République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Polytechnique d'Alger

Département d'Automatique

Projet de Fin d'Etudes

En vue de l'obtention des diplômes

D'Ingénieur d'Etat et du Master2 en Automatique

Thème :

Commande à base d'une DSP de la nouvelle structure du Robot SCARA à 3ddl

---

Proposé et encadré par :

- Mr. H.CHEKIREB
- Mr. O.STIHI

Réalisé par :

- Mr. AITEUR Imad Eddine
- Mr. SADELLI Lounis

Juin 2012

ملخّص:

إن الهدف من هذا العمل هو تحسين الذراع الآلية من نوع سكارا المتواجدة على مستوى مخبر التحكم في الآليات؛ لهذا الغرض قمنا بتشخيص العيوب و تحسينها بالاستعانة بمهندسين ميكانيكيين . قمنا بعد ذلك بحساب النماذج الهندسية و الحركية لهذا الذراع و بدراسة عن واجهة التحكم من نوع UMAC DELTA TAU المخصصة للتحكم في الآليات ؛ كل هذا العمل كان بغرض برمجة مختلف المهام بعد تخطيطها و التي يمكن لهذه الذراع القيام بها مثل: رسم المسارات؛ كتابة سلسلة الحروف الكتابة على لوحة المفاتيح و اختيار و وضع الأشياء

كلمات مفتاح : سكارا ؛تحسين الذراع؛ واجهة التحكم؛تخطيط المهام ؛برمجة المهام

Résumé :

Le travail effectué, dans le cadre de ce mémoire, a pour objectif l'amélioration de la structure du robot SCARA existant au sein LCP de l'ENP et la planification de différentes tâches pour ce robot. Dans ce but, nous avons tout d'abord relevé les différents défauts affectant cette structure et isolé la source de chacun de ces défauts. Une fois ces modifications ont été apportées à cette structure, nous avons alors déterminé, pour la nouvelle structure du robot, ses modèles géométriques et cinématiques. Pour la commande des articulations du robot, nous avons exploité la carte DSP (UMAC de la firme DELTA TAU), dédiée spécialement à la commande d'axes, et son logiciel de programmation PEWIN32 Pro. Ceci nous a permis de programmer différentes tâches à exécuter par ce robot entre autres : le tracé des trajectoires, l'écriture d'une chaine de caractère, la saisie sur un clavier et la tâche « Pick and Place ». Aussi, ce robot est parvenu à effectuer ces différentes taches avec des performances jugées satisfaisantes dans le contexte de ce travail.

Mots clés : Robot SCARA, améliorations, UMAC Delta TAU, PEWIN 32 Pro, planification de tâches, programmation de tâches.

Summary:

The present work focuses on improving the existing structure of the SCARA robot in the LCP laboratory of ENP. For this, we started with a diagnosis of robot's defects using the expertise of mechanical engineers in the field; this preliminary work was followed by changes in the structure and calculation of geometric and kinematic models of the new structure. To program the new robot, we did a study about DSP boards and especially the UMAC of the firm DELTA TAU, especially dedicated to robot control, and programming software PEWIN32 Pro. This allowed us to program various tasks the robot can perform which are: set out trajectory, writing a character string, type on a keyboard and pick and place task.

Key words: SCARA Robot, improvement, UMAC Delta tau, PEWIN 32 Pro, tasks planning, programming tasks.

Remerciements

On tient à remercier en premier lieu « ALLAH » le tout puissant, qui nous a donné la force, le courage et la volonté pour mener à bien ce modeste travail.

Nous exprimons notre profonde gratitude et nos vifs remerciements à **Mr. H.CHEKIREB** pour son aide inestimable, ses conseils judicieux et son encouragement et qui nous a orientés tout le long de ce travail.

Nos vifs remerciements à **Mr. O.STIHI** qui nous a dirigés dans tous les travaux techniques effectués au sein du Laboratoire de Robotique.

Nous tenons à remercier particulièrement, **Mr. R. ILLOUL,** pour son aide précieux dans la concrétisation de ce travail.

Nos remerciements vont aussi à tous les enseignants de l'Ecole Nationale Polytechnique, et spécialement à ceux du Département d'Automatique.

Nous remercions également les membres de jury qui nous ont fait l'honneur d'examiner notre travail.

Finalement, nous remercions toute personne qui nous a soutenus de pré ou de loin tout au long de notre parcours pour la réalisation de ce travail.

Dédicace

Je dédie ce travail à lumière de mes yeux, à l'être que j'aime et que je chéris plus que tout au monde : à maman.

Je dédie aussi ce travail à mon père, A mes sœurs Narimene et Manel que j'adore beaucoup, ainsi mes deux frères Dhirar et Oussama

Je ne saurais oublier mes amis (es), et en particulier Salim, Halim, mon binôme Lounis et toute la promotion d'Automatique (2009-2012). Je leur souhaite à tous de très beaux succès dans la vie.

Aussi, je cite tous les professeurs et enseignants que j'ai eus depuis mes premières années du primaire jusqu'au jour d'aujourd'hui. Je ne leur exprimerais jamais assez mon estime, mes respects et ma gratitude.

AITEUR Imad Eddine

Dédicace

Je dédie ce travail qui est le fruit de cinq années de travail acharné

A mes parents pour qui, aucun mot n'est assez fort pour leur exprimer la reconnaissance sincère, que je leur porte pour la richesse de leurs enseignements et leurs sacrifices, merci du fond du cœur.

A mes chers frères et sœurs

A ma belle-sœur et mon cher neveu Younes

A tous mes cousins et cousines

A tous mes amis d'enfance

A mes chers amis Louenas, Sofiane et Faouzi

A tous les gens qui m'ont soutenue

A mon binôme avec qui j'ai partagé tant de moments

Et à toute la promotion de l'automatique 2012

Lounis

Table des Matières

CHAPITRE 3 : LE REGULATEUR PID IMPLEMENTE

CHAPITRE 4 : PLANIFICATION DES TACHES ET IMPLEMENTATION DES PROGRAMMES

Liste des Figures

CHAPITRE 1

CHAPITRE 2

CHAPITRE 3

CHAPITRE 4

ANNEXE A

Liste des Tableaux

Introduction générale

Les robots sont largement utilisés dans le domaine industriel afin d'augmenter la productivité et pour réaliser les tâches pénibles et dangereuses pour l'homme. A moyen terme, ils tendent à investir de plus en plus notre vie quotidienne.

De ce fait, la robotique constitue un marché porteur et très lucratif. A l'avenir, les experts annoncent une forte croissance du nombre de robots notamment dans le domaine de la robotique de service, l'assistance aux personnes handicapées, la robotique médicale ou encore pour les loisirs. Les robots manipulateurs appelés SCARA (Selective Compliant Assembly Robot Arm) ont fait leur apparition dans les années 1970, ils sont très répandus dans l'industrie, notamment pour les tâches de palettisation.

L'essor de la robotique a révélé de nouvelles techniques de commande et d'asservissement très sophistiquée. Ceci est rendu possible grâce à l'évolution de l'informatique et du traitement de signal de l'information. En effet, la commande numérique est de plus en plus utilisée et des firmes spécialisées dans ce domaine proposent des cartes très variées de commande telle que l'interface Turbo UMAC de la firme américaine DELTA TAU.

Cette interface est un contrôleur de mouvement puissant et flexible, permettant d'une part une bonne performance de la commande des moteurs grâce à leur vitesse de traitement de l'information et d'autre part une meilleure interactivité avec l'utilisateur. Ceci est dû à la disponibilité d'un ensemble de logiciels dédié à cette interface.

L'objectif principal de notre travail consiste à effectuer certaines modifications du robot SCARA existant au sein du LCP dans le sens d'améliorer cette structure. Ainsi, ce travail est présenté dans quatre chapitres.

Dans le **premier chapitre**, est présenté en premier lieu la structure existante et puis nous développons les différentes modifications apportées à cette structure afin d'apporter les améliorations escomptées. Par la suite, nous déterminons le modèle géométrique et le modèle cinématique direct et inverse correspondant à cette nouvelle structure.

Dans le **deuxième chapitre**, nous présentons d'une part le fonctionnement de l'interface UMAC, qui sera utilisée par la suite dans la commande de notre robot et d'autre part, le logiciel de programmation PEWIN32 PRO conçu par la même firme américaine DELTA TAU.

Le troisième chapitre est consacré aux explications relatives à la méthode d'asservissement des moteurs, au régulateur PID implémenté au niveau de la carte DSP ainsi qu'à la méthode de détermination des paramètres de régulation.

Le chapitre quatre est le plus important de notre travail. C'est dans ce chapitre que sont planifiées plusieurs tâches à exécuter par notre robot. Pour chacune d'elle, le programme correspondant est développé, puis implémenté et enfin exécuté en exploitant le logiciel PEWIN 32 déjà présenté au chapitre 2. Dans ce chapitre, on trouvera aussi, les détails relatifs à l'application des PLC. Ainsi que leur utilisation pour distinguer ou alterner entre différentes tâches selon le besoin.

CHAPITRE 1
PRESENTATION ET MODELISATION DU ROBOT SCARA

I.1 Introduction

Dans l'industrie on trouve plusieurs types des robots, on s'intéresse dans ce chapitre aux robots manipulateur de type série qui sont utilisés dans l'industrie pour augmenter la production, avoir de bonne qualité de produit ainsi que pour automatiser les tâches effectuer pour fabriquer un produit. Ces robots manipulateurs se composent d'un ensemble de corps rigides reliés entre eux par des articulations rotoïdes(R) ou prismatique (P). Des moteurs sont attachés aux articulations pour pouvoir commander la position de chaque articulation et cela dans le but d'avoir le mouvement désiré pour effectuer la tâche attribuer au robot manipulateur. A la fin de la dernière articulation, un outil est attaché pour accomplir une tâche bien précise.

I.2 Les robots manipulateurs séries

Les robots manipulateurs séries se sont des chaînes cinématiques poly articulées ouvertes. Les structures les plus utilisées dans l'industrie sont [1] :

- Les bras manipulateurs de type anthropomorphe, à six liaisons rotoides d'usage général.
- Les bras manipulateurs de type SCARA, qui sont fréquemment utilisés dans les cellules de production pour les opérations de palettisation.

Les figures1.1et 1.2 montrent un exemple de bras manipulateurs du type anthropomorphe et du type SCARA

Figure 1.1: Bras manipulateur de type anthropomorphe

Figure 1.2 : Bras manipulateur de type SCARA

Ils sont utilisés dans l'industrie pour accomplir le plus souvent les tâches suivantes : le chargement et le déchargement, l'assemblage des pièces et la peinture dans l'industrie d'automobile, le soudage, le collage et l'ébavurage…etc.

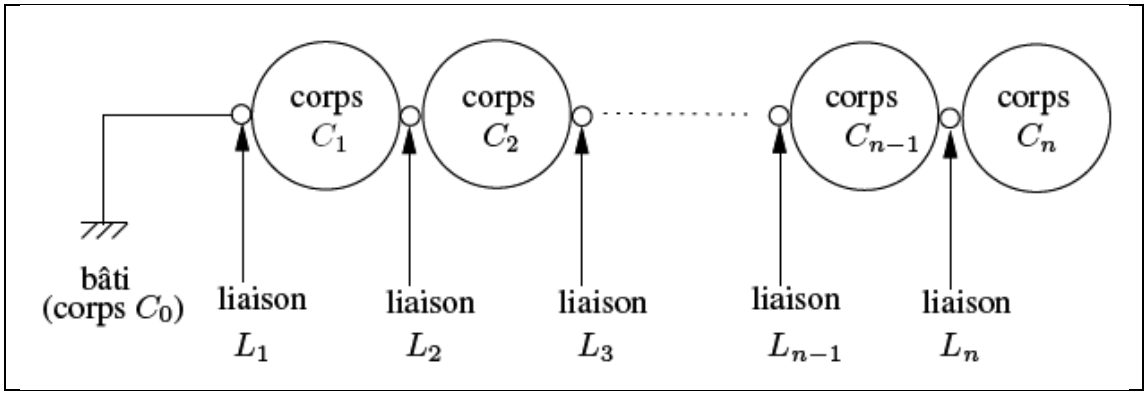


Figure 1.3 : Chaîne cinématique d'un bras manipulateurs série

I.3 Modélisation d'un robot de type SCARA RRRP

I.3.1 Les transformations homogènes

Notre but dans ce travail est de réaliser un robot de type SCARA à 4 degré de liberté ayant trois articulations rotoïdes et une prismatique. La disposition des repères est représentée à la figure 1.4 :

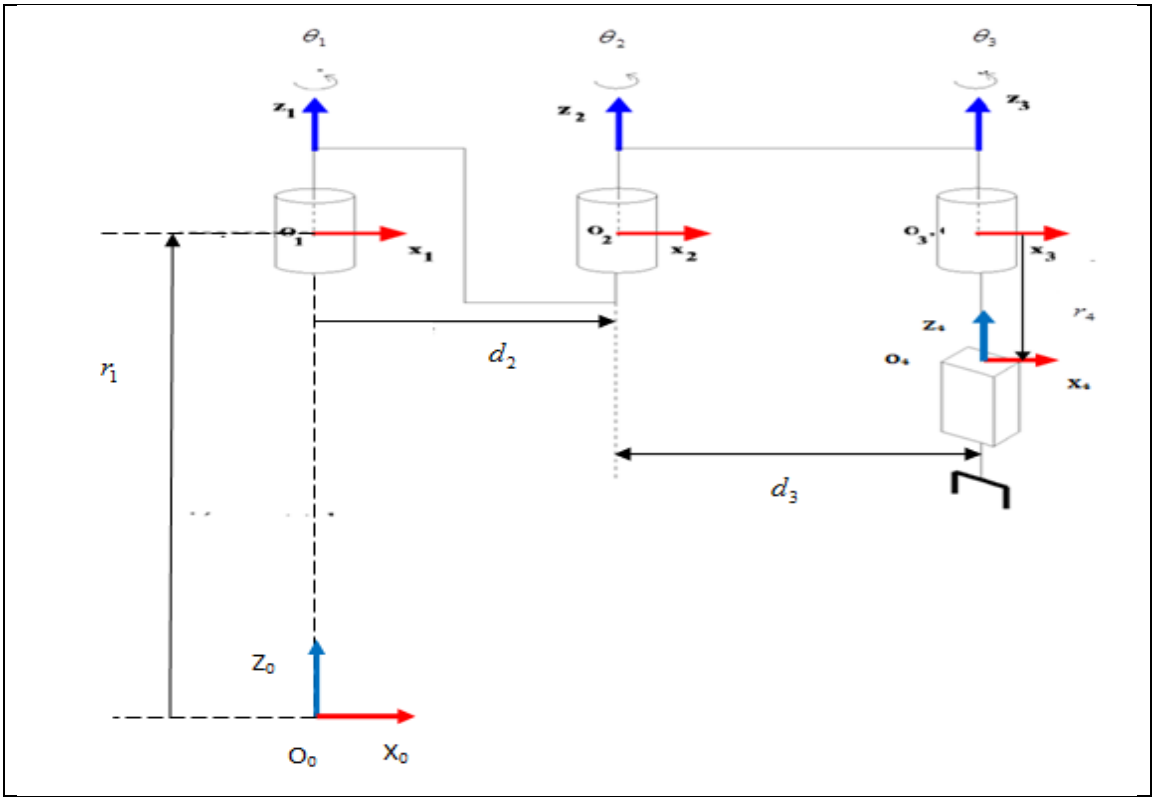


Figure 1.4 : Bras manipulateur à 4degrés de liberté

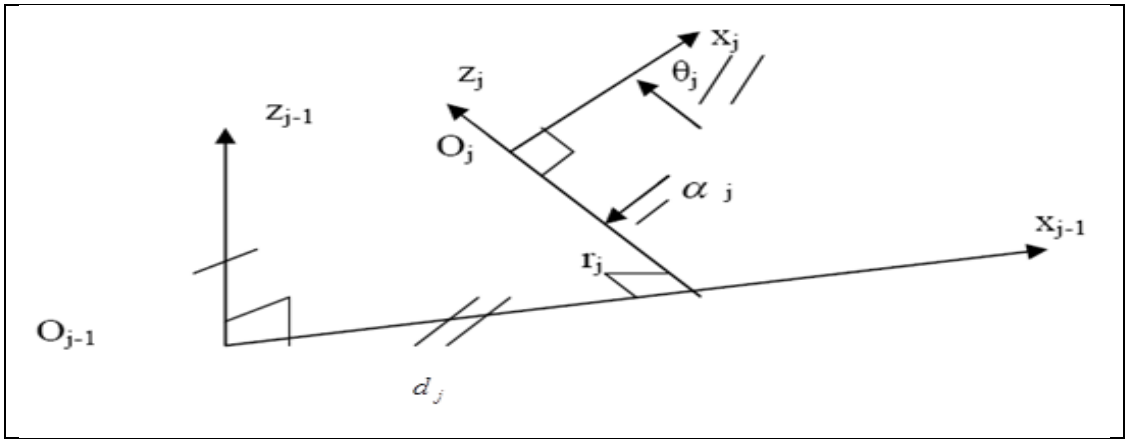


Figure 1.5 : Représentation des paramètres D-H

La détermination de la matrice de transformation entre le repère R_0 et le repère R_4 est obtenue en utilisant les paramètres de Denavit-Hartenberg (D-H).

Les paramètres (D-H) notés $\sigma_i, d_i, \alpha_i, \theta_i, r_i$ sont définies comme suit (Figure1.5):

σ_i : Egal à zéro ou bien un pour une articulation rotoïde ou bien prismatique respectivement,

d_i : distance entre z_{i-1} et z_i le long de x_{i-1},

α_i : angle entre les axes z_{i-1} et z_i correspondant à une rotation autour de x_{i-1},

θ_i : angle entre les axes x_{i-1} et x_i correspondant à une rotation autour de z_i,

r_i : distance entre x_{i-1} et x_i le long de z_i,

Pour le cas du robot SCARA de la figure 1.4, ces paramètres sont donnés au tableau 1.1 où $\theta_1, \theta_2, \theta_3$ et r_4 représentent les variables articulaires et d_2, d_3 et r_1 sont des coefficients constants.

i	σ_i	d_i	α_i	θ_i	r_i
1	0	0	0	θ_1	r_1
2	0	d_2	0	θ_2	0
3	0	d_3	0	θ_3	0
4	1	0	0	0	r_4

Tableau 1.1 : paramètres D-H du robot SCARA RRRP

Dans le cas du robot SCARA utilisé dans notre application, les paramètres constants ont été mesurés et nous avons obtenus les valeurs suivantes:

$$r_1 = 385mm \quad d_2 = 186.5mm \quad d_3 = 172.5mm \quad r_4 = \begin{cases} r_{4\min} = 365mm \\ r_{4\max} = 415mm \end{cases}$$

On a aussi les relations suivantes :

La variable articulaire en général est donnée par la relation (1.1):

$$q_i = \bar{\sigma}_i \theta_i + \sigma_i r_i \qquad (1.1)$$

On note par

$$\cos(\beta_i) = c\beta_i$$
$$\sin(\beta_i) = s\beta_i$$

La transformation entre deux repères adjacents R_{i-1} et R_i est donnée par la relation (1.2) :

$$T_i^{i-1} = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & d_i \\ c\alpha_i s\theta_i & c\alpha_i c\theta_i & -s\alpha_i & -r_i s\alpha_i \\ s\alpha_i s\theta_i & s\alpha_i c\theta_i & c\alpha_i & r_i c\alpha_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (1.2)$$

Ainsi, la matrice T_i^{i-1} définit le repère R_i dans le repère R_{i-1}

En utilisant la relation (1.2), les transformations homogènes T_1^0 T_2^1, T_3^2 , T_4^3 du robot RRRP sont telles que :

$$T_1^0 = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & r_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad T_2^1 = \begin{bmatrix} c_2 & -s_2 & 0 & d_2 \\ s_2 & c_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_3^2 = \begin{bmatrix} c_3 & -s_3 & 0 & d_3 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad T_4^3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & r_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

I.3.2 Le modèle géométrique directe (MGD)

Le MGD d'un bras manipulateur donne la position de son organe terminal(OT) en fonction de sa configuration [1]:

$$f : N \to M$$
$$q \mapsto X = f(q)$$

Avec :

- N : espace de configuration de dimension n appelée indice de mobilité,
- M : espace opérationnelle de dimension m,
- q : coordonnées généralisés de n composantes,
- X : coordonnées opérationnelles de m composantes.

