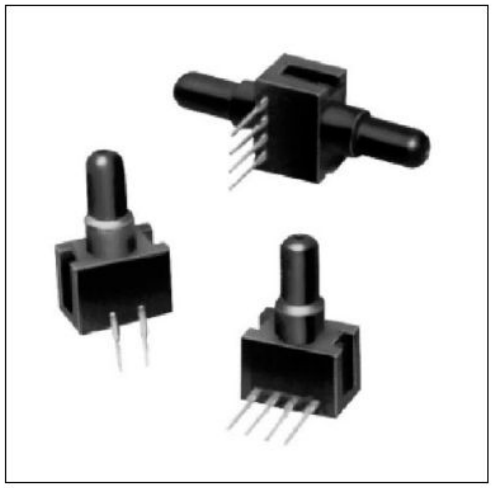
ANNEXE A

26PC Series *(mbar)*

Temp. compensated and calibrated pressure sensors

FEATURES

- **0...50 mbar to 0...16 bar gage or differential**
- **High impedance bridge**
- **Miniature package**
- **Different pinning configurations**
- **Usable for wet/wet applications[8]**



Scale: 1 cm
1 inch

SERVICE

All media compatible with

port 1: - polyetherimide
- silver-filled silicone
- silicon nitride

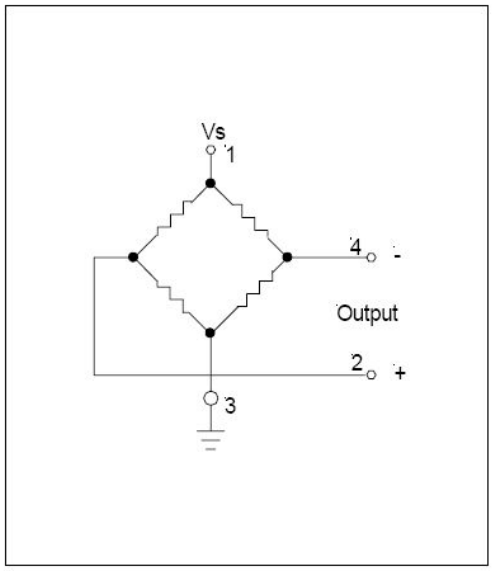
port 2[9]: - polyetherimide
- fluor-silicone
- silicon

SPECIFICATIONS

Maximum ratings

Supply voltage	16 V
Temperature limits	
Storage	-55 to +100°C
Operating	-40 to +85°C
Lead temperature (10 sec. soldering)	300°C
Humidity limits	0...100 %RH
Vibration (MIL-STD-202, Meth. 213)	150 g half sine 11 msec.
Mechanical shock (qualification tested)	150 g
Proof pressure[1]	
all 50, 100 and 250 mbar devices	1.4 bar
all 1 bar devices	3 bar
all 2 bar devices	4 bar
all 5 bar devices	12 bar
all 10 and 16 bar devices	35 bar