Pour le calcul de la matrice de transformation entre le repère R_0 et R_4 on introduit les notations suivantes :

$$\cos(\theta_i) = c_i \quad \cos(\theta_i + \theta_j) = c_{ij} \quad \cos(\theta_i + \theta_j + \theta_k) = c_{ijk}$$
$$\sin(\theta_i) = s_i \quad \sin(\theta_i + \theta_j) = s_{ij} \quad \sin(\theta_i + \theta_j + \theta_k) = s_{ijk}$$

$$T_4^2 = T_3^2 T_4^3 = \begin{bmatrix} c_3 & -s_3 & 0 & d_3 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & r_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_4^2 = \begin{bmatrix} c_3 & -s_3 & 0 & d_3 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & r_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_4^1 = T_2^1 T_4^2 = \begin{bmatrix} c_2 & -s_2 & 0 & d_2 \\ s_2 & c_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_3 & -s_3 & 0 & d_3 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & r_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_4^1 = \begin{bmatrix} c_{23} & -s_{23} & 0 & d_2 + d_3 c_2 \\ s_{23} & c_{23} & 0 & d_3 s_2 \\ 0 & 0 & 1 & r_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_4^0 = T_1^0 T_4^1 = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & r_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_{23} & -s_{23} & 0 & d_3 c_2 + d_2 \\ s_{23} & c_{23} & 0 & d_3 s_2 \\ 0 & 0 & 1 & r_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_4^0 = \begin{bmatrix} c_{123} & -s_{123} & 0 & d_3 c_{12} + d_2 c_1 \\ s_{123} & c_{123} & 0 & d_3 s_{12} + d_2 s_1 \\ 0 & 0 & 1 & r_4 + r_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

On définit le vecteur X de l'espace opérationnel comme étant la position dans l'espace cartésien 3D et l'orientation φ définie par l'angle entre les axes x_4 et x_0.

$$X = \begin{bmatrix} x & y & z & \varphi \end{bmatrix}^T$$

Alors on obtient le modèle géométrique donné par le système (1.3)

$$\begin{cases} x = d_2 c_1 + d_3 c_{12} \\ y = d_2 s_1 + d_3 s_{12} \\ z = r_1 + r_4 \\ \varphi = \theta_1 + \theta_2 + \theta_3 \end{cases} \quad (1.3)$$

I.3.3 Le modèle géométrique inverse (MGI)

Le MGI d'un bras manipulateur donne la ou les configurations correspondantes à une position donnée de l'organe terminal [1]:

$$f^{-1} : M \to N$$
$$X \mapsto q = f^{-1}(X)$$

Supposons que X soit une situation accessible par le bras manipulateur c.à.d. X appartient à l'espace de travail du robot, alors on a :

- Si $n < m$, il n'existe pas de solution au MGI
- Si $n = m$, il existe un nombre fini de solutions en dehors de certaines configurations, appelées configurations singulières.
- Si $n > m$, il existe une infinité de solutions au MGI

Dans notre cas on a $n = m = 4$

On considère le cas où le repère de l'outil est disposé tel que la transformation désirée $\left(T_4^0\right)^*$ soit de la forme :

$$\left(T_4^0\right)^* = \begin{bmatrix} S_x & N_x & 0 & P_x \\ S_y & N_y & 0 & P_y \\ 0 & 0 & 1 & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Il en découle alors :

$$\begin{cases} P_x = d_2 c_1 + d_3 c_{12} & (1.4) \\ P_y = d_2 s_1 + d_3 s_{12} & (1.5) \\ P_z = r_1 + r_4 & (1.6) \end{cases}$$

La relation (1.6) conduit à obtenir directement r_4

$r_4 = P_z - r_1$

On élève au carré les relations (1.4) et (1.5) puis on les ajoute membre à membre d'où on obtient :

$$d_3^2 + d_2^2 + 2 d_2 d_3 c_2 = P_x^2 + P_y^2 \qquad (1.7)$$

La relation (1.7) permet d'obtenir c_2

$$c_2 = \frac{P_x^2 + P_y^2 - d_2^2 - d_3^2}{2 d_2 d_3} \qquad (1.8)$$

$$s_2 = \pm\sqrt{1 - c_2^2}$$

Sachant que :

$$\cos(\theta) = a \qquad a \in [-1 \quad 1] \qquad \Rightarrow \theta = \pm Ar\cos(a)$$

De la relation (1.8) on obtient θ_2

$$\theta_2 = \pm Ar\cos\left(\frac{P_x^2 + P_y^2 - d_2^2 - d_3^2}{2 d_2 d_3}\right) \qquad \text{Avec} \qquad -1 \le \frac{P_x^2 + P_y^2 - d_2^2 - d_3^2}{2 d_2 d_3} \le 1$$

$$\theta_2 = ATAN2(s_2, c_2)$$

Pour calculer θ_1 on utilise les relations (1.4) et (1.5) réécrites sous la forme suivante :

$$\begin{cases} (d_2 + d_3 c_2) c_1 - (d_3 s_2) s_1 = P_x \\ (d_3 s_2) c_1 + (d_2 + d_3 c_2) s_1 = P_y \end{cases}$$

On résout ce système par rapport à c_1 et s_1

On aura alors

$$\begin{bmatrix} c_1 \\ s_1 \end{bmatrix} = \frac{1}{\Delta}\begin{bmatrix} d_2 + d_3 c_2 & d_3 s_2 \\ -d_3 s_2 & d_2 + d_3 c_2 \end{bmatrix}\begin{bmatrix} P_x \\ P_y \end{bmatrix}$$

Avec

$$\Delta = (d_2 + d_3 c_2)^2 + (d_3 s_2)$$
$$\Delta = d_2^2 + d_3^2 + 2 d_2 d_3 c_2$$

On aboutit à la solution :

$$c_1 = \frac{(d_2 + d_3 c_2)P_x + (d_3 s_2)P_y}{d_2^2 + d_3^2 + 2 d_2 d_3 c_2}$$

$$s_1 = \frac{-(d_3 s_2)P_x + (d_2 + d_3 c_2)P_y}{d_2^2 + d_3^2 + 2 d_2 d_3 c_2}$$

Enfin on obtient θ_1 :

$$\theta_1 = \arctan\left(\frac{s_1}{c_1}\right)$$

$$\theta_1 = \arctan\left(\frac{-(d_3 s_2)P_x + (d_2 + d_3 c_2)P_y}{(d_2 + d_3 c_2)P_x + (d_3 s_2)P_y}\right)$$

$$\theta_1 = ATAN2(s_1, c_1)$$

Il reste à déterminer θ_3, à partir de la matrice de rotation on a :

$$S_x = c_{123}$$
$$S_y = s_{123}$$

D'où

$$\theta_1 + \theta_2 + \theta_3 = \arctan\left(\frac{S_y}{S_x}\right)$$
$$\theta_1 + \theta_2 + \theta_3 = ATAN2(S_y, S_x)$$

D'où

$$\theta_3 = \arctan\left(\frac{S_y}{S_x}\right) - (\theta_1 + \theta_2)$$

On obtient alors le MGI donné par le système suivant :

$$\begin{cases} \theta_2 = \pm Ar\cos\left(\dfrac{x^2+y^2-d_2^2-d_3^2}{2d_2d_3}\right) \\ \theta_1 = \arctan\left(\dfrac{-(d_3s_2)x+(d_2+d_3c_2)y}{(d_2+d_3c_2)x+(d_3s_2)y}\right) \\ \theta_3 = \arctan\left(\dfrac{S_y}{S_x}\right)-(\theta_1+\theta_2) \\ r_4 = P_z - r_1 \end{cases}$$

I.3.4 Le modèle cinématique directe (MCD)

Le MCD d'un bras manipulateur donne la relation entre les vitesses opérationnelles et généralisées qui sont $\dot{X}$ et $\dot{q}$ respectivement, [1]:

$$\dot{X} = J(q)\dot{q}$$

La matrice $J(q)$ de dimension $(m\times n)$ est appelée matrice jacobienne de la fonction f, la matrice J est définie comme suit :

$$J : T_qN \to T_xM \qquad J = \frac{\partial f}{\partial q}$$
$$\dot{q} \mapsto \dot{X} = J\dot{q}$$

Où T_qN et T_xM représentent respectivement la tangente à N en q et à M en X

Détermination du modèle cinématique directe :

En calculant la dérivée temporelle des variables opérationnelles du système (1.3), on obtient :

$$\begin{cases} \dot{x} = -d_2s_1\dot{\theta}_1 - d_3s_{12}\dot{\theta}_1 - d_3s_{12}\dot{\theta}_2 \\ \dot{y} = d_2c_1\dot{\theta}_1 + d_3c_{12}\dot{\theta}_1 + d_3c_{12}\dot{\theta}_2 \\ \dot{z} = \dot{r}_4 \\ \dot{\varphi} = \dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3 \end{cases} \Rightarrow \begin{cases} \dot{x} = -y\dot{\theta}_1 - d_3s_{12}\dot{\theta}_2 \\ \dot{y} = x\dot{\theta}_1 + d_3c_{12}\dot{\theta}_2 \\ \dot{z} = \dot{r}_4 \\ \dot{\varphi} = \dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3 \end{cases}$$

D'où le modèle cinématique directe ci-dessous:

$$\dot{X} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} -(d_2 s_1 + d_3 s_{12}) & -d_3 s_{12} & 0 & 0 \\ (d_2 c_1 + d_3 c_{12}) & d_3 c_{12} & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \dot{r}_4 \end{bmatrix}$$

Avec

$$J = \begin{bmatrix} -(d_2 s_1 + d_3 s_{12}) & -d_3 s_{12} & 0 & 0 \\ (d_2 c_1 + d_3 c_{12}) & d_3 c_{12} & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

$$\det(J) = \begin{vmatrix} -(d_2 s_1 + d_3 s_{12}) & -d_3 s_{12} \\ (d_2 c_1 + d_3 c_{12}) & d_3 c_{12} \end{vmatrix} \begin{vmatrix} 0 & 1 \\ 1 & 0 \end{vmatrix} = -d_3 d_2 s_2$$

Les singularités du premier ordre du bras manipulateur représenté sur la figure 1.4 sont données par :

$\det(J) = 0 \Rightarrow \theta_2 = 0 \quad ou \quad \pm\pi$

Donc les singularités correspondent à la situation où la liaison 2 est en extension $(\theta_2 = 0)$ ou replié sur la liaison 1 $(\theta_2 = \pm\pi)$

I.3.5 Le modèle cinématique inverse (MCI)

Le MCI d'un bras manipulateur donne les vitesses généralisées $\dot{q}$ en fonction des vitesses opérationnelles $\dot{X}$, si $n = m$:

$$\dot{q} = J^{-1}\dot{X}$$

Pour le calcul du MCI, il n'y a pas de méthode systématique tout dépend de la structure du robot à modéliser.

Maintenant on va calculer le MCI du bras manipulateur de la figure 1.4.

On a :

$$J=\begin{bmatrix} -(d_2 s_1 + d_3 s_{12}) & -d_3 s_{12} & 0 & 0 \\ (d_2 c_1 + d_3 c_{12}) & d_3 c_{12} & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

Calcul de J^{-1} :

Lorsque la matrice J a la forme suivante :

$$J=\begin{pmatrix} A & 0 \\ B & C \end{pmatrix}$$

Alors

$$J^{-1}=\begin{pmatrix} A^{-1} & 0 \\ -C^{-1}BA^{-1} & C^{-1} \end{pmatrix} \quad (1.9)$$

En appliquant la relation (1.9) on obtient :

$$A=\begin{bmatrix} -(d_2 s_1 + d_3 s_{12}) & -d_3 s_{12} \\ (d_2 c_1 + d_3 c_{12}) & d_3 c_{12} \end{bmatrix} \Rightarrow A^{-1}=\frac{1}{d_3 d_2 s_2}\begin{bmatrix} d_3 c_{12} & d_3 s_{12} \\ -(d_2 c_1 + d_3 c_{12}) & -(d_2 s_1 + d_3 s_{12}) \end{bmatrix}$$

$$C=C^{-1}=\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad B=\begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}$$

$$-C^{-1}BA^{-1}=-\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} \frac{d_3 c_{12}}{d_3 d_2 s_2} & \frac{d_3 s_{12}}{d_3 d_2 s_2} \\ \frac{-(d_2 c_1 + d_3 c_{12})}{d_3 d_2 s_2} & \frac{-(d_2 s_1 + d_3 s_{12})}{d_3 d_2 s_2} \end{bmatrix}$$

$$-C^{-1}BA^{-1}=-\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 0 & 0 \\ \frac{-c_1}{d_3 s_2} & \frac{-s_1}{d_3 s_2} \end{bmatrix}=\begin{bmatrix} \frac{c_1}{d_3 s_2} & \frac{s_1}{d_3 s_2} \\ 0 & 0 \end{bmatrix}$$

Par conséquent J^{-1} est de la forme :

$$J^{-1}=\frac{1}{d_2 d_3 s_2}\begin{bmatrix} d_3 c_{12} & d_3 s_{12} & 0 & 0 \\ -(d_2 c_1 + d_3 c_{12}) & -(d_2 s_1 + d_3 s_{12}) & 0 & 0 \\ d_2 c_1 & d_2 s_1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Donc le MCI est donné par :

$$\dot{q} = \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \dot{r}_4 \end{bmatrix} = \frac{1}{d_2 d_3 s_2} \begin{bmatrix} d_3 c_{12} & d_3 s_{12} & 0 & 0 \\ -(d_2 c_1 + d_3 c_{12}) & -(d_2 s_1 + d_3 s_{12}) & 0 & 0 \\ d_2 c_1 & d_2 s_1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\varphi} \end{bmatrix}$$

I.4 L'espace de travail du robot SCARA

L'espace de travail du robot SCARA que nous avons dans Le laboratoire de commande des processus est un disque creux, d'origine (0,0), de rayon externe égal à $R_{ext} = (d_2 + d_3) = 359mm$ et de rayon interne égal à $R_{in} = 117mm$, l'angle $\theta_{2\max} = \pm 142°$

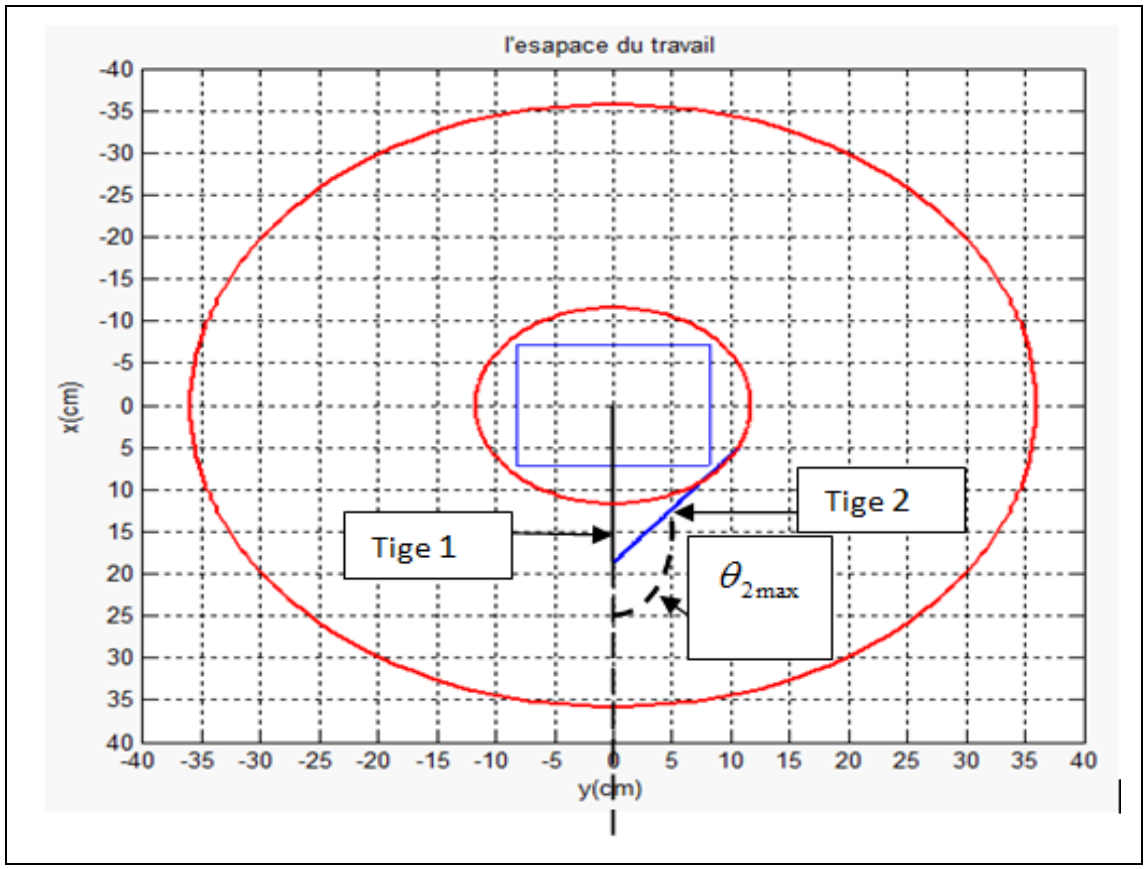


Figure 1.6 : l'espace du travail du robot

I.5 Description de la structure

I.5.1 Structure existante

La structure existante au sein du LCP est un robot à deux degrés de liberté qui présente plusieurs imperfections :

- jeux mécaniques au niveau des articulations,
- absence de roulements au niveau des articulations rotoïdes ce qui conduit à des frottements importants,
- Le poids des liaisons constituants le bras influe sur les déplacements horizontaux et provoque un fléchissement de la dernière liaison horizontale.

Notre but est d'apporter quelques améliorations à la structure existante.

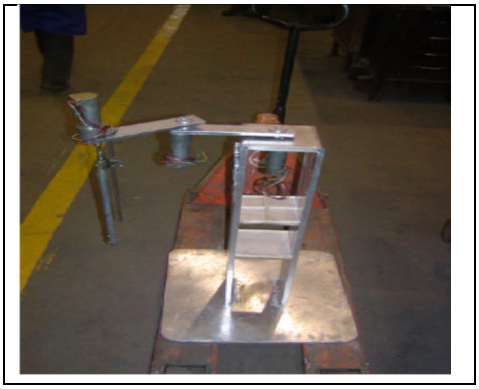


Figure 1.7 : la structure existante du robot SCARA

Cette structure est composée des éléments suivants :

- Une base
- Deux tiges d'aluminium
- Deux moteurs à courants continu

I.5.2 Modifications apportées à la structure

Notre objectif est de réaliser un robot SCARA à 4 degrés de liberté muni de 3 articulations rotoides et d'une articulation prismatique et on prévoit d'utiliser un outil possédant un axe de symétrie autour de l'axe z4, aussi la rotation autour de cet axe n'est pas nécessaire donc on prend $\theta_3 = 0$.

Toutes les modifications apportées à la structure étaient dans le but d'améliorer le fonctionnement de cette structure. Par conséquent nous avons effectué les travaux suivants :

- installation des roulements au niveau des articulations,
- diminution de la longueur des liaisons horizontales,
- diminution de la hauteur du Robot,
- réalisation de l'articulation prismatique en utilisant un vérin pneumatique pour obtenir le mouvement vertical du robot SCARA,
- réalisation d'une pince mécanique où le mouvement de serrage d'un doigt est contrôlé par un moteur.

I.5.3 Présentation des actionneurs utilisés

Les actionneurs équipant notre structure sont deux types : les moteurs à courants continus et les vérins pneumatiques.

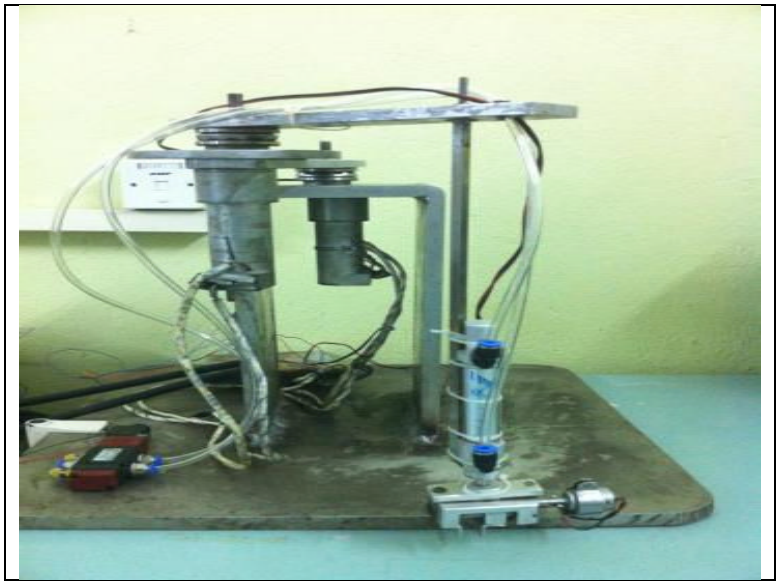


Figure 1.8: la structure modifiée

I.5.3.a Caractéristiques des moteurs utilisés

Les moteurs électriques installés au niveau des articulations rotoïdes sont des moteurs à courant continu à aimants permanents munis d'encodeurs incrémentaux optiques ayant une résolution de 500CPR (compte par tour). Ce sont les moteurs de référence GM9236 du fabriquant américain Pittman [2].

Figure 1.9 : Différents moteurs Pittman

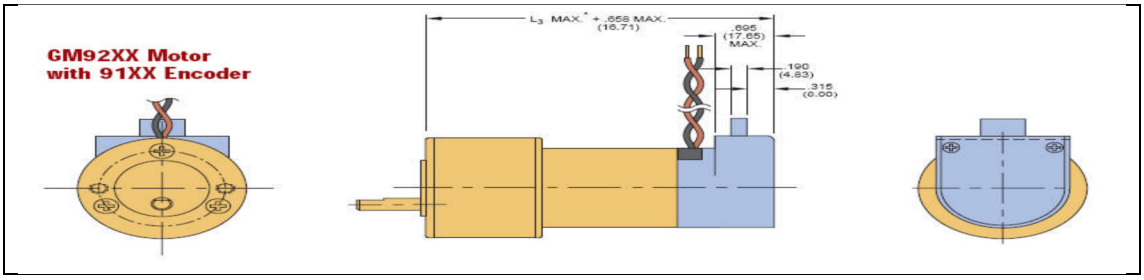


Figure 1.10 : Différentes vues du moteur GM9236 avec encodeur

Les caractéristiques des moteurs à courant continu utilisés sont résumées au tableau 1.2 :

Paramètre	Notation	Unité	Valeur
Constante de couple	K_i	$N.m$	$48.8\ 10^{-3}$
Constante de la force Contre électromotrice	K_b	$V.s/rad$	$48.8\ 10^{-3}$
Inertie du moteur	J_M	$Kg.m^2$	$7.06\ 10^{-3}$
Constante de temps électrique	τ_E	ms	1.06
Constante de temps mécanique	τ_M	ms	8.5
Masse du moteur	W_M	g	391.2
Résistance	R_a	Ω	2.49
Inductance	L_a	mH	2.63
Tension d'alimentation	V_a	$Volts$	24
Rapport de réduction	M	_	65.5

Tableau 1.2 : Caractéristique du moteur PITMAN GM9236027

I.5.3.b Caractéristique du vérin utilisé

Le vérin utilisé est un vérin à double effet avec une course de 5cm et de longueur totale de 20 cm. La pression d'air est de 6 bars. Pour commander le vérin nous avons utilisé un distributeur 5/2 monostable, il comporte 5 orifices : l'alimentation en pression, la sortie1, la sortie2, l'échappement1 et l'échappement2. La bobine de ce distributeur sera alimentée par une tension de 24 V et pour cela on a utilisé un relais pour pouvoir commander le distributeur du vérin par la carte input/output numérique de la DSP sachant que cette sortie délivre une tension de $5V$ alors qu'on dispose d'une bobine de $24V\ DC$.

La sortie TOR de la DSP sera activée ou désactivée en lançant un programme PLC. La tension délivrée à cette sortie doit exciter le transistor pour activer la bobine du relais, qui permet au contact k de se fermer et ainsi, la bobine du distributeur pourra être alimentée en 24 V. Donc si la sortie TOR de la DSP est active, le piston du vérin avance, sinon le piston revient à sa position initiale.

Pour plus de détail sur les vérins et les distributeurs consulter l'annexe A

Le schéma de commande est donné sur la figure 1.11.

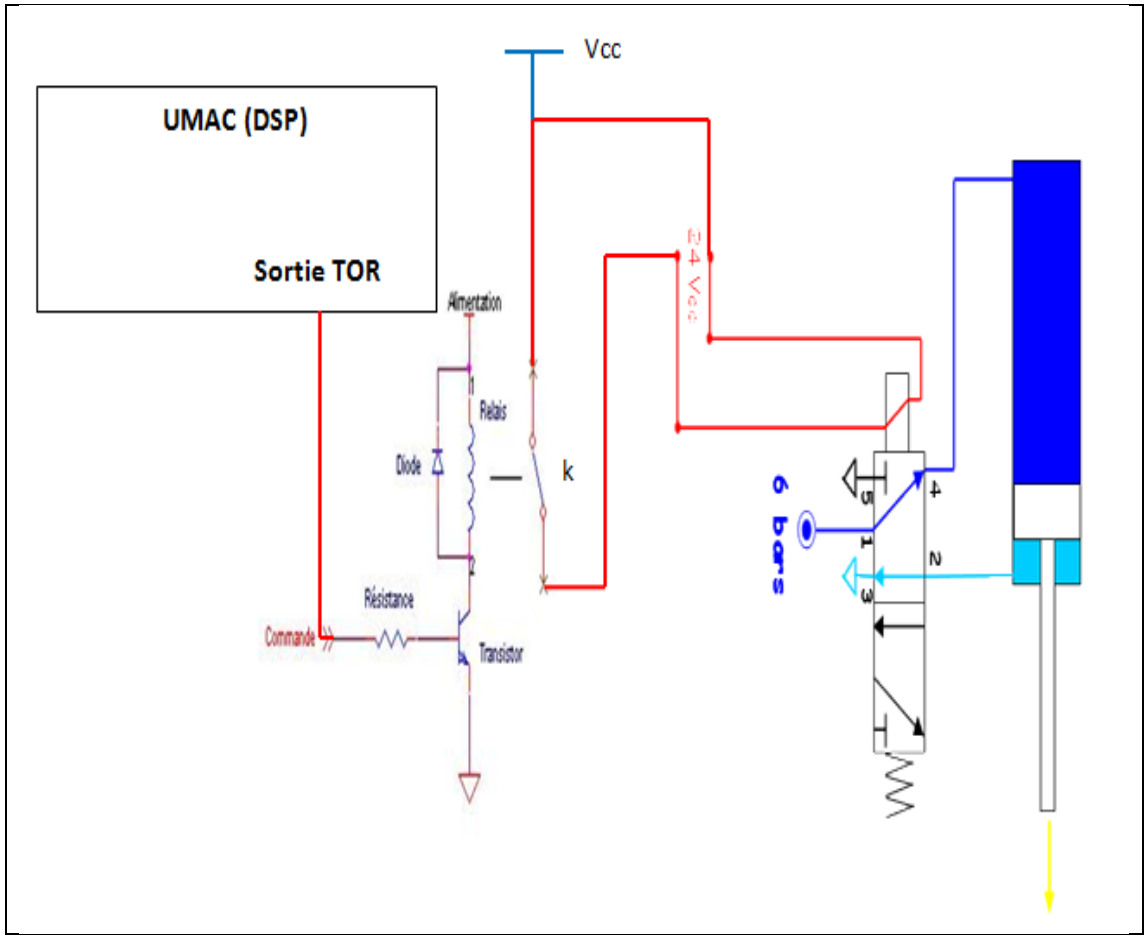


Figure 1.11 : schéma de commande du vérin

I.6 La pince mécanique utilisé

Pour la mise en pratique de la tâche « pick and place », nous avons conçu en atelier de mécanique une pince mécanique dont le modèle est donné à la figure 1.12.

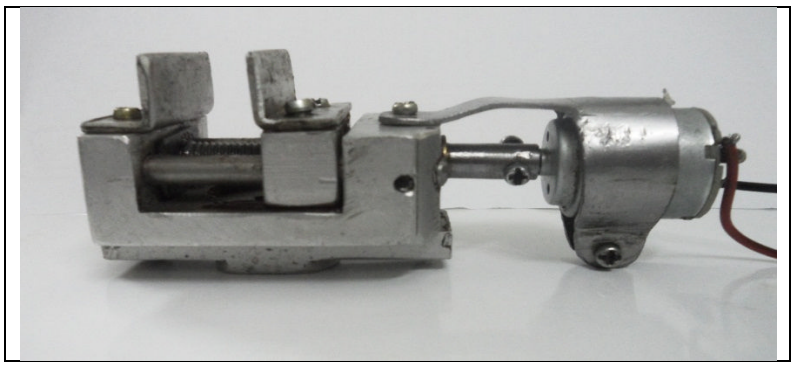


Figure 1.12 : Pince mécanique fabriquée

Son principe est simple ; l'axe d'un moteur à courant continu est couplé à une vis sans fin. La rotation du moteur dans les deux sens entraine le mouvement de la vis sans fin qui produit à son tour l'ouverture ou la fermeture d'un doigt de la pince, l'autre doigt est fixé comme le montre la figure 1.12. La commande de cette pince sera détaillée au chapitre 4.

I.7 Génération du mouvement

La génération du mouvement est la phase la plus importante pour les robots, car dans cette phase s'effectuent les calculs des consignes de référence pour la position, la vitesse ainsi que l'accélération afin d'obtenir la trajectoire désirée.

On distingue 4 types de mouvement qui sont le mouvement entre deux points avec ou sans points intermédiaires dont la trajectoire est libre entre les points ou bien le mouvement entre deux points avec ou sans points intermédiaires dont la trajectoire est imposée (ex : rectiligne, circulaire,…etc.) entre les points. Et pour cela il existe deux méthodes pour la génération du mouvement qui sont [3] :

- La génération du mouvement dans l'espace articulaire
- La génération du mouvement dans l'espace opérationnel

Le choix de la méthode dépend du problème à traiter, chaque méthode a ses propres limites, car les contraintes sont exprimées dans l'espace articulaire ou opérationnel.

I.7.1 Génération du mouvement dans l'espace articulaire

Cette méthode est utilisée dans le cas où la trajectoire est libre entre les points, elle présente plusieurs avantages qui sont [3] :

- Moins de calcul car elle n'utilise pas le MGI et le MCI.
- Les vitesses et les couples maximaux sont déduits directement des limites physiques des actionneurs.
- Le passage sur les configurations singulières n'affecte pas le mouvement.

Cette méthode est appropriée pour la réalisation des déplacements rapides dans un espace dégagé non encombré.

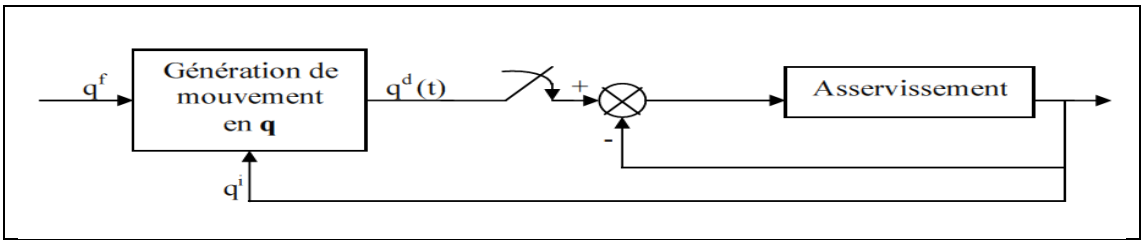


Figure 1.13 : Génération du mouvement dans l'espace articulaire

I.7.2 Génération du mouvement dans l'espace opérationnel

Cette méthode est utilisée dans le cas où la trajectoire est imposée entre les points, elle permet de contrôler la géométrie de la trajectoire, dans cette méthode on a [3] :

- La transformation en coordonnées articulaires de chaque point de la trajectoire.
- La génération de mouvement est mise en échec au passage de la trajectoire calculée pour une position singulière.
- Les limites de vitesses et de couples dans l'espace opérationnel varient selon la configuration du robot.

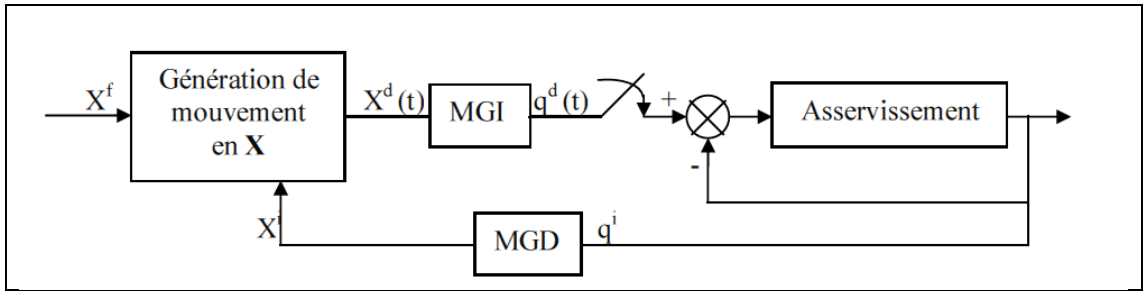


Figure 1.14 : Génération du mouvement dans l'espace opérationnel

I.8 Conclusion

Dans ce chapitre, nous avons présenté quelques notions sur les robots manipulateurs industriels de type série, ensuite nous avons modélisé un bras manipulateur à 4 degré de liberté RRRP et cela en déterminant son modèle géométrique directe et inverse ainsi que son modèle cinématique directe et inverse. Après cela, nous avons présenté la structure existante au labo enfin nous avons présenté toutes les modifications apportées à la structure ainsi que les composants utilisés pour la réaliser.

CHAPITRE 2
DESCRIPTION ET PROGRAMMATION DE L'INTERFACE UMAC

II.1 Introduction

L'évolution de la technologie à un grand impact sur la technique de la commande numérique, aussi actuellement la commande numérique a pratiquement supplanté la commande analogique. Ceci est dû à l'apparition de dispositifs très performants qui permettent de commander des systèmes tels que les moteurs électriques, des procédés industriels et des machines-outils…etc. Parmi ces dispositifs, on trouve les automates programmables et la DSP (Digital Signal Processor), cette dernière est plus utilisée dans le domaine de l'automatique grâce à son architecture.

L'interface UMAC (Universal Motion and Automation Controller) est versée vers la commande des moteurs électriques, et cela par la programmation de cette interface avec un ordinateur via un port série RS232/422, elle peut commander plusieurs moteurs à la fois.

II.2 Le système UMAC

L'UMAC est un système modulaire en format 3U utilisant les contrôleurs PMAC (Programmable Multi Axis and Controller) conçu pour des applications diverses. Il utilise la technologie la plus récente des DSP (Digital Signal Processor), par exemple les microprocesseurs DSP série 56k de MOTOROLA, en outre son pouvoir de calcul et de rapidité dans l'exécution permettent de calculer les trajectoires du mouvement, ainsi que les commandes appropriées et de commander plusieurs dispositifs en parallèle.

Il est constitué de plusieurs cartes : la Turbo PMAC2-3U CPU, la carte d'alimentation, la carte d'entrées sorties (I/O), la carte de conversion des signaux et de calcul de la commande, la carte d'amplification des signaux et la carte mère (UBUS). Sur la carte mère sont connectées toutes les autres cartes, en plus ces dernières communiquent entre elles à travers la surface arrière(UBUS). On peut aussi ajouter des cartes supplémentaires si l'UBUS possède plusieurs emplacements. Les systèmes UMAC peuvent communiquer entre eux via un câble fibre optique, ou bien ETHERNET [4].

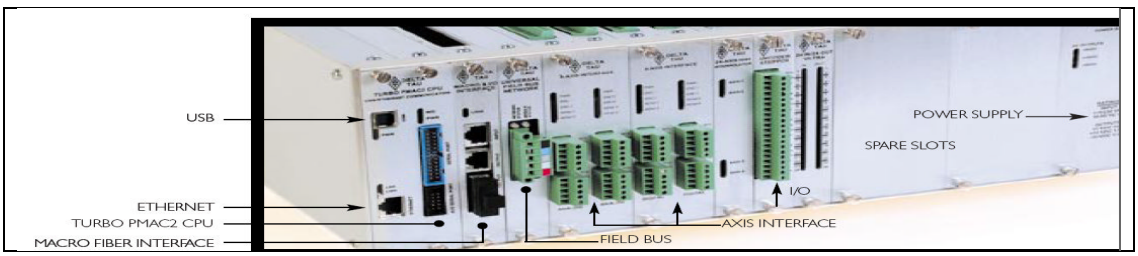


Figure 2.1 : Vue de derrière de L'UMAC

II.3 Les cartes de L'UMAC

L'UMAC possède plusieurs cartes qui sont connectées sur le même rack (carte mère) ou bien l'UBUS [4] :

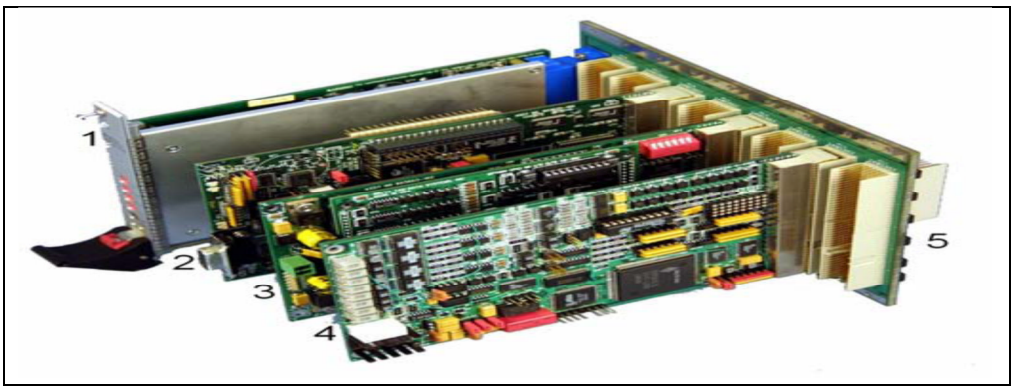


Figure 2.2 : les différentes cartes de L'UMAC

1- carte d'alimentation
2-Turbo PMAC2-3U CPU
3-carte d'entrées, sorties E/S
4-carte d'interface pour les axes
5-carte mère(UBUS)

II.3.1 Turbo PMAC2-3U CPU

La Turbo PMAC2-3U CPU est l'unité de traitement, elle peut commander jusqu'à 32 axes. Elle contient un microprocesseur DSP56303 de MOTOROLA, des mémoires SRAM 128k × 24 pour les programmes et les données, une mémoire flash 1M × 8 pour stocker les microprogrammes, un connecteur série RS232/422 pour la communication avec l'ordinateur et un connecteur sur L'UBUS pour la communication avec les autres cartes.

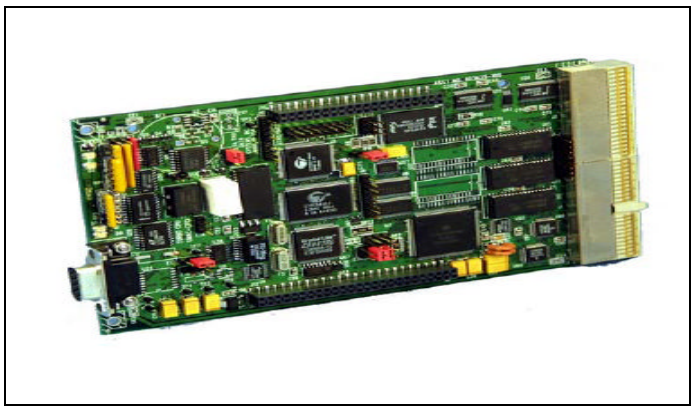


Figure 2.3 : Turbo CPU

II.3.1.a Le microprocesseur DSP 56303

Il contient des mémoires pour les programmes et les données comme ROM, SRAM et DRAM, ainsi qu'un bus pour les périphériques d'E/S, un bus pour programmer la ROM, deux bus de données pour l'accès aux données X et Y, deux bus d'adresses pour les mémoires de X et Y, un bus de données global entre le servo de commande et les autres structures de noyau…etc. donc grâce à son architecture et la disposition de la mémoire, le microprocesseur DSP 56303 offre la possibilité d'accéder aux instructions et les données X et Y du programme pendant le même cycle d'horloge. Donc le microprocesseur dispose d'une grande vitesse d'exécution des programmes, par conséquent L'UMAC peut gérer plusieurs mouvements en même temps [5].

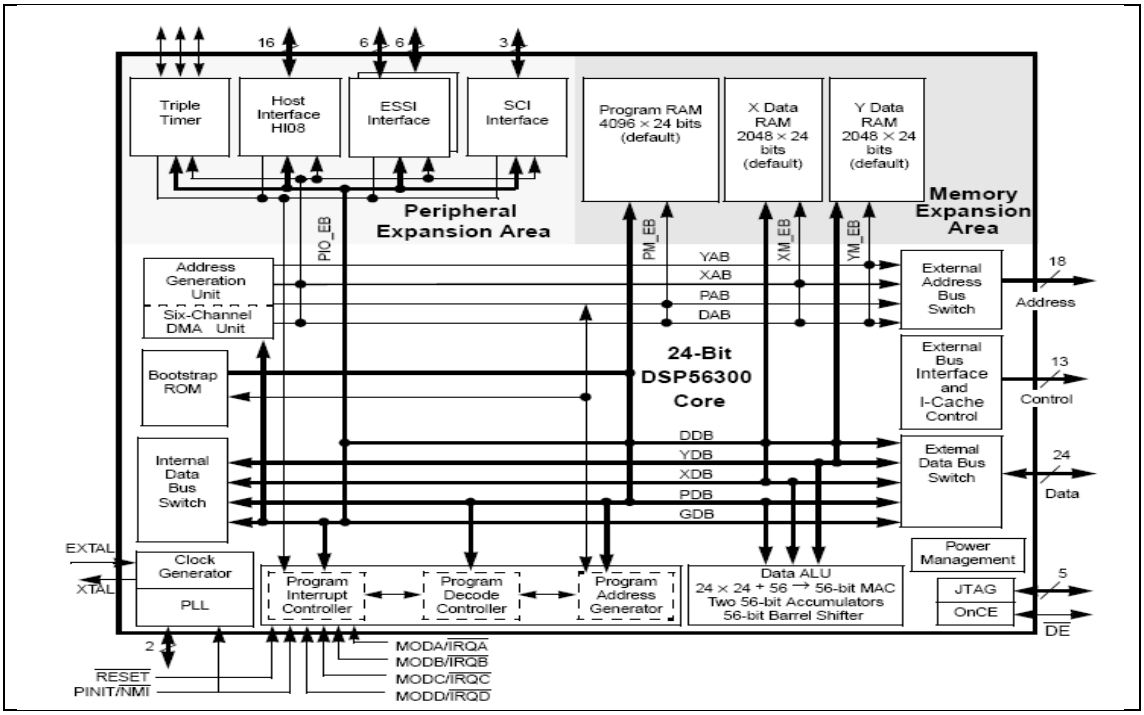


Figure 2.4 : Schéma bloc de DSP56303

II.3.1.b Interface de communication

L'UMAC utilise plusieurs connecteurs pour la communication, comme le connecteur RS 232/422 DB25 et Ethernet ou bien USB (Universal Serial Bus).

II.3.1.c Wachtdog Timer (temporisateur du chien de garde)

Son rôle est de détecter un certain nombre de conditions qui pourraient conduire à un fonctionnement incorrect d'une carte ou d'un circuit donc le Watchdog Timer arrête la carte lors d'un éventuel dysfonctionnement. Les deux conditions pour que le temporisateur se déclenche sont :

- Une tension DC inférieure à $4.75V$, le relais du circuit se déclenchera. Ceci empêche la corruption des registres dus à une tension insuffisante.
- Une fréquence d'exécution inférieure à $25HZ$, le relais du circuit se déclenchera.

II.3.1.d Configuration de la mémoire

Le Turbo PMAC2-3U CPU possède trois types de mémoire :

- Une mémoire SRAM pour les programmes assemblés et compilés, cette mémoire se trouve dans le champ mémoire P. elle est utilisée pour stocker les sous programmes, les PLC_s compilés, les algorithmes écrit par l'utilisateur ainsi que toutes les données intermédiaires lors de la compilation et l'exécution des programmes de mouvement.
- Une mémoire SRAM pour les données X/Y, cette mémoire se trouve dans les champs mémoires X et Y. elle est utilisée pour stocker les programmes de mouvement, les programmes PLC non compilés, les tables d'utilisateur ainsi que les données obtenues lors de l'exécution d'un programme.
- Une mémoire flash non volatile, elle est utilisée pour les microprogrammes, les variables d'installation d'utilisateur, les tables de conversion, les programmes d'utilisateurs ainsi que les algorithmes de commande.

II.3.2 L'UBUS

La famille ACC-Cx relative aux panneaux de la carte mère (d'UBUS compact) donne la possibilité de communication entre la CPU ainsi que toutes les cartes qui sont connectées à L'UBUS où, x représente le nombre rainures que possède L'UBUS.

L'UBUS représenté sur la figure 2.5 contient 8 rainures plus un autre emplacement réservé au bloc d'alimentation.

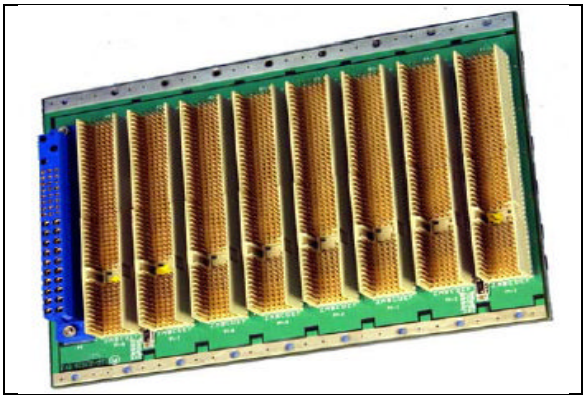


Figure 2.5 : L'UBUS

II.3.3 La carte d'alimentation

La carte d'alimentation ACC-E1 permet d'alimenter toutes les cartes connectées à l'UBUS et cela par la conversion de la source du courant alternatif (85V-240V) pour obtenir $\pm 15VDC$ avec courant maximal de $1.5A$pour alimenter les circuits analogique et $5V$ avec un courant maximal de $14\,A$ pour alimenter les circuits numériques [6].

Figure 2.6 : La carte d'alimentation

II.3.4 DSPGATE

La DSPGATE contient 4 canaux avec 32 registres de mémoires, elle assure la communication entre Turbo PMAC2-3U CPU, les amplificateurs et les capteurs. Chaque canal commande un axe et contient les éléments suivants [7] :

- Une sortie série pour les convertisseurs numérique/analogique de 16 bits
- Une entrée pour la quadrature numérique ou l'impulsion de direction de feedback avec index
- 4 entrées (home, +/-limit, user) qui peuvent déclencher le capteur d'encodeur
- Amplificateur de sortie

- Un comparateur pour le signal de sortie et de référence
- Une entrée d'un convertisseur analogique/numérique de 16 bits
- Trois sorties de commande configurables
- 5 entrées supplémentaires pour la commutation de hall et pour compter/décompter les données ou le code d'erreur
- Trois paires de signal de sortie PWM
- Deux sorties (DAC) analogique $\pm 10V$ pour commander la vitesse, le couple et les amplificateurs sinusoïdaux de commutation
- Deux sorties numériques pour commander des amplificateurs numériques avec un signal direct PWM (Pulse Width Modulation)

Nous disposons de deux DSPGATEs ACC-24E4A, alors on peut commander jusqu'à huit moteurs.