ELECTRICAL CONNECTION



SENSORTECHNICS www.sensortechnics.com

ANNEXE B

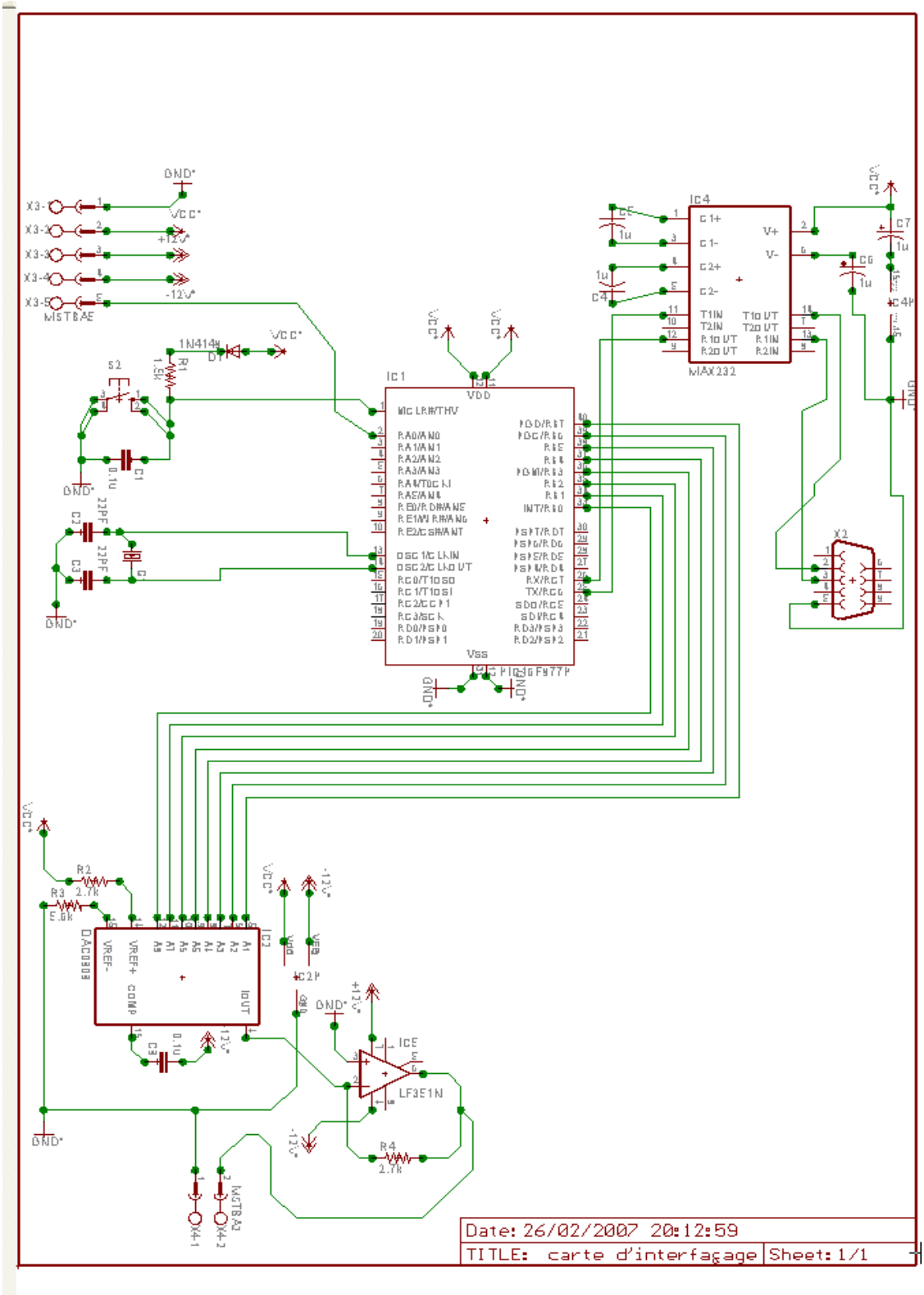


Schéma de la carte à microcontrôleur PIC 16F877

ANNEXE C

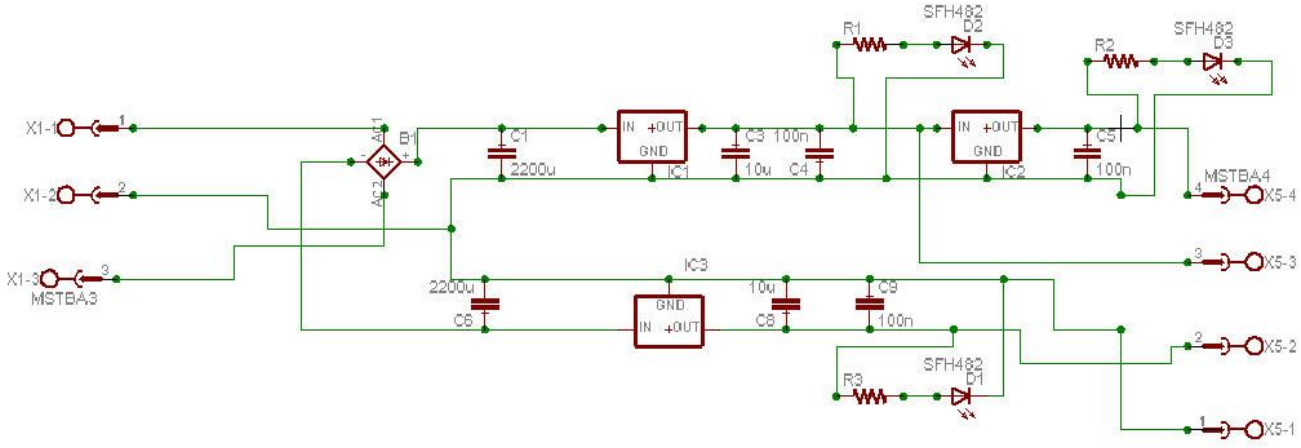


Schéma de la carte d'alimentation stabilisé

ANNEXE D

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	13	14	30	I	ST/CMOS(4)	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	14	15	31	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLK-OUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
$\overline{MCLR}$/VPP/THV	1	2	18	I/P	ST	Master clear (reset) input or programming voltage input or high voltage test mode control. This pin is an active low reset to the device.
						PORTA is a bi-directional I/O port.
RA0/AN0	2	3	19	I/O	TTL	RA0 can also be analog input0
RA1/AN1	3	4	20	I/O	TTL	RA1 can also be analog input1
RA2/AN2/VREF-	4	5	21	I/O	TTL	RA2 can also be analog input2 or negative analog reference voltage
RA3/AN3/VREF+	5	6	22	I/O	TTL	RA3 can also be analog input3 or positive analog reference voltage
RA4/T0CKI	6	7	23	I/O	ST	RA4 can also be the clock input to the Timer0 timer/ counter. Output is open drain type.
RA5/$\overline{SS}$/AN4	7	8	24	I/O	TTL	RA5 can also be analog input4 or the slave select for the synchronous serial port.
						PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.
RB0/INT	33	36	8	I/O	TTL/ST(1)	RB0 can also be the external interrupt pin.
RB1	34	37	9	I/O	TTL	
RB2	35	38	10	I/O	TTL	
RB3/PGM	36	39	11	I/O	TTL	RB3 can also be the low voltage programming input
RB4	37	41	14	I/O	TTL	Interrupt on change pin.
RB5	38	42	15	I/O	TTL	Interrupt on change pin.
RB6/PGC	39	43	16	I/O	TTL/ST(2)	Interrupt on change pin or In-Circuit Debugger pin. Serial programming clock.
RB7/PGD	40	44	17	I/O	TTL/ST(2)	Interrupt on change pin or In-Circuit Debugger pin. Serial programming data.
						PORTC is a bi-directional I/O port.
RC0/T1OSO/T1CKI	15	16	32	I/O	ST	RC0 can also be the Timer1 oscillator output or a Timer1 clock input.
RC1/T1OSI/CCP2	16	18	35	I/O	ST	RC1 can also be the Timer1 oscillator input or Capture2 input/Compare2 output/PWM2 output.
RC2/CCP1	17	19	36	I/O	ST	RC2 can also be the Capture1 input/Compare1 output/ PWM1 output.
RC3/SCK/SCL	18	20	37	I/O	ST	RC3 can also be the synchronous serial clock input/output for both SPI and I^2C modes.
RC4/SDI/SDA	23	25	42	I/O	ST	RC4 can also be the SPI Data In (SPI mode) or data I/O (I^2C mode).
RC5/SDO	24	26	43	I/O	ST	RC5 can also be the SPI Data Out (SPI mode).
RC6/TX/CK	25	27	44	I/O	ST	RC6 can also be the USART Asynchronous Transmit or Synchronous Clock.
RC7/RX/DT	26	29	1	I/O	ST	RC7 can also be the USART Asynchronous Receive or Synchronous Data.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as an external interrupt.
2: This buffer is a Schmitt Trigger input when used in serial programming mode.
3: This buffer is a Schmitt Trigger input when configured as general purpose I/O and a TTL input when used in the Parallel Slave Port mode (for interfacing to a microprocessor bus).
4: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
						PORTD is a bi-directional I/O port or parallel slave port when interfacing to a microprocessor bus.
RD0/PSP0	19	21	38	I/O	ST/TTL(3)	
RD1/PSP1	20	22	39	I/O	ST/TTL(3)	
RD2/PSP2	21	23	40	I/O	ST/TTL(3)	
RD3/PSP3	22	24	41	I/O	ST/TTL(3)	
RD4/PSP4	27	30	2	I/O	ST/TTL(3)	
RD5/PSP5	28	31	3	I/O	ST/TTL(3)	
RD6/PSP6	29	32	4	I/O	ST/TTL(3)	
RD7/PSP7	30	33	5	I/O	ST/TTL(3)	
						PORTE is a bi-directional I/O port.
RE0/$\overline{RD}$/AN5	8	9	25	I/O	ST/TTL(3)	RE0 can also be read control for the parallel slave port, or analog input5.
RE1/$\overline{WR}$/AN6	9	10	26	I/O	ST/TTL(3)	RE1 can also be write control for the parallel slave port, or analog input6.
RE2/$\overline{CS}$/AN7	10	11	27	I/O	ST/TTL(3)	RE2 can also be select control for the parallel slave port, or analog input7.
VSS	12,31	13,34	6,29	P	—	Ground reference for logic and I/O pins.
VDD	11,32	12,35	7,28	P	—	Positive supply for logic and I/O pins.
NC	—	1,17,28, 40	12,13, 33,34		—	These pins are not internally connected. These pins should be left unconnected.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as an external interrupt.
2: This buffer is a Schmitt Trigger input when used in serial programming mode.
3: This buffer is a Schmitt Trigger input when configured as general purpose I/O and a TTL input when used in the Parallel Slave Port mode (for interfacing to a microprocessor bus).
4: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