Figure 2.7 : DSPGATE (ACC-24E 4A) avec 4 canaux

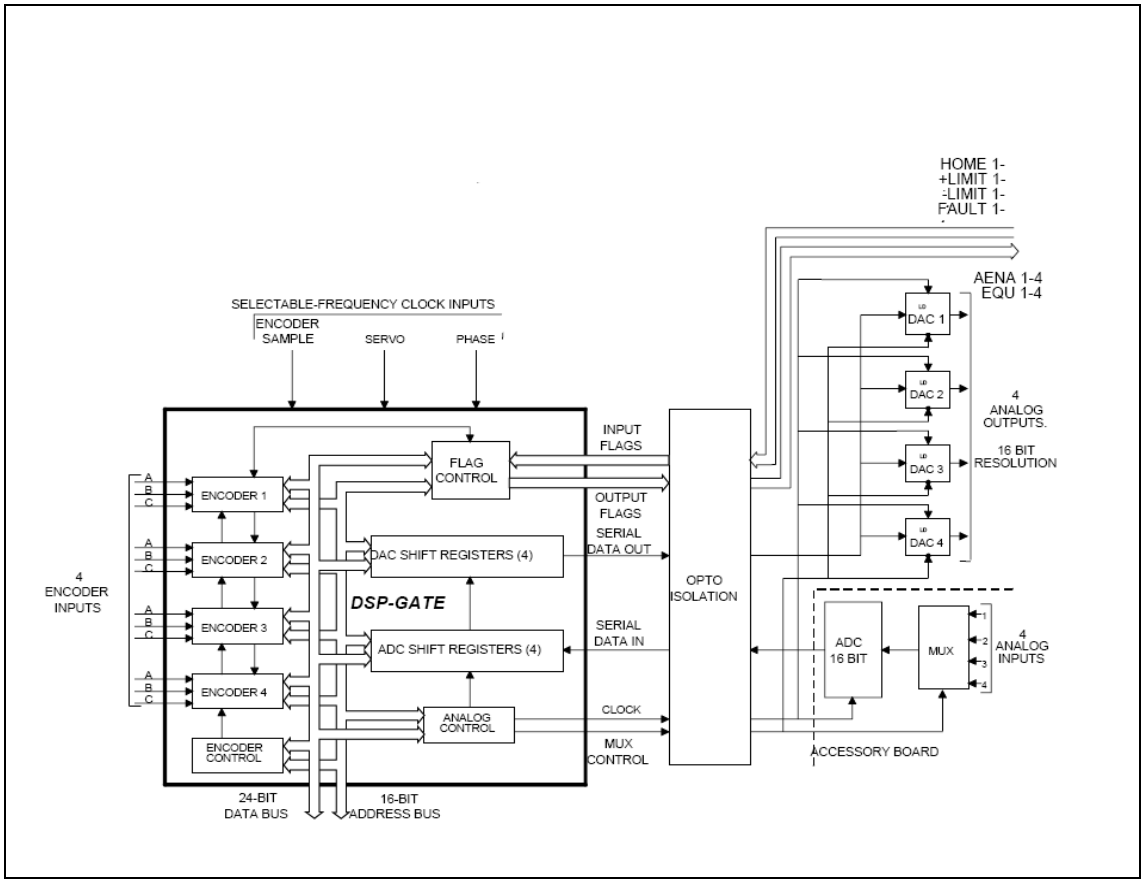


Figure 2.8 : les différentes connections avec la DSPGATE

II.3.5 IOGATE

Nous disposons d'une carte IOGATE ACC-12E possédant 24 entrées et 24 sorties isolées optiquement. Les entrées sont à 12V jusqu'à 24V en continu, les sorties peuvent accepter un courant de 1A et une tension qui peut aller jusqu'à 60V pour chaque sortie ou AC avec un courant de 1A et une tension qui peut aller jusqu'à 240VAC.

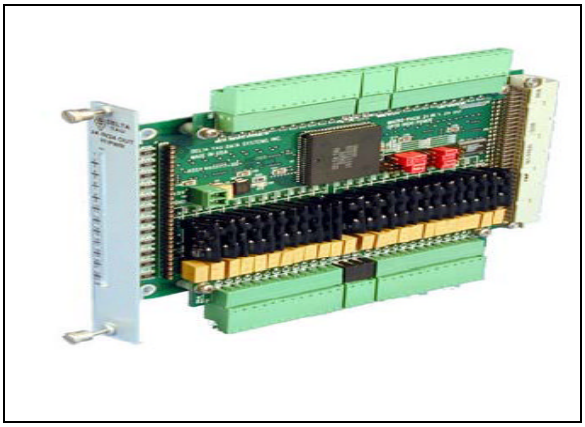


Figure 2.9 : la carte ACC-12E

II.3.6 Amplificateurs

L'UMAC que nous avons utilisé possède huit amplificateurs rangés sur deux cartes AMP-2, le rôle de ces amplificateurs et de donner aux signaux de commande issu de la DSPGATE la puissance nécessaire pour alimenter les moteurs. La tension maximale d'alimentation des cartes AMP-2 et de 40 VDC avec un courant maximum de 3A.

II.4 Fonctionnement de l'UMAC

Pour faire fonctionner L'UMAC et commander les moteurs ainsi que d'autres machines, on doit d'abord écrire un programme sur un ordinateur grâce au logiciel PWIN 32 puis on le charge dans une zone mémoire de la carte TURBO PMAC2-3U CPU via un port série RS 232/422,ensuite les signaux de commande sont envoyés par la DSPGATE aux amplificateurs, ces dernier génèrent des courants proportionnels aux signaux de commande enfin les signaux d'entrées/sorties sont conditionnés et sauvegardés dans des registres afin que la CPU puisse les utilisés. Le fonctionnement de L'UMAC est illustré sur la figure 2.10 [8].

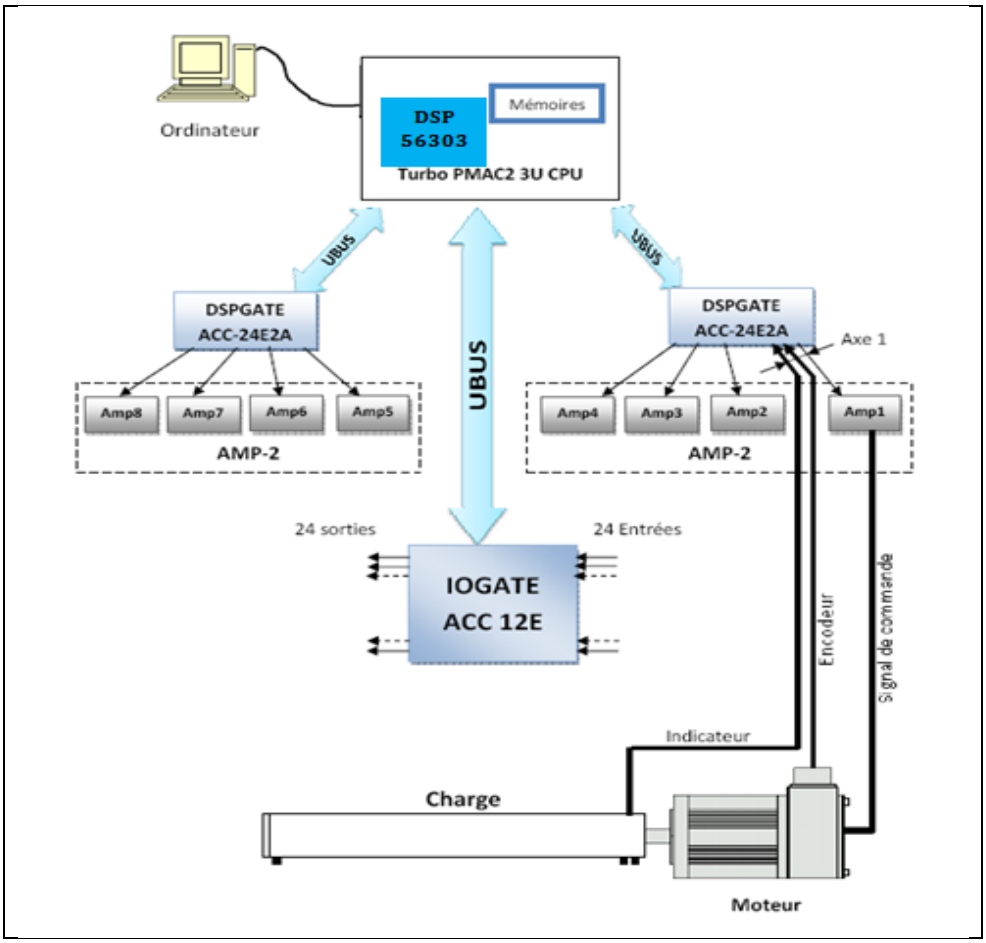


Figure 2.10 : Fonctionnement de L'UMAC

II.5 Les différents Câblages de L'UMAC

II.5.1 Câblage de l'encodeur du moteur à la carte DSPGATE

Les encodeurs digitaux sont les capteurs les plus communs utilisés avec TURBO PMACs. Les encodeurs possèdent cinq fils dont trois sont utilisés pour la détermination de la position angulaire du moteur et les deux autres pour assurer l'alimentation de l'encodeur. Les trois signaux utiles sont CHA, CHB et CHC.

La DSPGTAE reçoit sur **CHA** des impulsions lorsque le moteur tourne ainsi que le disque de l'encodeur, ces impulsions permettent de déterminer la position du disque et le temps correspondant, les calculs nécessaires sont faits par la DSPGATE, cette dernière reçoit aussi des impulsions sur **CHB** qui sont utilisés pour la détermination du sens de rotation. Les impulsions reçues sur CHB sont en quadrature par rapport à celles reçues sur **CHA.** Et à chaque fois le disque de l'encodeur effectue un tour la DSPGATE reçoit une impulsion sur **CHC,** donc la DSPGATE va compter le nombre d'impulsions reçues sur CHC, pour calculer le nombre de tours effectué par le disque. Le câblage du moteur, l'encodeur et l'amplificateur est donné sur la figure 2.11 et 2.12 [9].

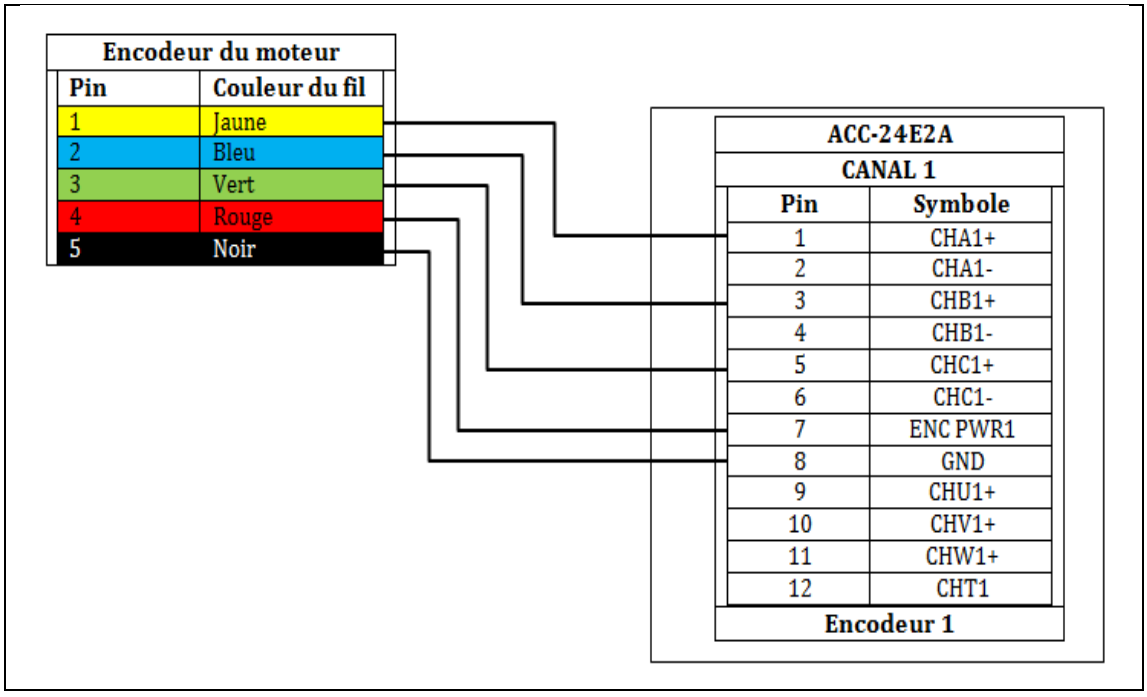


Figure 2.11 : câblage de l'encodeur

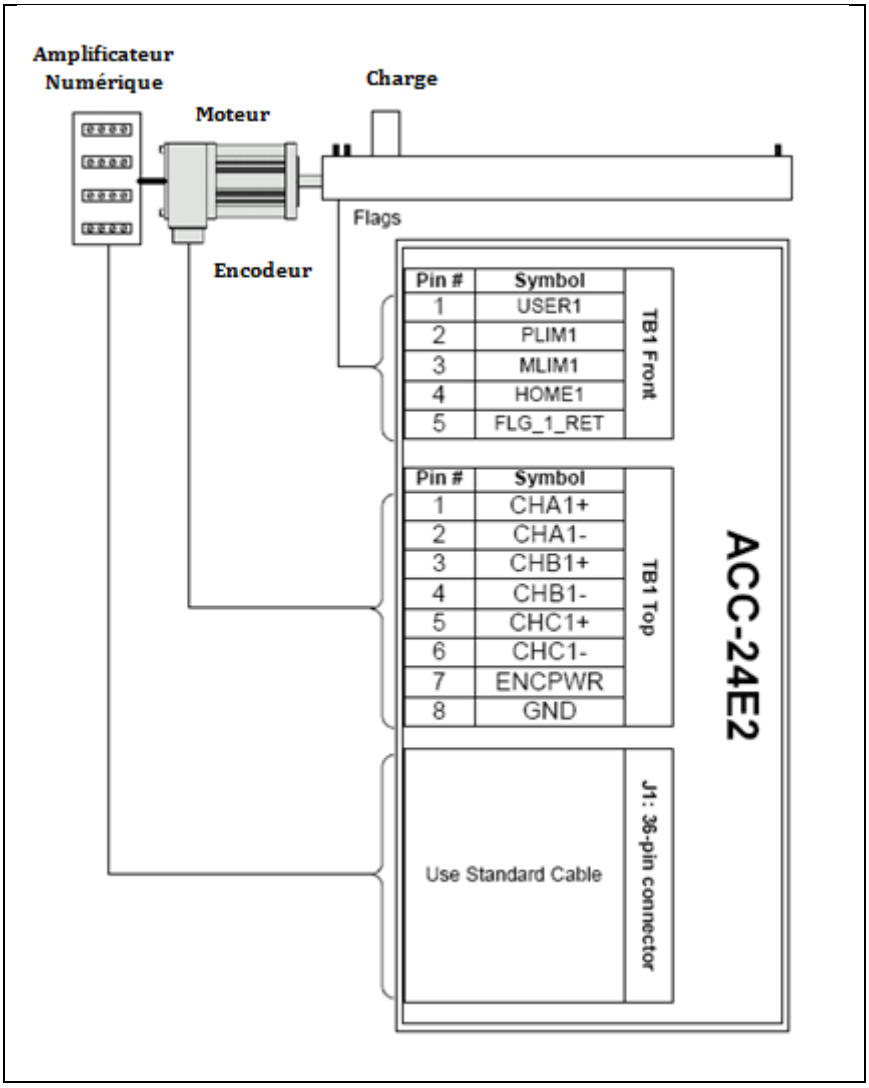


Figure 2.12 : câblage du moteur et de l'amplificateur numérique

II.6 Programmation de L'UMAC

Une fois que le système UMAC est câblé à la machine et les moteurs sont correctement accordés, il est possible d'effectuer n'importe quelle commande d'E/S et ainsi que l'élaboration d'un programme de mouvement. Et pour cela on utilise le logiciel PWIN 32 PRO.

II.6.1 Logiciel de programmation PEWIN 32 PRO

Pour programmer L'UMAC on a utilisé un logiciel conçu par DELTA TAU spécialement pour la programmation de leurs différentes cartes DSP. Ce logiciel est le PEWIN 32 PRO. Les différents composants du logiciel sont : [6]

- **PEWIN 32 Pro** : c'est le programme principal pour le développement et le maintien de n'importe quelle application PMAC
- **PMAC Plot Pro** : permet de tracer les trajectoires des mouvements exécutés à partir d'un terminal
- **UMAC ConfigPro :** il est utilisé pour établir et vérifier la configuration matérielle de l'UMAC
- **PMAC Tuning Pro :** permet de déterminer et d'optimiser les paramètres d'asservissement du régulateur PID implémenté dans cette carte
- **P1 Setup 32 Pro :** fournit une méthode étape par étape pour configurer toute PMAC(1) contrôleur de mouvement de type (analogique).
- **P2 Setup 32 Pro :** fournit une méthode étape par étape pour configurer n'importe quel typePMAC2 (numérique) contrôleur de mouvement.
- **Turbo Setup32 Pro :** fournit une méthode étape par étape pour configurer n'importe quel contrôleur de mouvement de type Turbo PMAC

Il y a trois manières de commander les E/S et le mouvement des moteurs qui sont :

II.6.2 Les Commandes en ligne

Les commandes en ligne peuvent ordonner aux moteurs un mouvement de rotation en lui donnant l'angle de rotation. Ces commandes sont écrites directement, en utilisant la fenêtre terminale de PWIN32PRO, ce mode de commande est très utilisé pour l'essai des différentes commandes de mouvement sur les moteurs.

- **CTRL+A :** On appuie sur les touches « control » et « A » au même temps pour arrêter tous les programmes de mouvement ainsi que toutes les commandes de mouvement
- **CTRL+D** : On appuie sur les touches « control » et « D » au même temps pour arrêter l'exécution de tous les programmes PLC
- **#1J^2000** : le moteur tourne et lorsque l'encodeur compte 2000 impulsions le moteur s'arrête, on appuie sur la touche « Entrée » pour exécuter la commande.

II.6.3 Les Programmes de mouvement

Les programmes de mouvement sont écrits et chargés dans la mémoire en utilisant un éditeur de texte de PWIN32PRO. Un programme permet la synchronisation des mouvements le long des axes. La trajectoire du mouvement obtenue est circulaire, rectiligne…etc. selon la méthode d'interpolation utilisée.

```
CLOSE                   ;fermé tous les buffers ouverts
END GATHER              ; arrêter la réception de données
DELETE GATHER           ; effacer toutes les données des buffers
UNDEFINE ALL            ; effacer toutes les définitions liées à l'axe
&1                      ; système 1 de coordonnée
#1->2000X               ; liée le moteur 1 à l'axe X avec 1unité=2000 impulsion
                        d'encodeur
OPEN PROG1 CLEAR        ; ouvrir le programme 1 pour l'édition
TA 100                  ; le temps d'accélération linéaire est de 100ms
TS 0                    ; Aucune accélération de S-curve
TM 1000                 ; le temps de mouvement est de 1000ms
INC                     ; le mode incrémental
LINEAR                  ; mode d'interpolation linéaire
X 1                     ; le moteur déplace  d'une unité=2000
CLOSE                   ; fermer le buffer ouvert
```

Et pour exécuter ce programme en tape la commande « #1J&1B1R » dans la fenêtre terminale de PWIN32PROpuis on presse sur la touche « Entrée »

II.6.4 Les programmes PLC

Les programmes PLC sont écrits et chargés dans la mémoire en utilisant un éditeur de texte de PWIN32PRO. Ils sont idéaux pour commander les E/S numériques, ils sont utilisés pour donner un menu pour que l'utilisateur ait la main pour choisir le programme de mouvement à exécuter.

```
I5=I512                 ; permet au PLC de s'exécuter
OPEN PLC 1 CLEAR        ; ouvrir PLC 1 pour l'édition
        P1=P1+1         ; à chaque balayage incrémenté P1
        IF(P1=1000)     ; si P1=1000 alors remettre P1 à zéro
        P1=0
        ENDIF
CLOSE
```

Et pour exécuter ce programme PLC 1 en tape la commande « ENABLE PLC 1 » dans la fenêtre terminale de PWIN32PROpuis on presse sur la touche « Entrée ».

II.7 Les variables de Turbo PMAC

Turbo PMAC possède plusieurs types de variables, chaque variable est spécifie par une lettre (I, P, Q ou M) suivi d'un nombre de 0 à 8191. Lors de l'exécution d'un programme les variables seront utilisées en lecture ou bien en écriture.

II.7.1 Les variables I

Les variables I d'installations ou d'initialisations déterminent la configuration d'une carte pour une application donnée. Elles sont aux endroits fixes dans la mémoire et elles ont des significations prédéfinies par le constructeur. Les variables I sont organisés comme suit :

- I0 – I99 configuration globale de la carte
- Ixy00 – Ixy99 configuration du moteur xy
- I3300 – I4799 configuration supplémentaire du moteur
- I4900 – I4999 état de la configuration
- I(5+x)(0+y)00 – I(5+x)(0+y)99 configuration du système de coordonné xy
- I6800 – I6999 configuration de MACRO IC
- I7000 – I7999 configuration de Servo IC
- I8000 – I8191 configuration de la table de conversion d'encodeur

Les variables I sont détaillées dans l'annexe B

La valeur attribuée à une variable I peut être soit une constante ou bien une expression. Les commandes pour faire une attribution sont en ligne (immédiate) si aucun buffer n'est ouvert lors de l'envoi ou bien avec des instructions dans un programme si un buffer est ouvert.

Exemple

$I120 = 45$

$I120 = I120 + 5$

$I(P1 * 100 + 20) = 10$

II.7.2 Les variables P

Les variables P sont des variables d'utilisateurs à usage général, elles ont des emplacements fixes dans la mémoire de TURBO PMAC. Tous les systèmes de coordonnées ont accès aux variables P, ces dernières sont utilisées dans des programmes pour les positions, les distances, les vitesses, le temps et les calculs intermédiaires…etc. il y a 8192 variables P de P0 à P8191. Si la variable I46=0 ou 2 alors les variables P sont situées dans la mémoire principale qui dispose d'un accès rapide, mais dont les valeurs ne sont pas conservés sans la commande **SAVE** en revanche si I46=1 ou 3 alors

les variables P sont situées dans la mémoire RAM sauvegardée par pile qui dispose d'un accès lent, mais dont les valeurs sont conservés automatiquement par la batterie lorsque l'alimentation est coupée.

II.7.3 Les variables Q

Les variables Q sont comme les variables P à usage général. Il y a 8192 variables Q de Q0 à Q8191. Si la variable I46=0 ou 2 alors les variables Q sont situées dans la mémoire principale qui dispose d'un accès rapide, mais dont les valeurs ne sont pas conservés sans la commande SAVE en revanche si I46=1 ou 3 alors les variables Q sont situées dans la mémoire RAM sauvegardée par pile qui dispose d'un accès lent, mais dont les valeurs sont conservés automatiquement par la batterie lorsque l'alimentation est coupée. Plusieurs variables Q ont des utilisations spéciales.

Exemple

- La commande **ATAN2**prend Q_0 comme argument de cosinus c.-à-d. $ATAN2(Q_8) = ATAN\left(\frac{Q_8}{Q_0}\right)$
- La commande **READ** place les lettres A à Z dans Q_{101} à Q_{126}
- Lors de l'utilisation d'un sous-programme cinématique, on utilise les variables Q_1 à Q_9 pour les positions et Q_{11} à Q_{19} pour les vitesses
- La variable Q_{10} indique si des mouvements de PVT sont convertis

II.7.4 Les variables M

Les variables M sont utilisées pour donner aux utilisateurs la facilité d'accès à la mémoire Turbo PMAC ainsi que les entrées/ sorties. Chaque variable est définie par l'emplacement, la taille et le format de la valeur. Il y a 8192 variables M de M0 à M8191, le nombre de la variable M peut être indiqué avec une constante (M300) ou bien avec une expression (M (P1+40)). Une variable M peut avoir un type parmi les suivants :

- X : points fixes de 1 à 24 bits dans la mémoire X
- Y : points fixes de 1 à 24 bits dans la mémoire Y
- D : point fixe de 48 bits à travers la mémoire de X et de Y
- L : point mobile de 48 bits à travers la mémoire de X et de Y
- …etc.