Description de pic 16F877

ANNEXE D

MICROCHIP

PIC16F87X

28/40-Pin 8-Bit CMOS FLASH Microcontrollers

Devices Included in this Data Sheet:

- PIC16F873
- PIC16F874
- PIC16F876
- PIC16F877

Microcontroller Core Features:

- High performance RISC CPU
- Only 35 single word instructions to learn
- All single cycle instructions except for program branches which are two cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- Up to 8K x 14 words of FLASH Program Memory,
  Up to 368 x 8 bytes of Data Memory (RAM)
  Up to 256 x 8 bytes of EEPROM Data Memory
- Pinout compatible to the PIC16C73B/74B/76/77
- Interrupt capability (up to 14 sources)
- Eight level deep hardware stack
- Direct, Indirect and relative addressing modes
- Power-on Reset (POR)
- Power-up Timer (PWRT) and
  Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code protection
- Power saving SLEEP mode
- Selectable oscillator options
- Low power, high speed CMOS FLASH/EEPROM technology
- Fully static design
- In-Circuit Serial Programming™ (ICSP) via two pins
- Single 5V In-Circuit Serial Programming capability
- In-Circuit Debugging via two pins
- Processor read/write access to program memory
- Wide operating voltage range: 2.0V to 5.5V
- High Sink/Source Current: 25 mA
- Commercial, Industrial and Extended temperature ranges
- Low-power consumption:
  - < 0.6 mA typical @ 3V, 4 MHz
  - 20 µA typical @ 3V, 32 kHz
  - < 1 µA typical standby current