Exemple

La définition d'une variable M :

M1->Y :$078C02, 8,1
M1->Y :$078003, 8,16, S
M161->D :$8B
M5141->L :$2041

II.8 Ecriture et exécution d'un programme de mouvement

Les programmes de mouvement sont le principal mécanisme de Turbo PMAC pour décrire le mouvement désiré, L'UMAC offre un moyen simple pour décrire le mouvement avec des opérations synchrones. Turbo PMAC peut supporter jusqu'à 224 programmes de mouvement en même temps, les 16 systèmes de coordonnées peuvent exécuter n'importe quel de ces programmes.

Un programme de mouvement peut appeler les autres programmes comme des sous-programmes sans ou avec argument.

Lors de l'utilisation des calculs cinématique dans un programme les variables P1 à P32 et Q1 à Q10 ne devraient pas être utilisées car Turbo PMAC écrira dans ces dernières variables lors d'exécution d'un programme ainsi que Q11 à Q19 et P101 à P132 car elles sont utilisées pour les vitesses calculées.

Plusieurs commandes sont utilisées pour élaborer un programme de mouvement qui seront détailler dans la suite de chapitre ainsi que dans l'annexe C.

II.8.1 Système de coordonnée

Un système de coordonnée dans Turbo PMAC regroupe un ou plusieurs moteurs afin de synchroniser les mouvements. La définition d'un système de coordonnée dans un programme se fait par l'instruction **&n** avec **n** c'est le numéro du système de coordonnée.

II.8.2 Les axes

Un axe est un élément d'un système de coordonnées, il peut y avoir jusqu'à neuf axes indépendant dans un système de coordonnée choisi parmi X, Y, Z, A, B, C, U, V et W. on peut attribuer un moteur à un axe donnée par l'instruction #suivi du **n->**avec **n** c'est lu numéro du moteur puis le rapport de définition d'axe qui définit la graduation des unités d'utilisateur des axes et enfin suivi de la lettre de l'axe. Les sont définies comme suit :

- **X, Y et Z** : sont Les axes principaux linéaires
- **A, B et C** : sont les axes de rotations avec l'axe A tourne autour de l'axe X, B autour de Y et C autour de Z
- **U, V et W** : sont les axes secondaires linéaires

Exemple

Le moteur 1 est attaché à l'axe X donc on écrit #1->1000X

Si on écrit dans le programme X 4 cela veut dire que le moteur va faire un déplacement égal à 4000 incréments d'encodeur

II.8.3 Ecriture d'un programme de mouvement

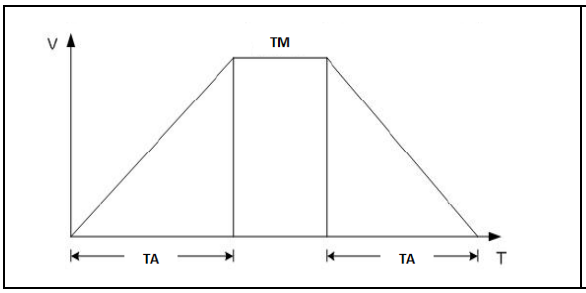
L'écriture d'un programme est l'étape la plus importante car dans ce programme on définit le type de mouvement, la vitesse, le mode incrémental ou absolue ainsi que les trajectoires désirées, et pour cela on va utiliser plusieurs commandes citées dans l'annexe B pour élaborer ce programme. La structure générale d'un programme de mouvement est la suivante :

```
CLOSE                    ; fermé tous les buffers ouverts
DELETE GATHER            ; effacer toutes les données
UNDEFINE ALL             ; effacer toutes les définitions qui se trouvent dans les
                         systèmes de coordonnées
&1  #1->2000X            ; attacher le moteur 1 à l'axe X avec
                         1unité=2000incréments d'encodeur
OPEN PROG 1 CLEAR        ; préparer un buffer pour écrire le programme 1
{
Les instructions du programme
}
CLOSE                    ; fermer le buffer
```

II.8.4 Les types de mouvement

II.8.4.a Le mouvement linéaire

Le mouvement linéaire est le mode par défaut dans un programme de mouvement, la commande utilisée pour indiquer ce mode est **LINEAR**, les profils de vitesse du mouvement linéaire sont donnés par les Figures2.13 et 2.14.



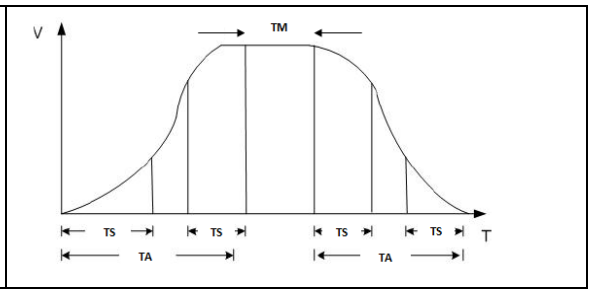


Figure 2.13 : Mouvement linéaire sans S-curve **Figure 2.14 Mouvement linéaire avec S-curve**

Avec :

- **TM :** le temps de mouvement
- **TA :** le temps d'accélération
- **TS :** le temps de courbure

Dans un programme de mouvement le temps de mouvement est spécifié par **TM** ou bien en donnant la vitesse avec **F**. on peut aussi spécifier pour chaque axe sa propre vitesse de déplacement en utilisant la commande **FRAX(X,Y)**ou bien **FRAX(X,Y,Z)**.

Exemple

FRAX(X,Y)
X3 Y4 F10

$$D = \sqrt{3^2 + 4^2} = 5$$
$$TM = 5/10 = 0.5$$
$$V_x = 3/0.5 = 6$$
$$V_y = 4/0.5 = 8$$

FRAX(X,Y)
X3 Y4 Z12 F10

$$D = \sqrt{3^2 + 4^2} = 5$$
$$TM = 5/10 = 0.5$$
$$V_x = 3/0.5 = 6$$
$$V_y = 4/0.5 = 8$$
$$V_z = 12/0.5 = 24$$

FRAX(X,Y,Z)
X3 Y4 Z12 F10

$$D = \sqrt{3^2 + 4^2 + 12^2} = 13$$
$$TM = 13/10 = 1.3$$
$$V_x = 3/1.3 = 2.31$$
$$V_y = 4/1.3 = 3.08$$
$$V_z = 12/1.3 = 9.23$$

II.8.4.b Le mouvement circulaire

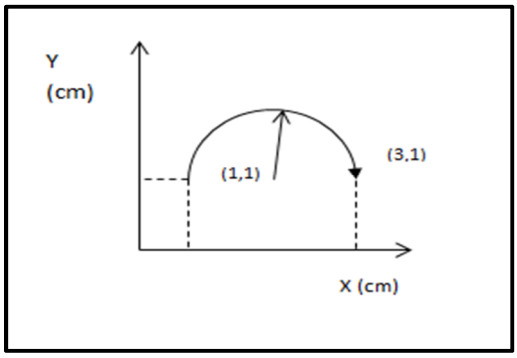
Le mouvement entre deux positions avec une trajectoire circulaire, et pour cela on doit définir le plan de la trajectoire circulaire par la commande **NORMAL**, on aura donc trois plans sont XY avec **NORMAL K-1**, ZX avec **NORMAL J-1**, YZ avec **NORMAL I-1**. Pour définir une trajectoire circulaire on utilise les commandes **CIRCLE1** pour un cercle dans le sens horaire ou bien **CIRCLE2** pour un cercle dans le sens antihoraire. La forme générale pour dessiner un arc est :

CIRCLE {1 ou 2}
TM (temps de mouvement) ou **F**(la vitesse)
X {n} **Y** {m} **R** {p}, la position (X, Y)= (n, m) en mode absolue ou incrémental et le rayon R=p
On ne peut pas dessiner un cercle avec une seule commande, donc il faut le deviser en deux arcs.
Exemple

```
X 10 Y 10
NORMAL K-1
ABS
CIRCLE 1
F 10
X 30 Y 10 R 10
```

Ou bien

```
X 10 Y 10
NORMAL K-1
INC
CIRCLE 1
F 10
X 20 Y 0 R 10
```



II.8.4.c Le mouvement SPLINE

Le mouvement SPLINE utilise l'interpolation cubique lors de déplacement entre deux points, les commandes utilisées pour avoir le mouvement SPLINE sont **SPLINE1**dans laquelle le mouvement est subdivisé en segments de temps TA égaux et**SPLINE2**dans laquelle le mouvement est subdivisé en segments de temps TA variables.

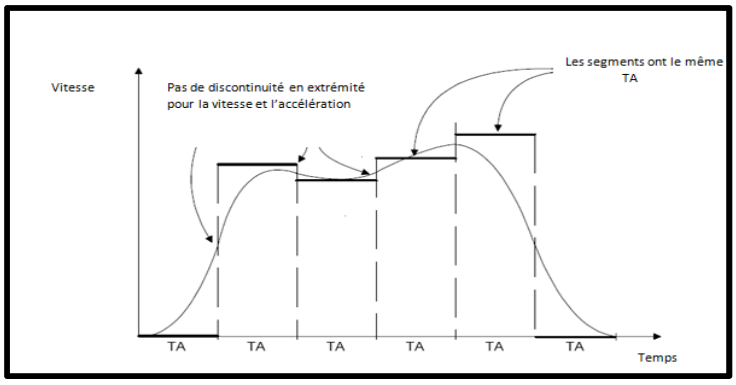


Figure 2.15 : le mouvement SPLINE

II.8.4.d Le mouvement en mode PVT (Position Velocity Time)

Ce mode de mouvement donne la possibilité à l'utilisateur de contrôler la position et la vitesse et cela on donnant la position finale ainsi que la vitesse finale. Ce mode permet de créer n'importe quelle forme de trajectoire.

Exemple

```
CLOSE
END GATHER
DELETE GATHER
UNDEFINE ALL
&1 #1->2000X
OPEN PROG 3 CLEAR
INC                                   ; le mode incrémental
PVT400                          ; le temps de mouvement =400ms
X 2 :40                               ; Déplacement de 2 unités avec une
                        vitesse finale =50
X3 :10                                ; Déplacement de 3 unités avec une
                        vitesse finale =10
CLOSE
```

II.9 Les programmes PLC et PLCC (PLC Compilés)

Turbo PMAC a 64 programmes dont 32 sont des PLC et les 32 autres sont des PLCC leurs fonctionnement est asynchrone. Les PLC sont conçues pour les calculs et les actions qui sont asynchrones au mouvement, ils sont utilisés aussi pour surveillance des entrées analogiques et numériques, la commandes des sorties, l'envoie des messages et surveillance des paramètres du mouvement…etc. La structure générale d'un programme PLC est donnée par :

```
CLOSE
DELETE GATHER
OPEN PLC n
CLEAR
{Les instructions de PLC}
CLOSE
ENABLE PLC n
```

La structure des PLCC est la même que celle des PLC_s, il suffit de remplacer PLC par PLCC

II.10 Conclusion

Dans ce chapitre nous avons décrit les déférentes cartes constituantes la DSP UMAC DELTA TAU ainsi que le logiciel de programmation PEWIN 32 Pro afin de pouvoir implémenter les différents programmes de mouvements assurant l'exécution des différentes tâches de notre robot SCARA .

CHAPITRE 3
LE REGULATEUR PID IMPLEMENTE

III.1 Introduction

La commande des robots manipulateurs a toujours constitué un sujet de recherche important pour les automaticiens. Ceci est dû aux spécificités que présente ces systèmes (fort couplages, frottement, non linéarités…etc.). Il apparaît donc clairement qu'il n'existe pas de stratégie de commande systématique des robots manipulateurs. Chaque manipulateur nécessite une stratégie de commande qui est spécifique à ses propres caractéristiques. Ces dernières sont la morphologie du manipulateur, son nombre de degrés de liberté ou encore le type d'actionneurs utilisés (électriques, pneumatique ou hydrauliques) [2].

III.2 Les commandes utilisées pour la commande des robots

Il existe plusieurs approches pour la commande des robots, les plus utilisées sont [3] :

- La commande classique de type PID (commande décentralisée)
- La commande par découplage NL
- La commande passive
- La commande fondée sur une fonction de Lyapunov
- La commande adaptative
- La commande robuste à structure variable (mode glissant)

Et pour appliquer ces commandes on aura besoins de connaitre le modèle dynamique du robot, soit en utilisant la modélisation ou l'identification.

Pour le calcul du modèle dynamique d'un robot on peut utiliser l'équation de Lagrange donnée par :

$$\Gamma_i = \frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} \qquad i = 1,...n$$

Avec :

- Γ_i : Couple articulaire engendré par l'actionneur i ,
- L : lagrangien du système égal à $E - U$,
- E : énergie cinétique totale du système
- U : énergie potentielle totale du système

L'énergie cinétique est donnée par :

$$E = \frac{1}{2}\dot{q}^T A\dot{q}$$

On obtient alors

$$\Gamma = A(q)\ddot{q} + C(q,\dot{q})\dot{q} + Q(q)$$
$$\Gamma = A(q)\ddot{q} + H(q,\dot{q})$$

$$C\dot{q} = \dot{A}\dot{q} - \frac{\partial E}{\partial q}$$

$$Q = [Q_1 \quad Q_2 \quad \dots \quad Q_n] \qquad Q_i = \frac{\partial U}{\partial q_i}$$

Où

- A : La matrice $(n \times n)$ représentant l'énergie cinétique ou bien l'inertie du robot
- $C(q,\dot{q})\dot{q}$: Le vecteur $(n \times 1)$ représentant les couples/forces de Coriolis et des forces centrifuges
- $H(q,\dot{q})$: Le vecteur $(n \times 1)$ représentant les couples/forces de Coriolis et des forces centrifuges
- Q : Le vecteur représentant les couples/forces de gravité

III.3 La commande décentralisée

Parmi les stratégies de commande des robots, la commande décentralisée est très utilisée, vu sa simplicité et la facilité de sa mise en œuvre. Il s'agira en fait de commander chaque articulation indépendamment en se basant sur les modèles d'actionneurs utilisés. Le robot est donc perçu comme un ensemble de systèmes linéaires du type SISO où l'effet des couplages est considéré comme des perturbations. La commande décentralisée est également la commande utilisée par l'interface UMAC [8].

Le schéma de commande est donné sur la figure 3.1

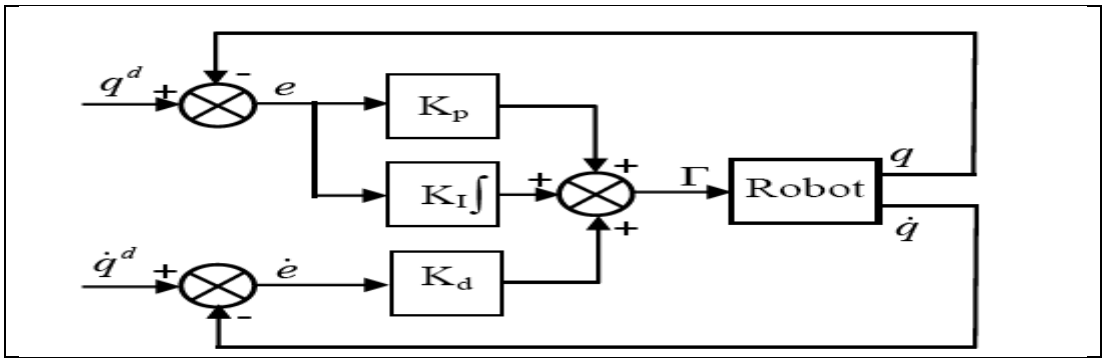


Figure 3.1 : schéma bloc de la commande décentralisé

Le couple articulaire engendré par l'actionneur i, est donnée par la relation suivante :

$$\Gamma_i = N_i K_{ai} K_{Ti} u_i$$

Avec :

- N_i : Le rapport de réduction du moteur i
- K_{ai} : Le gain d'amplificateur i
- K_{Ti} : La constante du couple moteur i
- u_i : Le signal d'entrée de l'amplificateur i

La synthèse de la loi de commande consiste à déterminer Γ_i puis u_i permettant de suivre la consigne désirée.

La loi de commande du PID (fig.3.1) est donnée par

$$\Gamma = K_p\left(q_d(t) - q(t)\right) + K_d\left(\dot{q}_d(t) - \dot{q}(t)\right) + K_I \int_{t_0}^{t} \left(q_d(t) - q(t)\right) dt$$

Où

- $\dot{q}_d(t)$ et $q_d(t)$ désignent respectivement les vitesses et les positions désirées dans l'espace articulaire
- K_p, K_d et K_I sont des matrices diagonales définies positives, de dimension $(n \times n)$, d'éléments génériques respectivement les gains proportionnels K_{pi}, dérivés K_{di} et intégraux K_{Ii}

Le calcul des gains K_{pi}, K_{di} et K_{Ii} est obtenu en procédant comme suit :

Le modèle dynamique de l'articulation i est donnée par :

$$\Gamma_i = a_i \ddot{q}_i + F_{vi} \dot{q}_i + \gamma_i$$

Où :

- $a_i = A_{ii\max}$ désigne la valeur maximale de l'élément diagonal A_{ii} de la matrice d'inertie du robot
- F_{vi} les frottements visqueux
- γ_i représente un couple pertubateur

En négligeant F_{vi} et γ_i , la fonction detransfert du système en boucle fermée est donnée par :

$$\frac{q_i(s)}{q_{id}(s)} = \frac{K_{di}s^2 + K_{pi}s + K_{Ii}}{a_i s^3 + K_{di}s^2 + K_{pi}s + K_{Ii}}$$

Le polynôme caractéristiue s'écrit donc :

$$P(s) = a_i s^3 + K_{di}s^2 + K_{pi}s + K_{Ii}$$

La solution la plus courante en robotique consiste à choisir les gains de manière à obtenir un pôle triple réel et négatif, ce qui donne la réponse la plus rapide possible sans oscillation.

Par conséquent le polynôme caractéristique se factorise de la façon suivante :

$$P(s) = a_i(s + \omega_i)^3$$

Avec :

$\omega_i > 0$ est choisi la plus grande possible , mais elle ne devra pas être supérieure à la pulsation de résonance ω_{ri} du système mécanique pour ne pas le déstabiliser. En général on prend $\omega_i = \frac{\omega_{ri}}{2}$.

On en déduit finalement les gains

$$\begin{cases} K_{pi} = 3a_i\omega_i^2 \\ K_{di} = 3a_i\omega_i \\ K_{Ii} = a_i\omega_i^3 \end{cases}$$

III.4 Régulateur PID pour la commande des moteurs électrique

On a vu dans le chapitre 1, que les moteurs utilisés sont des moteurs à courant continu de type PITMAN, et pour leurs commande on va utiliser des régulateurs PID qui sont suffisants pour assurer une bonne commande des moteurs.

III.4.1 Modélisation des moteurs à courant continu

Les équations représentant le fonctionnement du moteur à courant continu sont données ci-dessous:

$$\begin{cases} u(t) = e(t) + Ri(t) + L\dfrac{di(t)}{dt} \\ e(t) = K_e\omega(t) \\ c_m(t) = K_c i(t) \\ c_m(t) - c_r(t) = J_T \dfrac{d\Omega_m(t)}{dt} \end{cases}$$

Avec :

u : Tension appliqué à l'induit du moteur

e : Force électromotrice

i : Courant d'induit du moteur

Ω_m : Vitesse de rotation du rotor

c_m : Couple «électromagnétique du moteur

c_r : Couple résistant

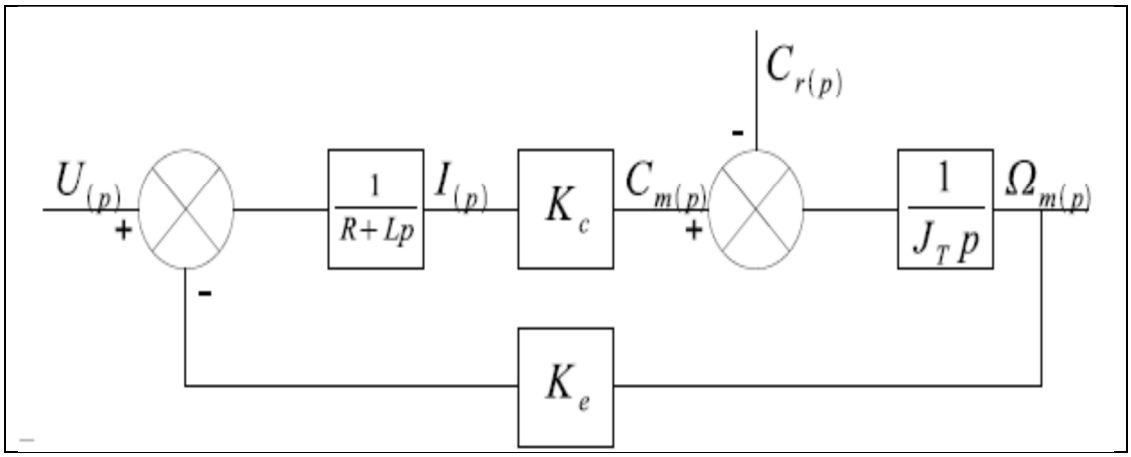


Figure 3.2 : Schéma bloc d'un modèle mathématique du moteur électrique

La fonction de transfert de la vitesse est donnée par :

$$\Omega_m(p) = \frac{K_c}{K_c K_e + RJ_T p + LJ_T p^2} U(p) - \frac{R + Lp}{K_c K_e + RJ_T p + LJ_T p^2} C_r(p)$$

La réponse de la vitesse à un échelon de la tension d'induit est représentée à la figure 3.3 :

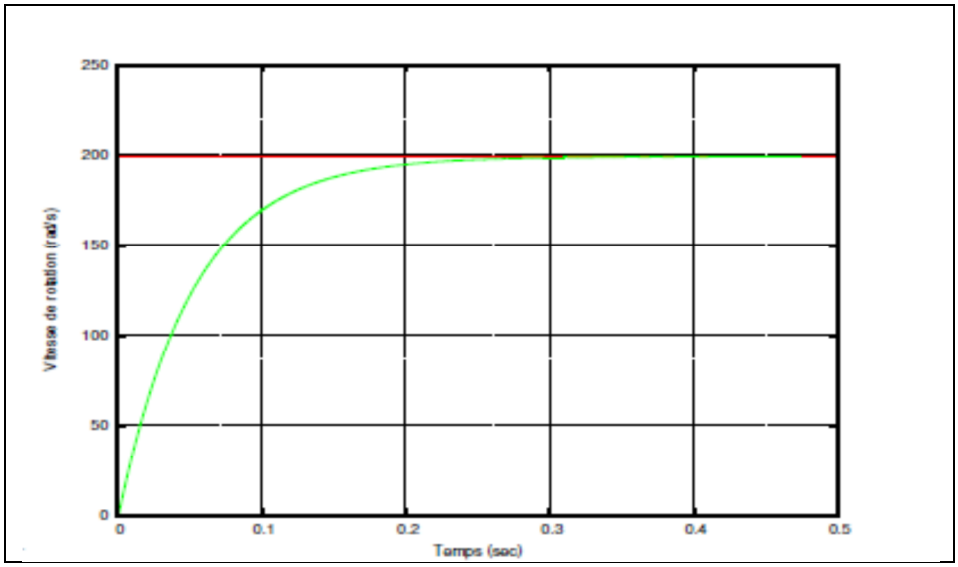


Figure 3.3 Réponse de la vitesse à un échelon de tension

III.4.2 Les différentes actions du régulateur

III.4.2.a L'action proportionnel P

Son rôle est d'amplifier virtuellement l'erreur pour que le système réagisse plus vivement, comme si l'erreur était plus grande qu'elle ne l'est en réalité.

Il permet aussi de vaincre les grandes inerties du système et de diminuer le temps de montée en donnant de la puissance au moteur (plus l'erreur est grande, plus on donne de puissance au moteur). Lorsque l'on augmente K_p, le système réagit plus vite et l'erreur statique s'en trouve améliorée, mais en contrepartie le système perd en stabilité. Le dépassement se fait de plus en plus grand, et le système peut même diverger dans le cas d'un K_p est trop élevé.

III.4.2.b L'action PI

$$consigne\ (t) = K_p \varepsilon(t) + K_i \int_0^t \varepsilon(\tau) d\tau$$

Le terme intégral complète l'action proportionnelle puisqu'il permet de compenser l'erreur statique et d'augmenter la précision en régime permanent. L'idée est d'intégrer l'erreur depuis le début et d'ajouter cette erreur à la consigne. Lorsque la référence se rapproche de la valeur désirée, l'erreur devient de plus en plus faible. Le terme proportionnel n'agit plus mais, le terme intégral subsiste et reste stable, ce qui maintient le moteur à la valeur demandée.

L'action intégrale agissant comme un filtre sur le signal intégré, elle permet de diminuer l'impact des perturbations, et il en résulte alors un système plus stable. Malheureusement un terme intégral trop important peut lui aussi entrainer un dépassement de la consigne, une stabilisation plus lente, voir même des oscillations divergentes.

III.4.2.c L'action PID

$$consigne(t) = K_p \varepsilon(t) + K_i \int_0^t \varepsilon(\tau) d\tau + K_d \frac{d}{dt} \varepsilon(t)$$

$$consigne(p) = K_p \varepsilon(p) + K_i \frac{\varepsilon(p)}{p} + K_d p \varepsilon(p) = \varepsilon(p) \left[K_p + K_i \frac{1}{p} + K_d p \right]$$

Les termes proportionnel et intégral peuvent amener un dépassement de la consigne et des oscillations. Ceci peut conduire à des inversions de polarité pour le moteur, pour limiter ce phénomène indésirable, un troisième élément est introduit c'est le terme dérivé. L'action dérivée devient prépondérante aux abords de la valeur de référence lorsque l'erreur devient faible, que l'action du terme proportionnel faiblit et que l'intégrale varie peu. Elle freine alors le système, limitant le dépassement et diminuant le temps de stabilisation.

L'action dérivée est surtout utilisée dans le cas de variables non bruités, car la dérivation est très sensible au bruitage du signal, donc il faut diminuer son influence dans un asservissement de vitesse, pour lequel la dérivée est l'accélération, variable soumise à de nombreuses perturbations.

Le tableau 3.1 donne l'influence de chaque action sur le système à commander

Coefficient	Temps de montée	Temps de stabilisation	Dépassement	Erreur statique
K_p	Diminue	Augmente	Augmente	Diminue
K_i	Diminue	Augmente	Augmente	Annule
K_d	-	Diminue	Diminue	-

Tableau 3.1 : Influence des différentes actions

III.5 Régulateur PID implémenté dans la carte DSP

La structure du régulateur utilisé par l'UMAC est représentée à la figure 3.4

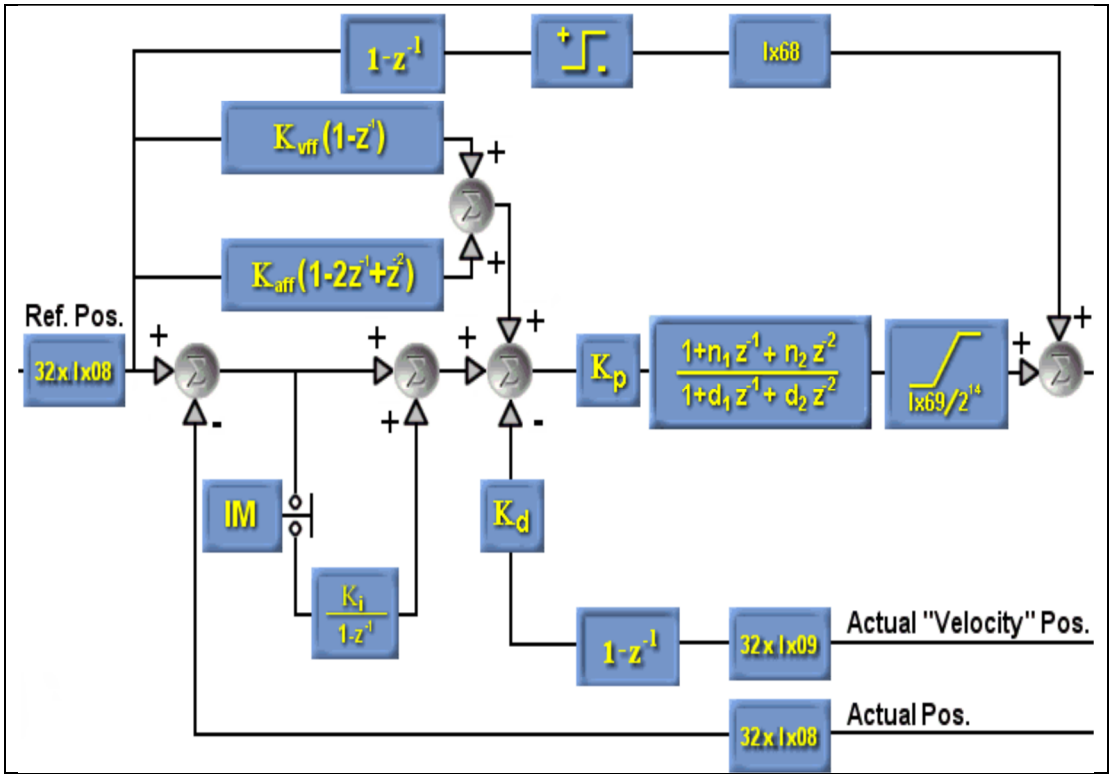


Figure 3.4 : La structure du régulateur PID utilisé

Ce régulateur est un régulateur PID en association avec une boucle de compensation feedforward et d'un filtre éjecteur :

- **Action PID :** Le gain proportionnel $K_p\,(Ixx30)$, le gain dérivé $K_d\,(Ixx31)$ le gain intégral $K_i\,(Ixx33)$
- **Gain de vitesse feedforward** $K_{vff}\,(Ixx32)$: il réduit les erreurs de poursuite, il est placé de manière non causale. Une mauvaise estimation de ce paramètre peut déstabiliser le système. Il est recommandé de mettre ce gain égal au gain dérivé.
- **Gain d'accélération feedforward** $K_{aff}\,(Ixx35)$**: il** réduit ou élimine les erreurs de poursuite dues à l'inertie du système qui sont proportionnelles à l'accélération.
- **Gain feed forward de frottement** *Ixx*68 **: ce** gain est activé lorsque l'erreur statique de la position n'est pas acceptable, une grande valeur de ce paramètre peut déstabiliser le système.

- **Paramètres de sécurité :** Dac limite $(Ixx69)$: pour limiter le courant à ne pas dépasser les valeurs maximales acceptables par l'amplificateur ou le moteur.
- **Limite fatal de l'erreur** $(Ixx11)$: l'erreur dépassant cette limite conduit à l'arrêt du moteur xx.

III.5.1 Détermination des paramètres de la boucle de régulation

Avant de déterminer les paramètres du régulateur assurant une commande satisfaisante des moteurs, nous procédons à l'essai en boucle ouverte du moteur, aussi la figure 3.5 en donne la réponse de la vitesse. Il est clair que la vitesse ne suit pas la consigne, aussi un réglage de celle-ci est nécessaire.

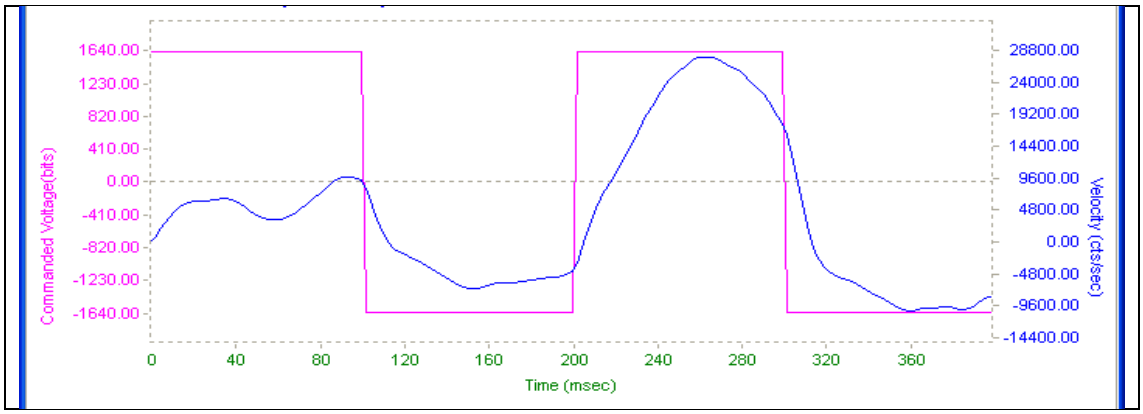


Figure 3.5 : Réponse de la vitesse en boucle ouverte du moteur 1

Nous donnons ci-dessous les deux étapes à suivre pour déterminer correctement les paramètres du régulateur implémenté sur la carte DSP.

- **Etape1 :** La première étape permet de disposer d'une première estimation des paramètres. Elle consiste en l'utilisation de l'auto Tuning de l'application Pmac Tuning Pro du logiciel de programmation PEWIN32 PRO.

L'utilisation de l'application Pmac Tuning Pro, nous a permis d'estimer, dans un premier temps, les paramètres du régulateur lesquels sont donnés au tableau 3.2. Pour ces valeurs des paramètres du régulateur, la réponse indicielle en BF de la position relative au moteur 1 est représentée à la figure 3.6 où, le signal en rouge est la consigne et le signal en bleu est la réponse de la position.
Cette figure montre que la position du moteur suit sa référence, cependant avec la persistance d'une erreur statique et un temps de montée de 0.24s jugé assez lent.

K_P	K_d	K_i	K_{vff}	K_{aff}	$Ixx68$	$Ixx69$	$Ixx11$
73779	6901	0	0	0	0	32767	0

Tableau3.2 : Paramètres estimés du régulateur PID

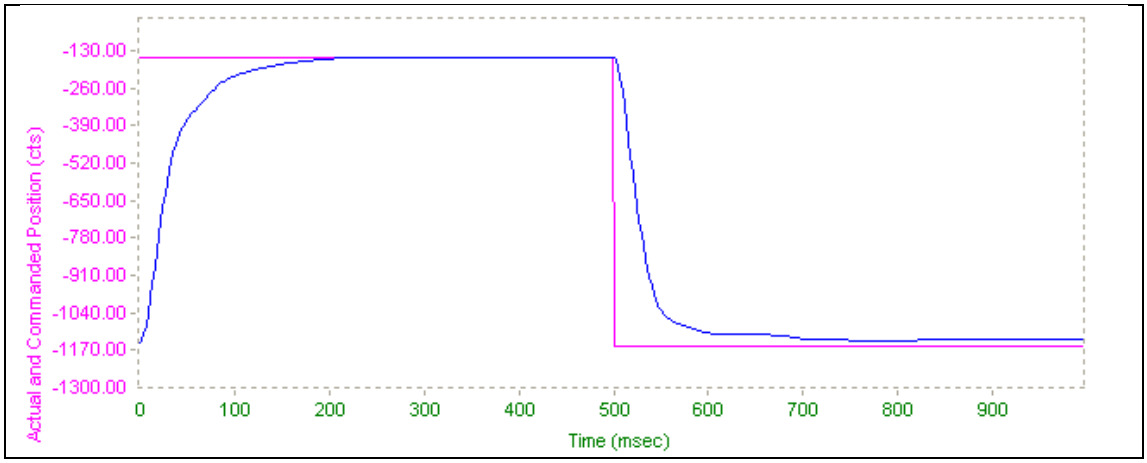


Fig.3.6 Réponse indicielle du moteur 1 en BF

- **Etape 2 :** Elle consiste à réajuster les paramètres du régulateur pour améliorer la réponse à un échelon, vu que l'auto estimation des paramètres s'est avérée insuffisante à donner des résultats satisfaisants. Ceci est obtenu en procédant à une estimation interactive des paramètres où la visualisation de la réponse à un échelon nous guide dans les modifications à effectuer sur les paramètres. Aussi, cette méthode est réellement empirique.

Les paramètres réajustés du régulateur sont donnés au tableau 3.3 et la réponse de la position du moteur à un échelon est représentée à la figure 3.7.

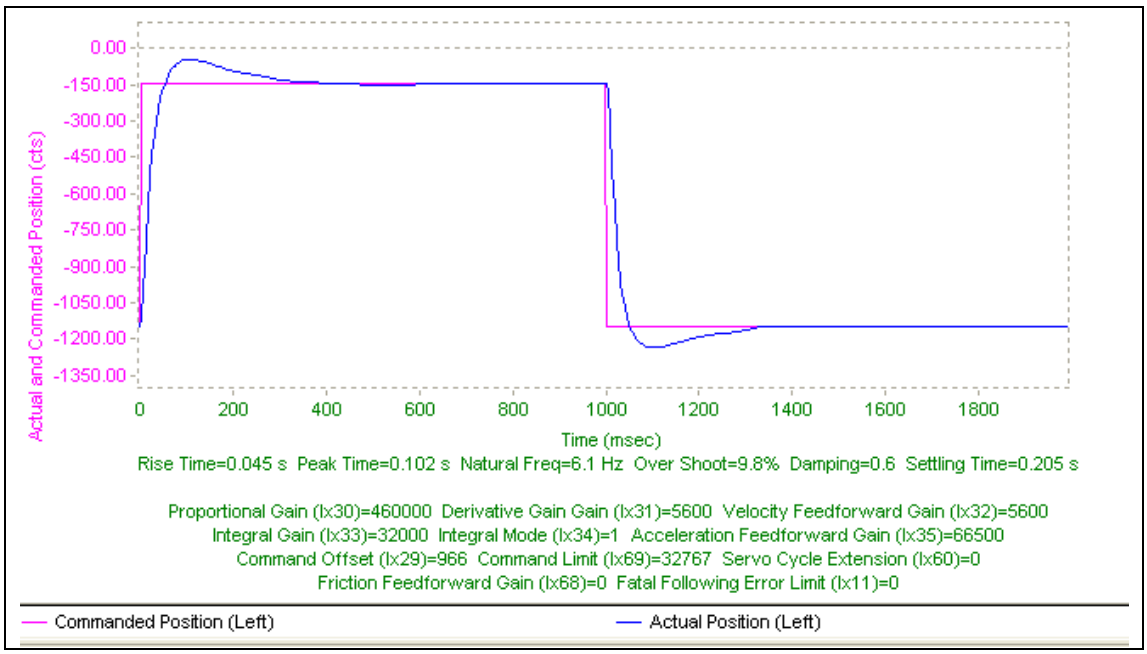


Fig.3.7 Réponse indicielle du moteur 1 en BF

De cette figure, il apparaît que la réponse du moteur suit la référence néanmoins, avec un dépassement de 9.8%. Pour cette réponse, l'erreur statique est nulle et le temps de montée est de 0.102s. Par conséquent, ces paramètres du régulateur PID (Tableau.3.3) conduisent à une réponse satisfaisante meilleure que la première en termes d'erreur statique et de temps de montée.

Aussi et pour tout le reste de notre travail, le régulateur PID de la carte DSP sera implémenté avec ces valeurs réajustées des paramètres.

K_P	K_d	K_i	K_{vff}	K_{aff}	$Ixx68$	$Ixx69$	$Ixx11$
460000	5600	32000	5600	66500	0	32767	0

Tab.3.3 Valeurs réajustées des paramètres du régulateur PID

III.6 Conclusion

Dans ce chapitre, nous nous sommes familiarisés avec les logiciels dédiés à l'exploitation de l'algorithme de commande implémenté sur la carte DSP. Ainsi nous avons utilisé la méthode empirique pour disposer des paramètres du régulateur PID assurant une réponse de vitesse satisfaisante en termes de temps de montée et d'erreur statique. Ces valeurs des paramètres seront chargées au niveau de cette carte pour que le robot puisse exécuter les différentes tâches.

CHAPITRE 4
PLANIFICATION DES TACHES ET IMPLEMENTATION DES PROGRAMMES

IV.1 Introduction

Ayant la nouvelle structure du robot SCARA et disposant de ses modèles géométrique et cinématiques, il est alors possible d'envisager l'exécution par ce robot d'un certain nombre de tâches. Aussi dans le cadre de ce chapitre, nous allons planifier différentes tâches, puis élaborer, implémenter et exécuter les programmes y inférant. Nous veillons à obtenir une précision satisfaisante lors de leurs exécutions en exploitant judicieusement les régulateurs implémentés sur la carte DSP.

IV.2 Implémentation des programmes géométriques et cinématiques :

Dans le premier chapitre, nous avons déterminé les modèles géométriques et les modèles cinématiques de notre robot qu'il faut implémenter actuellement sur la carte DSP. Ceci dans le but d'assurer la coordination du mouvement des moteurs, de plus les erreurs au niveau de ces modèles se répercutent directement sur les erreurs de réglage.

Les tâches à exécuter sont telles que la position verticale de l'outil terminal (h4) se ramène à l'état outil abaissé (hmax) ou l'état outil relevé (hmin) qui est prédéterminée. Par conséquent, il ne reste qu'à déterminer les variables articulaires (q1,q2) en fonction des positions (x,y) dans le plan horizontal et inversement.

La structure générale des programmes géométriques :

La forme générale est donnée par

```
&n OPEN FORWARD
CLEAR
{Les instructions et les fonctions du modèle}
CLOSE
```

IV.2.1 La forme générale d'un programme géométrique inverse :

La forme générale est donnée ci-dessous

```
&n OPEN INVERSE
CLEAR
{Les instructions et les fonctions du modèle}
CLOSE
```

IV.2.2 Les programmes implémentés :

```
I15=0                          ; pour  faire  le calcul des angles en degré
&1
Q91=186.5; la longueur d2
Q92=172.5; la longueur d3
Q93=363.889                    ; 1 degré correspond à 363.889 incréments d'encodeur
```

```
**********Le modèle géométrique directe**********
&1 OPEN FORWARD
CLEAR
Q7=Q91*COS(P1/Q93)+Q92*COS((P1+P2)/Q93)      ; la position x
Q8=Q91*SIN(P1/Q93)+Q92*SIN((P1+P2)/Q93); la position y
CLOSE
    **********Les  modèles géométrique et cinématique inverse**********
&1
#1->I
#2->I
M5182->Y:$00105C, 2
Q94=Q91*Q91+Q92*Q92; (d2)²+ (d3)²
Q95=2*Q91*Q92; 2*d2*d3
&1 OPEN INVERSE
CLEAR
Q20=Q7*Q7+Q8*Q8; x² + y²
Q21=(Q20-Q94)/Q95; cos(θ2)
IF (Q21<0.9998 AND Q21>-0.79); si 0 < θ2 < 142°
                Q22=ACOS(Q21); l'angle θ2 en degré
                Q23=-(Q92*SIN(Q22))*Q7+(Q91+Q92*COS(Q22))*Q8
                Q24=(Q91+Q92*COS(Q22))*Q7+Q92*SIN(Q22)*Q8
        Q25=ATAN(Q23/Q24)                ;   l'angle θ1 en degré
                P1=Q25*Q93; consigne pour le moteur 1
                P2=Q22*Q93; consigne pour le  moteur 2
ELSE
        M5182=1
IF(Q10=1)
        Q26=SIN(Q22)
          IF(Q26>0.0175)
       Q27=Q91*Q92*Q26
       Q28=COS(Q25+Q22)
              Q29=SIN(Q25+Q22)
                          Q30=(Q92*Q28*Q17+Q92*Q29*Q18)/Q27; θ̇1
                           Q31=-(Q7*Q17+Q8*Q18)/Q27;  θ̇2
                          P101=Q30*Q93; vitesse du moteur 1
                          P102=Q31*Q93; vitesse du moteur 2
ELSE
            M5182=1
         ENDIF
ENDIF
ENDIF
CLOSE
```

Nous avons choisi d'imposer le repère (OXYZ) de l'espace cartésien à la base du robot. Les axes (OX, OY) sont pris telles l'axe OY forme un trièdre directe avec l'axe vertical OZ et OX coïncide avec l'axe du bras tendu (θ_2 =0, θ_1=0) qui représente l'origine de l'espace opérationnel aussi vu que l'UMAC prend comme origine des positions la position des robots quand elle sera alimenté comme c'est montré à la Figure 4.1.

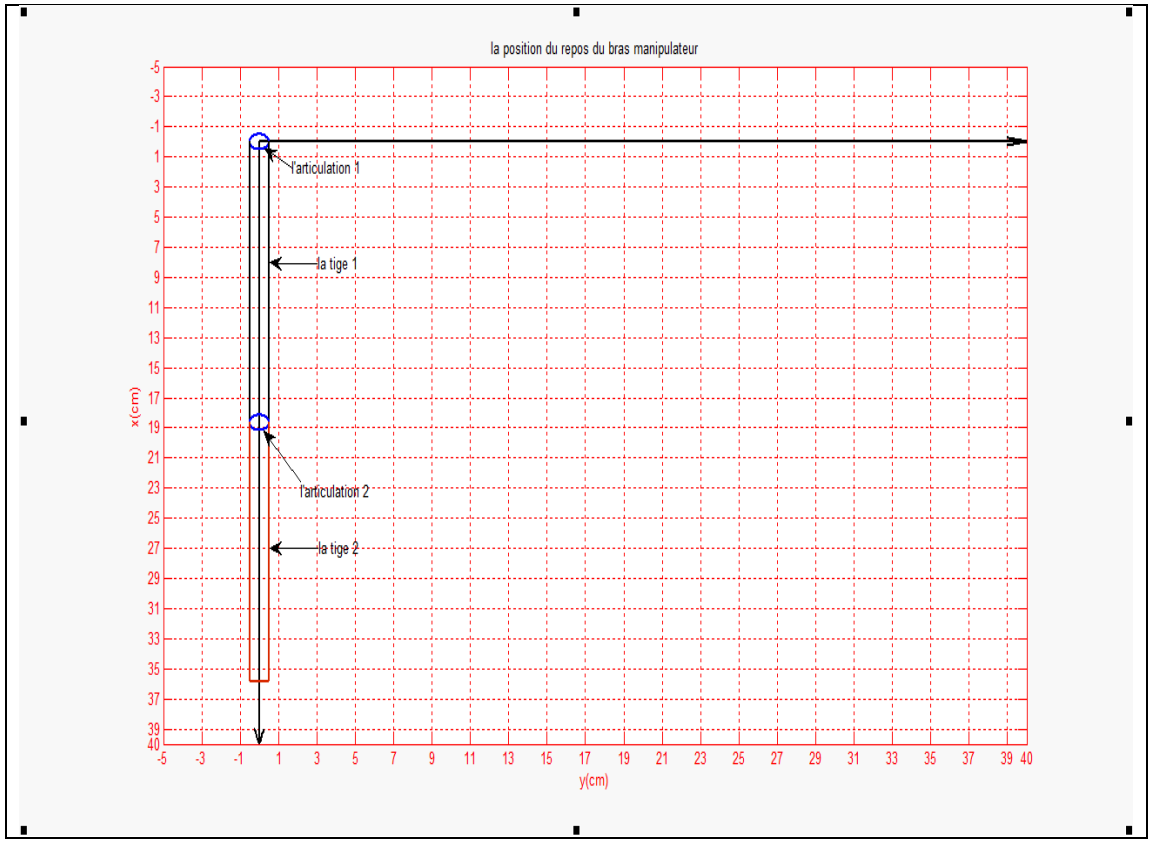


Figure 4.1 : Le placement des repères

Le calcul de ces modèles et l'implémentation de ces programmes d'une manière correcte est impérative pour obtenir une exécution correcte des différentes tâches.