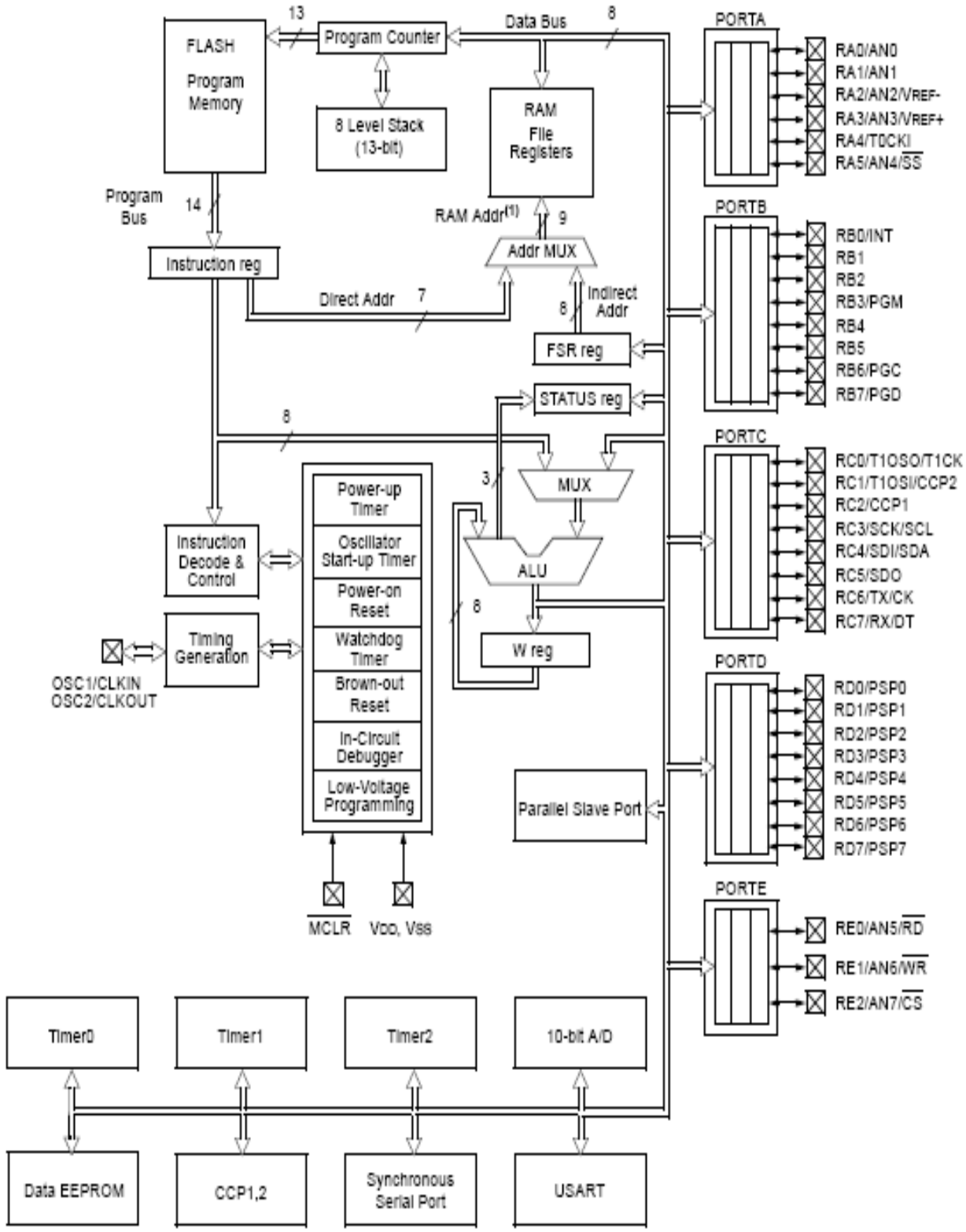
Pin Diagram

PDIP

Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during SLEEP via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- 10-bit multi-channel Analog-to-Digital converter
- Synchronous Serial Port (SSP) with SPI™ (Master mode) and I²C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) 8-bits wide, with external RD, WR and CS controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)

Datasheet du Pic 16F87X



Organisation interne du Pic 16F877

ANNEXE E

National Semiconductor

January 1995

DAC0808/DAC0807/DAC0806 8-Bit D/A Converters

General Description

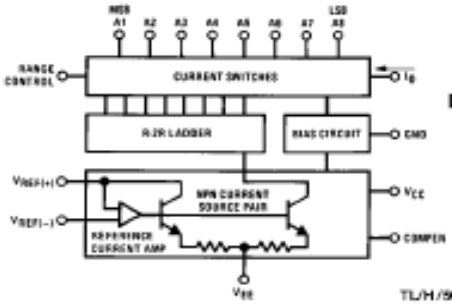
The DAC0808 series is an 8-bit monolithic digital-to-analog converter (DAC) featuring a full scale output current settling time of 150 ns while dissipating only 33 mW with ±5V supplies. No reference current (I_{REF}) trimming is required for most applications since the full scale output current is typically ±1 LSB of 255 I_{REF}/ 256. Relative accuracies of better than ±0.19% assure 8-bit monotonicity and linearity while zero level output current of less than 4 μA provides 8-bit zero accuracy for $I_{REF} \geq 2$ mA. The power supply currents of the DAC0808 series are independent of bit codes, and exhibits essentially constant device characteristics over the entire supply voltage range.

The DAC0808 will interface directly with popular TTL, DTL or CMOS logic levels, and is a direct replacement for the MC1508/MC1408. For higher speed applications, see DAC0800 data sheet.

Features

- Relative accuracy: ±0.19% error maximum (DAC0808)
- Full scale current match: ±1 LSB typ
- 7 and 6-bit accuracy available (DAC0807, DAC0806)
- Fast settling time: 150 ns typ
- Noninverting digital inputs are TTL and CMOS compatible
- High speed multiplying input slew rate: 8 mA/μs
- Power supply voltage range: ±4.5V to ±18V
- Low power consumption: 33 mW @ ±5V

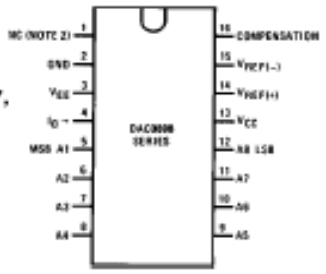
Block and Connection Diagrams



TL/H/5687–1

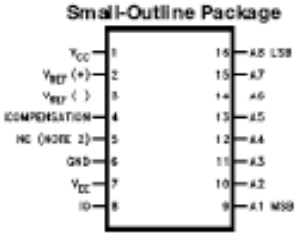
Order Number DAC0808, DAC0807, or DAC0806
See NS Package Number J16A, M16A or N16A

Dual-In-Line Package



TL/H/5687–2

Small-Outline Package



TL/H/5687–13

Top View

Ordering Information

ACCURACY	OPERATING TEMPERATURE RANGE	ORDER NUMBERS				
		J PACKAGE (J16A)*		N PACKAGE (N16A)*		SO PACKAGE (M16A)
7-bit	$0°C \leq T_A \leq +75°C$	DAC0807LCJ	MC1408L7	DAC0807LCN	MC1408P7	DAC0807LCM
6-bit	$0°C \leq T_A \leq +75°C$	DAC0806LCJ	MC1408L6	DAC0806LCN	MC1408P6	DAC0806LCM
				DAC0808LCN	MC1408P8	DAC0808LCM

*Note. Devices may be ordered by using either order number.

RRD-B30M115/Printed in U. S. A.

Datasheet du convertisseur numérique analogique DAC0808

ANNEXE F

National Semiconductor

December 1995

LF351 Wide Bandwidth JFET Input Operational Amplifier

General Description

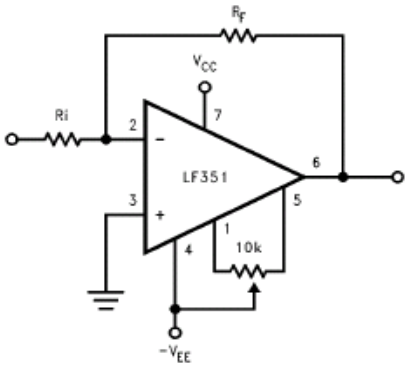
The LF351 is a low cost high speed JFET input operational amplifier with an internally trimmed input offset voltage (BI-FET II™ technology). The device requires a low supply current and yet maintains a large gain bandwidth product and a fast slew rate. In addition, well matched high voltage JFET input devices provide very low input bias and offset currents. The LF351 is pin compatible with the standard LM741 and uses the same offset voltage adjustment circuitry. This feature allows designers to immediately upgrade the overall performance of existing LM741 designs.

The LF351 may be used in applications such as high speed integrators, fast D/A converters, sample-and-hold circuits and many other circuits requiring low input offset voltage, low input bias current, high input impedance, high slew rate and wide bandwidth. The device has low noise and offset voltage drift, but for applications where these requirements are critical, the LF356 is recommended. If maximum supply current is important, however, the LF351 is the better choice.

Features

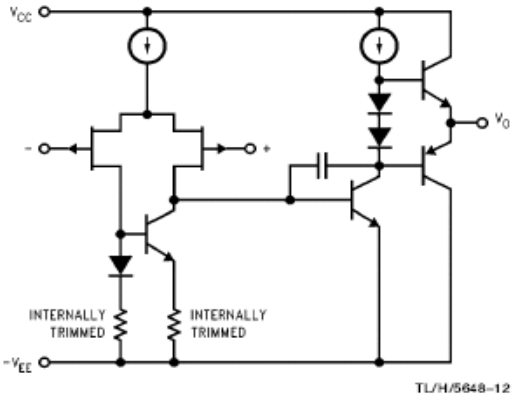
■ Internally trimmed offset voltage	10 mV
■ Low input bias current	50 pA
■ Low input noise voltage	25 nV/$\sqrt{Hz}$
■ Low input noise current	0.01 pA/$\sqrt{Hz}$
■ Wide gain bandwidth	4 MHz
■ High slew rate	13 V/μs
■ Low supply current	1.8 mA
■ High input impedance	$10^{12}\Omega$
■ Low total harmonic distortion $A_V = 10$, $R_L = 10k$, $V_O = 20$ Vp-p, BW = 20 Hz–20 kHz	<0.02%
■ Low 1/f noise corner	50 Hz
■ Fast settling time to 0.01%	2 μs