Lors de l'exécution des programmes de mouvements, on fait appel aux calculs géométriques et cinématiques en activant (I5150=1) et (Q10=1).

Donc pour s'assurer de la validité de ces programmes, nous procédons à un premier test. Nous avons tracé un arc de cercle de rayon R=, sur une feuille de papier millimétrée (fig.4.2). Puis en utilisant les axes (XOY), nous avons déterminées les cordonnées (x*,y*) désirées, puis pour le traçage de cet arc (Fig4.3) on fait appel à la procédure du MGI dans le programme de mouvement suivant :

```
DELETE GATHER                    ; effacer toutes les données des buffers
UNDEFINE ALL                     ; effacer toutes les définitions liées à l'axe
OPEN PROG 102 CLEAR
#1->I
#2->I
LINEAR                           ; interpolation linéaire
ABS                              ; le mode absolu
TA 150                           ; le temps d'accélération linéaire est de 150ms
TS 0                             ; Aucune accélération de S-curve
TM 2000                          ; le temps de mouvement est de 2000ms
P40=0                            ; pour soulever le vérin

X 170 Y 10
P40=1                            ; pour descendre  le vérin
DWELL2000                        ; arrêter le mouvement  2 s pour s'assurer que le vérin est
complètement sorti
X 160 Y 67
CIRCLE1
F 10
X 190 Y 156 R 60
P40=0
DWELL 2000          ; arrêter le mouvement  2 s pour s'assurer que le vérin est complètement
                                 rentré
X 359 Y 0           ; pour que le robot revient à sa position du repos
CLOSE
```

Les profils de position et de vitesse sont donnés sur les figues 4.2 et 4.3 :

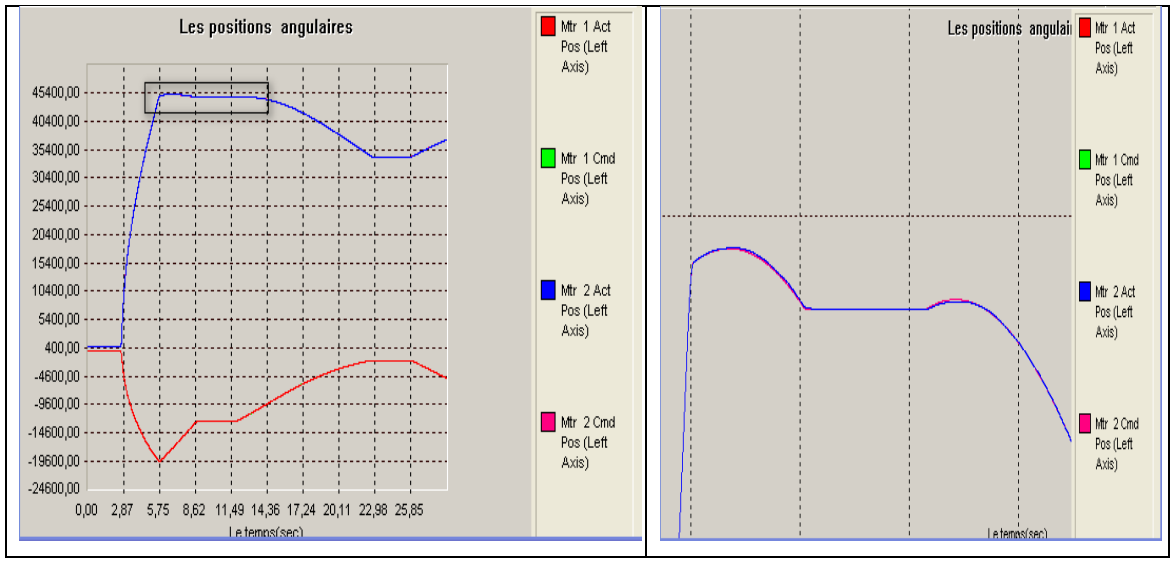


Figure 4.2 : Les positions angulaires des moteurs

En faisant un zoom sur le profil de position, on remarque que le suivi est presque parfait, ce qui coïncide avec les résultats pratiques trouvé.

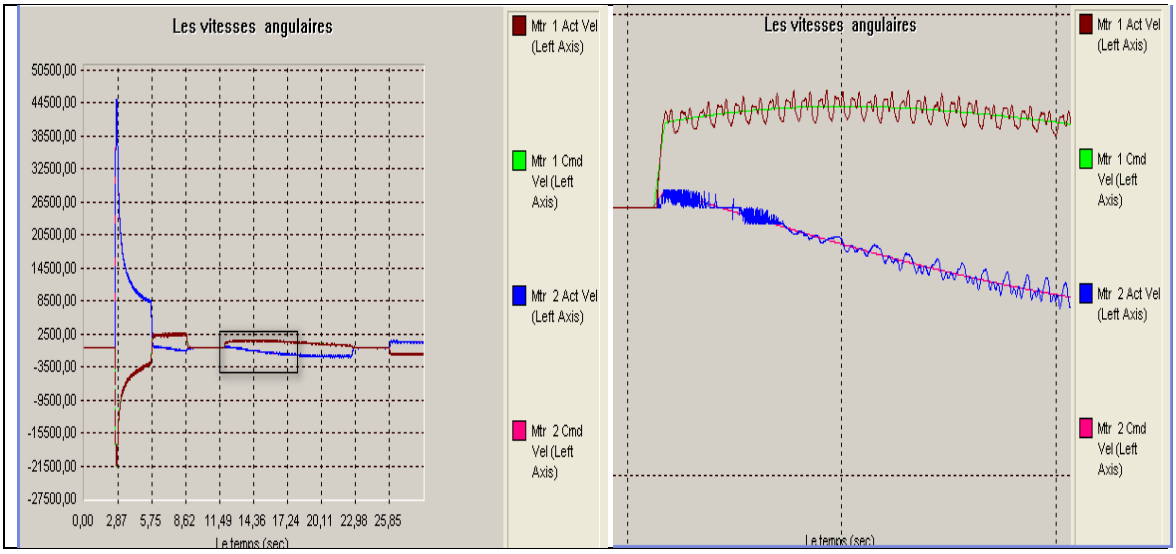


Figure 4.3 : Les vitesses angulaires des moteurs

La figure 4.3 montre que la vitesse des deux moteurs suit le profil de vitesse exigé, le premier pique est dû au déplacement du robot de son point de repos vers la position initiale de la trajectoire.

Les figures 4.4 et 4.5montrent l'arc dessiné à la main et la trajectoire dessiné par le robot respectivement après exécution du programme.

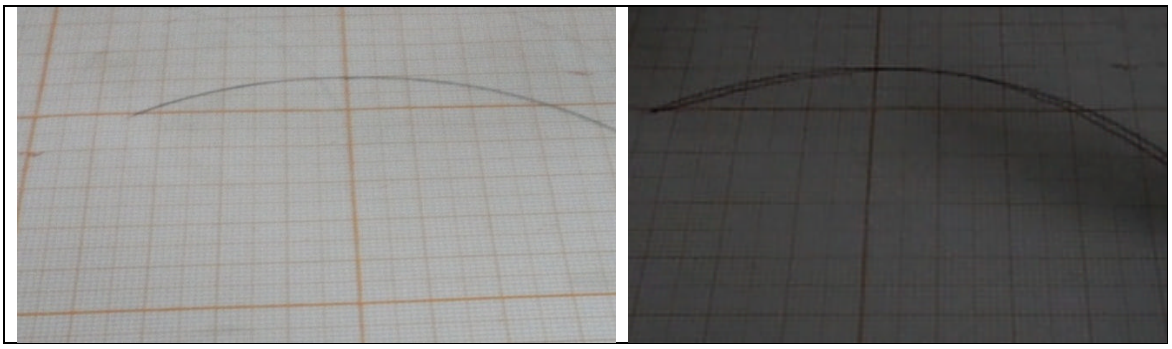


Figure 4.4 : Arc dessiné à la main

Figure 4.5 : Arc dessiné par le robot

Nous constatons que le robot a tracé un arc de cercle qui se superpose sensiblement sur l'arc représenté sur la feuille. Ceci montre une maîtrise satisfaisante des repères préalablement fixés et du modèle MGI, cependant un faible dépassement est enregistré. Ce dernier est dû au fait que :

- La plaque formant la base du robot n'est pas vraiment dans un plan horizontal à cause des défauts de réalisation
- Le vérin n'est pas fortement solidaire de la liaison 3 (tige verticale) du robot.
- Le stylo n'est pas bien fixé sur le corps mobile du vérin

IV.3 Commande du vérin:

Le vérin utilisé admet une commande numérique du type « tout ou rien », aussi on peut exploiter la carte ACC-12E pour le commander. Comme cette carte ne délivre qu'une tension de sortie de 5V et la bobine du distributeur de vérin nécessite une excitation de 24 V par conséquent un circuit externe est utilisé pour exciter la bobine comme indiqué au chapitre 1.

Le programme implémenté est le suivant :

```
M3303->Y:$78C07,0,8
I5=I512 ; active les PLCs
Open PLC 1 Clear
M3303=$07
            IF (P40=1)
               M3303=1                    ; active la sortie 1 pour l'avance du vérin
ELSE
M3303=0                    ; désactive la sortie 1 pour le recul du vérin
ENDIF

Disable PLC 1
Close
```

La variable M3303 va pointer vers l'adresse $78C07 où, la première sortie de la carte input/output ACC-12E est définie. Cette sortie sera activée par un programme PLC, lequel est lancé en parallèle avec le programme de mouvement. Le rôle de ce PLC est de vérifier dans le programme de mouvement s'il y a une instruction relative à l'activation ou à la désactivation d'une sortie numérique (dans notre cas, la désactivation correspond au recul du vérin).

IV.4 Applications :

Les différentes tâches à effectuer par le robot sont :

- tracer de trajectoires
- Ecriture d'une chaine de caractères
- La frappe d'un texte pour sa saisie sur micro-ordinateur
- La tâche « Pick and Place»

IV.4.1 Tracer des trajectoires

En utilisant des programmes de mouvements comme s'est définit dans le chapitre 2, on pourra tracer n'importe quelle trajectoire, on va prendre quelques exemples de trajectoire et de dessin :

IV.4.1.a Tracer du dessin d'un bras manipulateur

Les étapes à suivre pour la planification de cette tâche sont les suivantes :

- **Etape 1 :**

On commence tout d'abord par un dimensionnement de la trajectoire dans le repère absolu lié au robot afin de déterminer les coordonnées opérationnelles exactes de la trajectoire à suivre, comme s'est illustré sur la figure 4.6

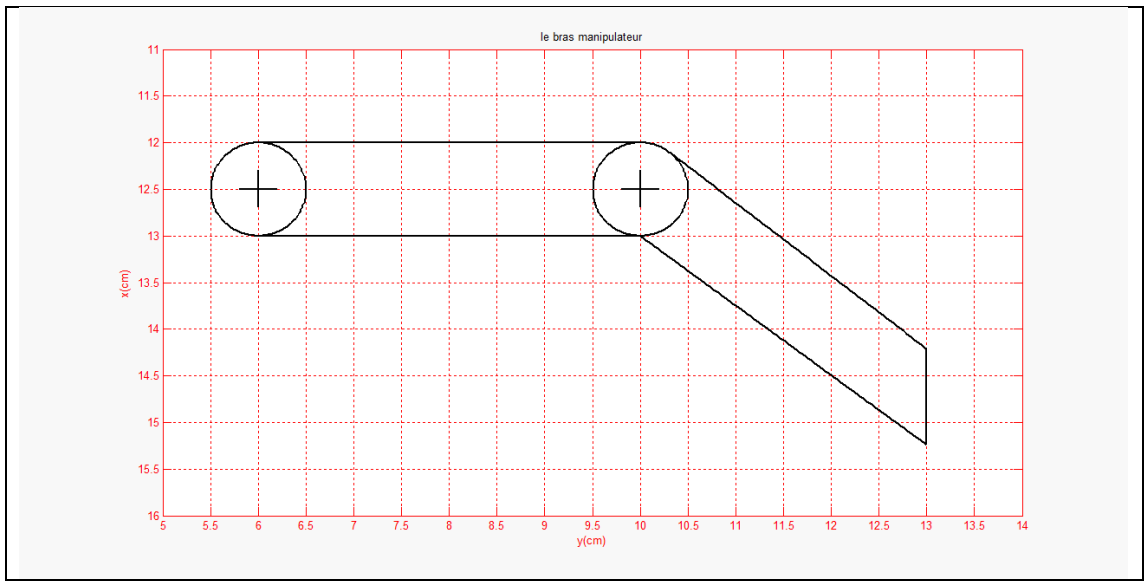


Figure 4.6 : schéma d'un bras manipulateur

Cette trajectoire contient 7 parties : 2 cercles et 5 droites, le déplacement du bras se fera comme suit :

- Tracer d'un cercle de rayon r=0.5 cm et de centre (12.5cm, 6cm)
- Tracer d'une droite à partir du point (12,6) au point (12,10)
- Tracer d'un cercle de rayon r=0.5cm de centre (12.5, 10)
- Tracer d'une droite à partir du point (12.12, 10.03) au point (14.21, 13)
- Tracer d'une droite à partir du point (14.21, 13) au point (15.24, 13)
- Tracer d'une droite à partir du point (15.24, 13) au point (13,10)
- Tracer d'une droite à partir du point (13,10) au point (13, 6)

- **Etape 2 : implémentation des programmes géométriques et cinématiques :**

Sachant que ces positions et ces vitesse sont donnée dans l'espace opérationnel, d'où la nécessité d'introduire le MGI et le MCI pour les transformer à l'espace articulaire afin de déterminer les positions et les vitesses angulaires qui seront utilisés par le robot dans le tracé des trajectoires.

- **Etape 3 : implémentation de programme du mouvement:**

Le tracer de la figure 4.4 est obtenu en implémentant le programme ci-dessous :

```
DELETE GATHER                    ; effacer toutes les données des buffers
UNDEFINE ALL                     ; effacer toutes les définitions liées à l'axe
OPEN PROG 100 CLEAR
#1->I
#2->I
LINEAR                           ; interpolation linéaire
ABS                              ; le mode absolu
TA 150                           ; le temps d'accélération linéaire est de 150ms
TS 0                             ; Aucune accélération de S-curve
TM 2000                          ; le temps de mouvement est de 2000ms
P40=0                            ; pour soulever le vérin
X 120 Y 60
P40=1                            ; pour descendre  le vérin
 DWELL 2000                      ; arrêter le mouvement  2 s pour s'assurer que le vérin
                                 est complètement sorti
CIRCLE1
F 10
X 130 Y 60 R 5
X 120 Y 60 R 5
X 120Y 100
X 130Y 100 R 5
X 120Y 100 R 5
P40=0
 DWELL 2000                      ; arrêter le mouvement  2 s pour s'assurer que le vérin
                                 est complètement rentré
X 130 Y 60 R 5
P40=1
X 121.2 Y 103.3
X 142.1 Y 130
X 152.4  Y 130
X 130  Y 100
X 130  Y 60
X 359 Y 0                        ; pour que le robot revient à sa position du repos
CLOSE
```

- **Etape 4 : Exécution du programme :**

Pour exécuter ce programme, on commence tout d'abord par le positionnement de notre robot et de désigné l'origine de référentiel, en suite on tape les commandes suivantes dans un terminal :

#1J/ ; pour fermer la boucle du moteur 1
#2J/ ; pour fermer la boucle du moteur 2
&1 ; choisir le système de coordonnées 1

B 100 ; pour diriger le système de coordonné 1 au programme 100
R ; pour commencer l'exécution du programme

On utilise également l'application PEWIN 32 Plot pour voir les profils de vitesse et de positions comme s'est illustré sur les figures 4.7 et 4.8 suivantes :

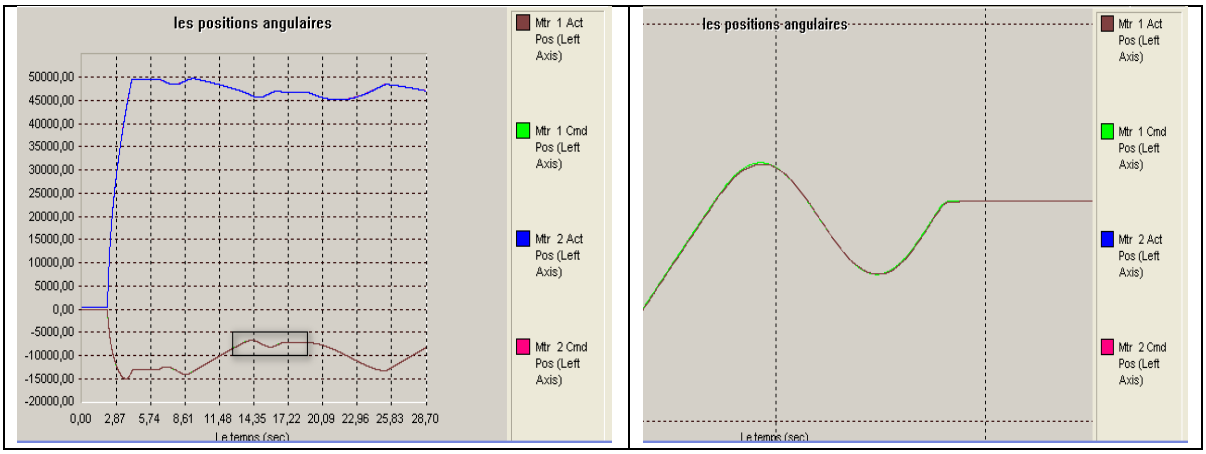


Figure 4.7 : Les positons angulaires des moteurs

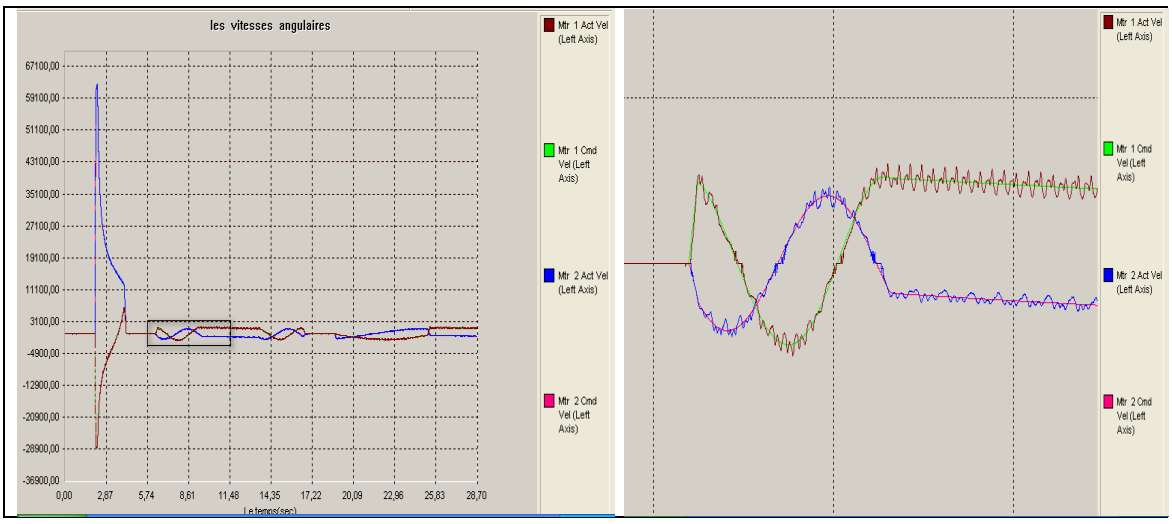


Figure 4.8 : Les vitesses angulaires des moteurs

IV.4.1.b Tracé du LOGO LCP:

Le deuxième exemple d'un dessin est le logo du LCP. Dans ce qui suit on suivra les mêmes étapes définit précédemment

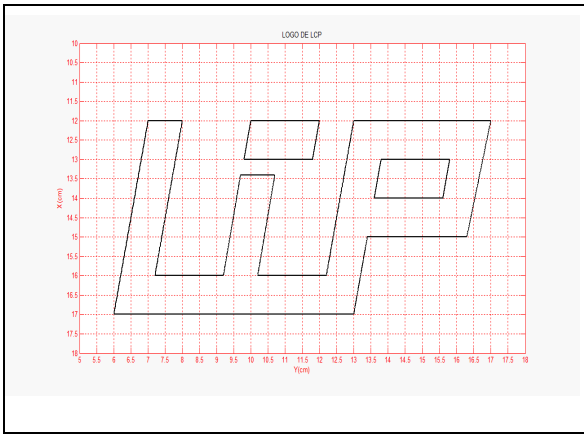


Figure 4.9 : le LOGO LCP sur MATLAB

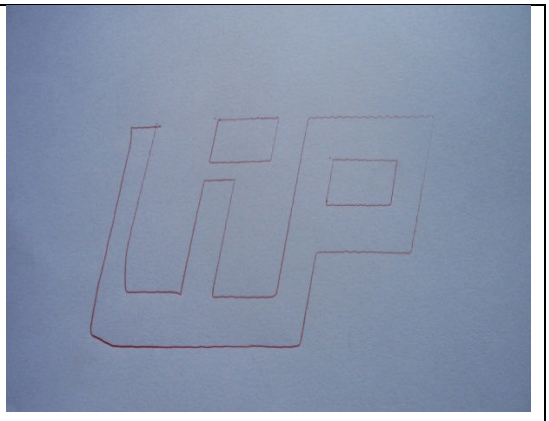


Figure 4.10 : Le LOGO LCP dessiné par le robot

Le programme de mouvement implémenté pour dessiner le logo LCP est le suivant :

```
DELETE GATHER                    ; effacer toutes les données des buffers
UNDEFINE ALL                     ; effacer toutes les définitions liées à l'axe
OPEN PROG 101 CLEAR
#1->I
#2->I
LINEAR                           ; interpolation linéaire
ABS                              ; le mode absolu
TA 100                           ; le temps d'accélération linéaire est de 100ms

TS 0                              ; Aucune accélération de S-curve
TM 3000                           ; le temps de mouvement est de 3000ms
P40=0                            ; pour soulever le vérin
DWELL2000                        ; arrêter le mouvement 2 s pour s'assurer que le vérin
                                 est complètement rentré
X 120 Y 70
P40=1                             ; pour descendre le vérin
DWELL 2000                       ; arrêter le mouvement 2 s pour s'assurer que le vérin est
                                   complètement sorti
X 170 Y 60                        ; pour aller à la position x=17cm ,y=6cm
X 170 Y 130
X 150 Y 134
X 150 Y 163
X 120 Y 170
X 120 Y 130
X 160 Y 122
X 160 Y 102
```

```
X 134 Y 107
X 134 Y 97
X 160 Y 92
X 160 Y 72
X 120 Y 80
X 120 Y 70
P40=0
DWELL 2000
X 120 Y 100
P40=1
DWELL 2000
X 130 Y 98
X 130 Y 118
X 120 Y 100
X 120 Y 120
X 120 Y 100
P40=0
DWELL 2000
X 130 Y 138
P40=1
DWELL 2000
X 140 Y 136
X 140 Y 156
X 130 Y 158
X 130 Y 138
P40=0
DWELL 2000
X 359 Y 0
CLOSE
```

Lors de l'exécution du programme relatif à cette tâche, nous avons relevé les positions et les vitesses des moteurs, les profils de ces derniers sont donnés aux Figures 4.11 et 4.12.