Typical Connection



TL/H/5648–11

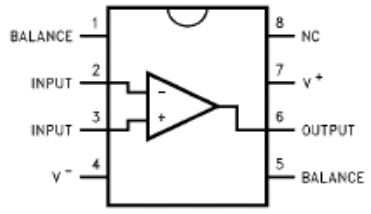
Simplified Schematic



TL/H/5648–12

Connection Diagrams

Dual-In-Line Package

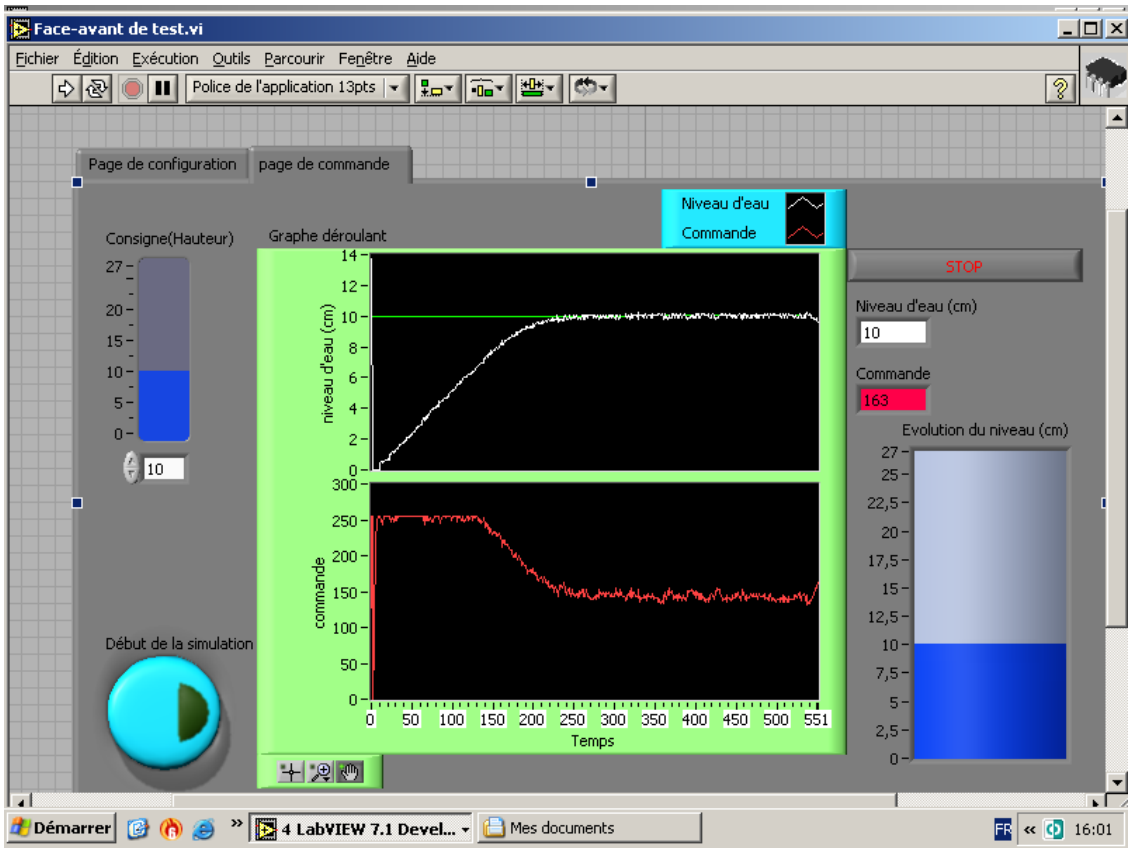


TL/H/5648–13

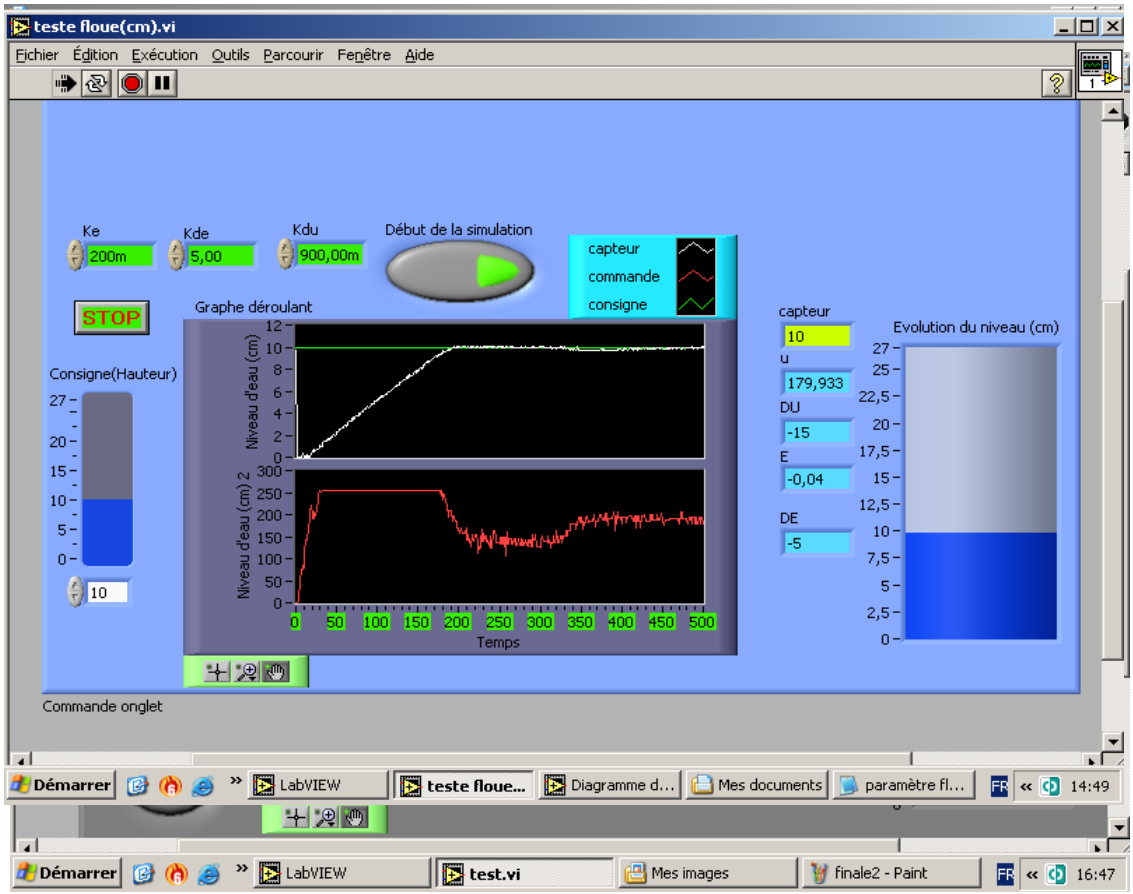
Order Number LF351M or LF351N
See NS Package Number M08A or N08E

Datasheet de l'amplificateur opérationnel LF 351

ANNEXE G


Face-avant de test.vi
Fichier Édition Exécution Outils Parcourir Fenêtre Aide
Police de l'application 13pts
Page de configuration
page de commande
Consigne(Hauteur)
Graphe déroulant
Niveau d'eau
Commande
STOP
Niveau d'eau (cm)
10
Commande
163
Evolution du niveau (cm)
niveau d'eau (cm)
commande
Temps
Début de la simulation
Démarrer
4 LabVIEW 7.1 Devel...
Mes documents
16:01

ANNEXE H


teste floue(cm).vi
Fichier Édition Exécution Outils Parcourir Fenêtre Aide
Ke
200m
Kde
5,00
Kdu
900,00m
Début de la simulation
capteur
commande
consigne
STOP
Graphe déroulant
Consigne(Hauteur)
capteur
10
u
179,933
DU
-15
E
-0,04
DE
-5
Evolution du niveau (cm)
Niveau d'eau (cm)
Niveau d'eau (cm) 2
Temps
Commande onglet
Démarrer
LabVIEW
teste floue...
Diagramme d...
Mes documents
paramètre fl...
14:49
Démarrer
LabVIEW
test.vi
Mes images
finale2 - Paint
16:47