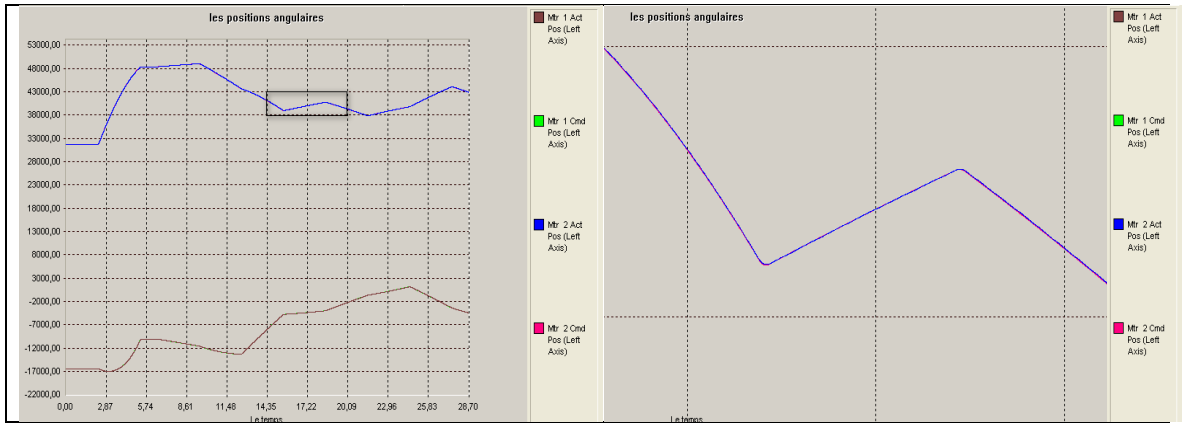
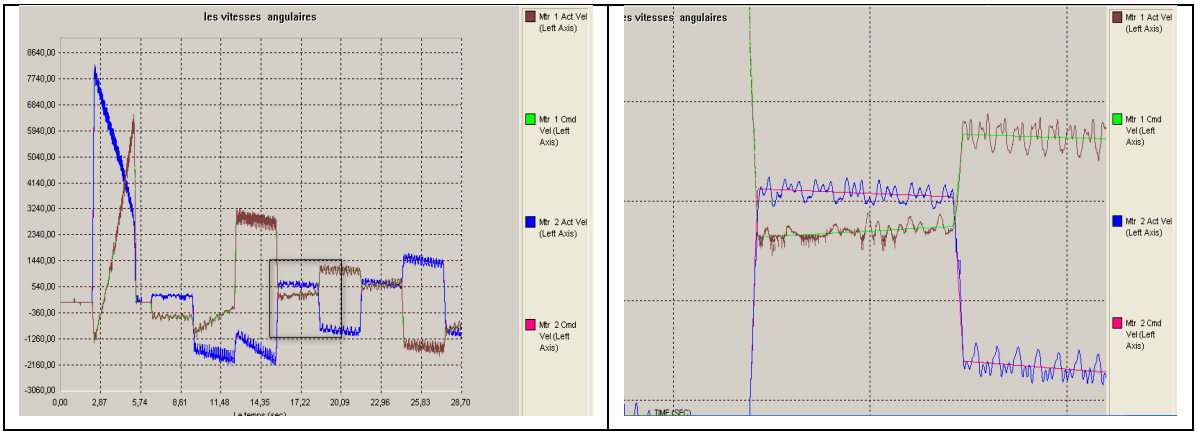


Figure 4.11 : Les positions angulaires des moteurs



Figures 4.12 : Les vitesses angulaires des moteurs

On observe que les positions des deux moteurs suivent les références imposées, ceci prouve un bon choix des coefficients du régulateur. , et le profil de vitesse nous montre que les modèles cinématiques calculés sont juste.

Gestion de programmes :

Pour pourvoir choisir entre les différentes tâches à exécuter (dans notre cas le symbole à dessiner), il est nécessaire d'utiliser un gestionnaire des programmes cette fonction est dévolue aux PLC (voir chapitre 2).

Si on associe la variable P50 pour activer le 1er programme et P51 pour activer le 2eme programme et on nomme B100 le programme correspondant au tracer du bras manipulateur, B101 le programme correspondant au tracer du logo du LCP

Le programme PLC est le suivant :

```
CLOSE
DELETE GATHER
OPEN PLC 1 CLEAR
IF (P60=0)
      CMD"#1J/"
      CMD "#2J/"
Q10=1
       I5150=1
        P60=1
ENDIF

           SENDS"SI VOUS VOULEZ dessiner  le bras manipulateur  TAPEZ P50=1''
           SENDS"SI VOUS VOULEZ dessiner  le LOGO du LCP  TAPEZ P51=1''
IF (P50=1)

          CMD"&1B100R''
ENDIF
```

```
IF (P51=1)

        CMD"&1B101R''
ENDIF

CLOSE
```

Pour lancer l'exécution de ce PLC on tape « ENABLE PLC 1», dans le terminal et pour l'arrêter on tape « DISABLE PLC 1»

IV.4.2 L'écriture des mots et des phrases

IV.4.2.a Principe et dimensionnement des lettres et chiffres

Pour l'écriture des lettres et des chiffres par le robot, on équipe l'effecteur d'un stylo à feutre par un stylo, puis nous procédons de la manière suivante :

Pour l'écriture de chacun des caractères de l'alphabet et de chacun des 9 chiffres, le sous-programme y afférant été développé, donc au final, nous disposons 37 sous programmes à implémenter (26 lettres, 10 chiffres, 1 réservé à l'espace blanc).Par conséquent lors de l'écriture d'un mot (ou d'une phrase), les sous programmes relatifs aux caractères constituants ce mot (ou cette phrase) vont être appelés dans leur ordre d'apparition dans le mot (ou la phrase).

Chaque lettre est dimensionnée dans un rectangle (2cm*1.5cm) et les chiffres (2cm*1cm), la forme des caractères sont sous la norme Calibri relative à l'OFFICE 2010 comme le montre les figures suivantes où juste quelques exemples sont représenté pour éviter d'encombrer notre document :

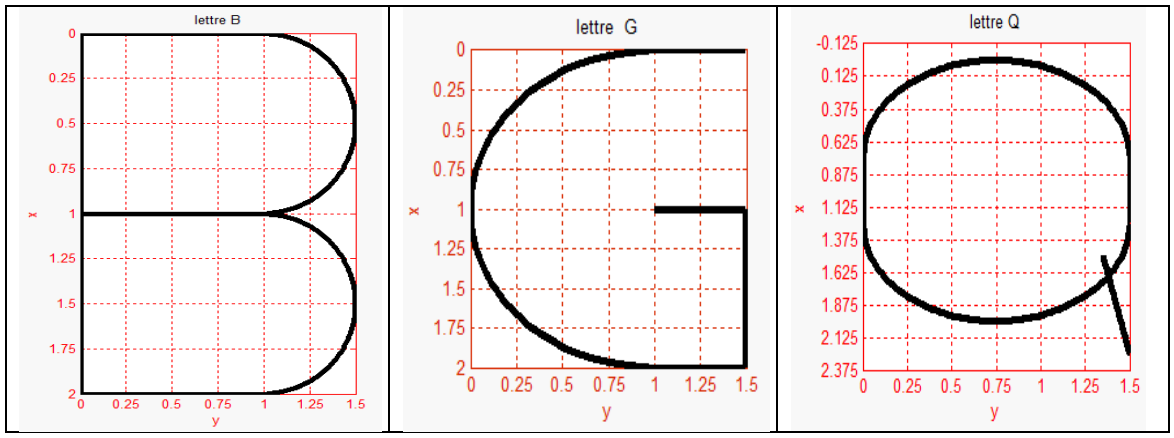


Figure 4.13 : Dimensionnement des caractères

Chaque caractère est un sous-programme dimensionné dans un repère relatif pour que le programme principal puisse les appeler en mode incrémental.

IV.4.2.b Structure générale des sous-programmes :

La forme générale est la suivante

```
OPEN PROG 1 CLEAR
TA 100
TS 10 ; pour définir la vitesse d'exécution
TM 1000
LINEAR
INC                    ; exécuter  la tâche en mode incrémental
{
Instructions de sous-programme
}
CLOSE
```

Si on prend l'exemple de lettres (R et Q) montrées dans les figures précédentes, les programmes correspondants sont les suivants :

```
********************** LETTRE Q **********************
OPEN PROG 17 CLEAR
TA 100
TS 10
TM 1000
LINEAR
INC
        X -7.5 Y 0
        P40=1
```

```
        X -5 Y 0
        CIRCLE1
        F 10
        X 0  Y 15 R 7.5
        X 5   Y 0
        X 0  Y -15 R 7.5
        P40=0
        X 7.5 Y 14.5
        P40=1
        X -5 Y -1
        P40=0
        X 5  Y 1.5
CLOSE
```

```
********** LETTRE R **********
OPEN PROG 18 CLEAR
TA 100
TS 10
TM 1000
LINEAR
INC
        P40=1
        X -20 Y 0
        X 0 Y 10
        CIRCLE1
        F 10
        X 10 Y 0 R 5
        X 0 Y -10
        P40=0
        X 0 Y 10
        P40=1
        X 10 Y 5
        P40=0
CLOSE
```

```
********** CHIFFRE 8 **********
OPEN PROG 35 CLEAR
TA 100
TS 10
```

```
TM 1000
LINEAR
INC
        P40=1
        X -20 Y 0
        X 0 Y 10
        X  20 Y 0
        X 0 Y -10
        P40=0
        X -10  Y 0
        P40=1
        X 0 Y 10
        P40=0
        X 10  Y 0
CLOSE
```

Les sous programmes sont nommés comme suit :

Lettre A → sous-programme 1
Lettre B → sous-programme 2
…..
…..
……
Lettre Z → sous-programme 26
Chiffre 0 → sous-programme 27
….
….
….
Chiffre 9 → sous-programme 36
Espace → sous-programme 37

IV.4.2.c Structure générale du programme principale

Selon la phrase voulue à écrire on va appeler les sous-programmes des lettres et des chiffres constituants la phrase comme suit :

```
DELETE GATHER
UNDEFINE ALL
OPEN PROG 101 CLEAR
#1->I
#2->I
```

```
LINEAR
ABS
TA 150
TS 0
TM  1000
X 310 Y 40
INC
CALL {constante}              ; constante : sous-programme N….
CALL {constante}              ; constante : sous-programme N….
CALL {constante}              ; constante : sous-programme N….
…..
…..
CALL {constante}              ; constante : sous-programme N….
CLOSE
```

IV.4.2.d Application

Comme application, on a choisi la phrase AUTO 2012, la figure suivante montre la forme dessinée sur MATLAB avec son dimensionnement:

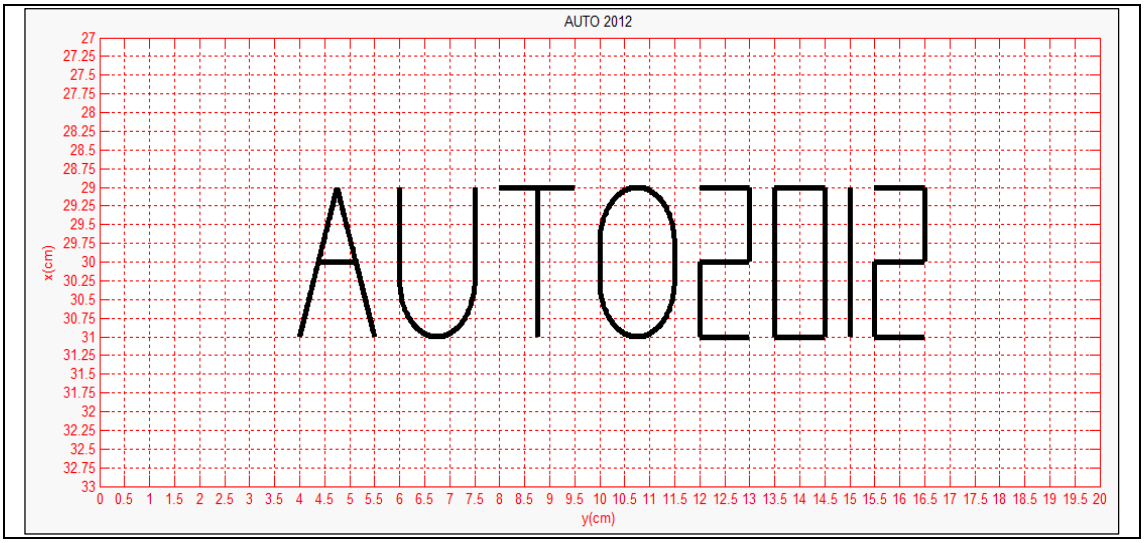


Figure 4.14 : AUTO 2012

Le programme principal implémenté est le suivant :

```
DELETE GATHER
UNDEFINE ALL
OPEN PROG 101 CLEAR
#1->I
#2->I
LINEAR
```

```
ABS
TA 150
TS 0
TM  1000
X 310 Y 40
INC
CALL 1
CALL 37
CALL 21
CALL 37
CALL  20
CALL 37
CALL  15
CALL  37
CALL  29
CALL  37
CALL 27
CALL  37
CALL  28
CALL  37
CALL  29
CLOSE
```

Les profils de vitesse et de position sont illustrés sur les figures 4.15 et 4.16 suivantes :

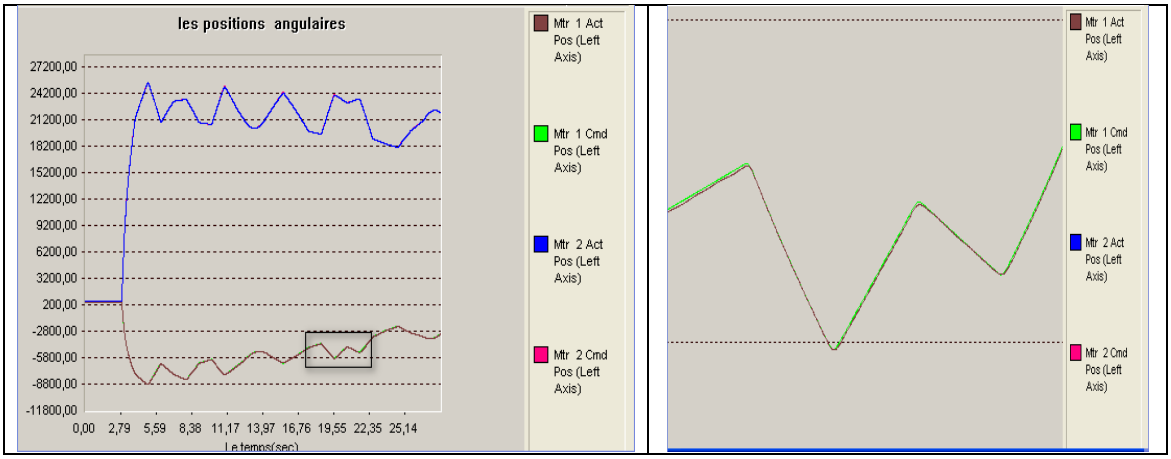


Figure 4.15: Les positions angulaires des moteurs

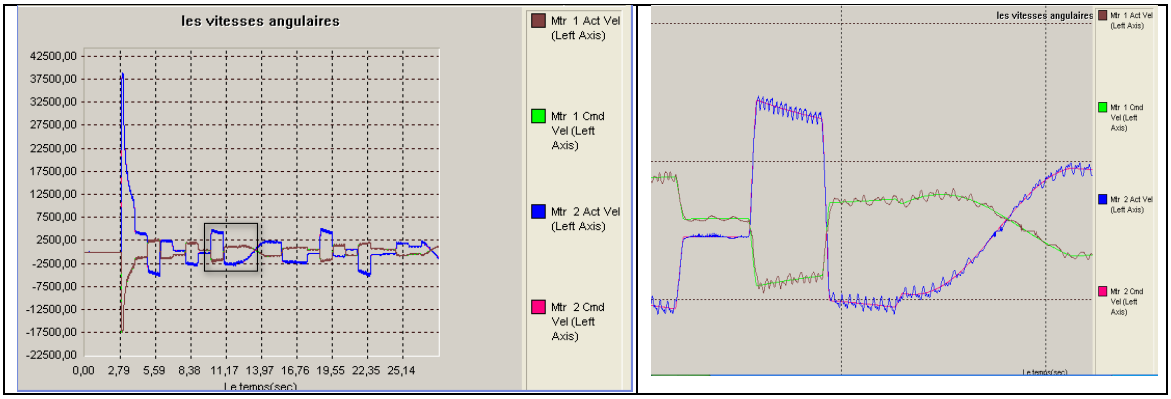


Figure 4.16: Les vitesses angulaires des moteurs

IV.4.3 La Frappe sur un clavier pour la saisie

IV.4.3.a Planification de la tâche :

Cette troisième tâche permet d'effectuer la frappe sur un clavier AZERTY pour saisir des mots, des phrases ou des textes sur un micro-ordinateur. Pour cela, il faut que le clavier soit bien positionné dans l'espace du travail du robot. Donc chaque robot a ses limites, tout dépend de la longueur des liaisons (voir chapitre 1), et ça pour utilisation soit dans l'industrie ou bien pour une utilisation personnelle.

La procédure mise en œuvre consiste à considérer chaque touche du clavier comme une position dans l'espace opérationnel du robot comme le montre la Figure 4.17.

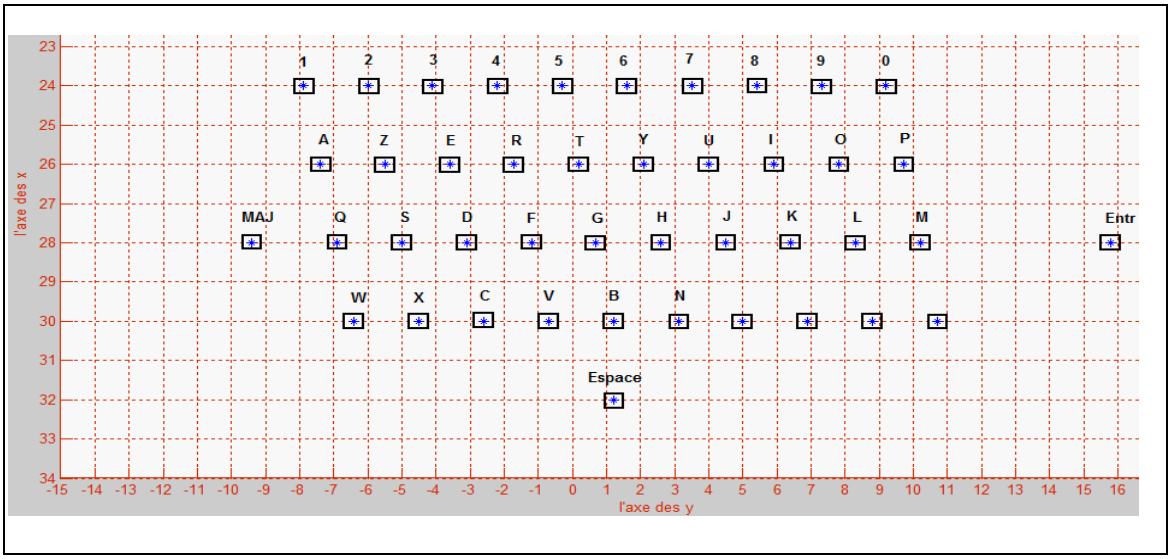


Figure 4.17: Disposition du clavier dans l'espace opérationnel (OXY)

Coordonnées des différentes touches du clavier

Ce positionnement judicieux des touches dans l'espace opérationnel assure à toutes les positions d'être incluses dans l'espace de travail et permet donc d'éviter les singularités.

A chaque position est associé un sous-programme mémorisant la position exacte dans le repère absolu, et lors de l'exécution d'un programme d'écriture ces sous programmes sont appelés.

Le tableau 4.1 donne les coordonnées exactes de chaque touche dans le repère absolu de l'espace du travail (OXY).

Touche	X (cm)	Y (cm)	Touche	X (cm)	Y (cm)
0	24	9.4	**Q**	28	-6.9
1	24	-7.9	**S**	28	-5.0
2	24	-6	**D**	28	-3.1
3	24	-4.1	**F**	28	-1.2
4	24	-2.2	**G**	28	0.7
5	24	-0.3	**H**	28	2.6
6	24	1.8	**J**	28	4.5
7	24	3.7	**K**	28	6.4
8	24	5.6	**L**	28	8.3
9	24	7.5	**M**	28	10.2
A	26	-7.4	**W**	30	-6.4
Z	26	-5.5	**X**	30	-4.5
E	26	-3.6	**C**	30	-2.6
R	26	-1.7	**V**	30	-0.7
T	26	0.2	**B**	30	1.2
Y	26	2 .1	**N**	30	3.1
U	26	4.0	**MAJ**	28	-9.4

I	26	5.9	**ENTREE**	28	15.8
O	26	7.8	**ESPACE**	32	1.2
P	26	9.7			

Tableau. 4.1 : Coordonnées des différentes touches du clavier

IV.4.3.b Structure du programme :

```
                    ******************** lettre A ********************
OPEN PROG 1 CLEAR
{
Instruction du programme
}
CLOSE
…….
……..
                    ******************** lettre Z********************
OPEN PROG 26 CLEAR
{
Instruction du programme
}
CLOSE
                    ******************** Espace********************
OPEN PROG 27 CLEAR
{
Instruction du programme
}
CLOSE
                    ******************** Entrée ********************
OPEN PROG 28 CLEAR
{
Instruction du programme
}
CLOSE
                  ******************** MAJUSCULE ********************
OPEN PROG 29 CLEAR
{
Instruction du programme
}
CLOSE

                  ******************** le chiffre 0********************
OPEN PROG 30 CLEAR
{
```

```
Instruction du programme
}
CLOSE
……
……..
                    ********** le chiffre 9**********
OPEN PROG 39 CLEAR
{
Instruction du programme
}
CLOSE
```

Donc pour taper une phrase, il suffit donc d'appeler les sous programmes correspondant aux lettres dans l'ordre d'appellation dans la phrase.

Exemple :
On prend comme exemple la phrase suivante :

« JE SUIS UN INGENIEUR »

Le programme d'exécution de cette tâche est le suivant :

```
CLOSE
DELETE GATHER
UNDEFINE ALL
OPEN PROG 100 CLEAR
#1->I
#2->I
LINEAR
ABS
TA150
TS0
TM 1000
CALL10
CALL5
CALL27
CALL19
CALL21
CALL9
CALL19
CALL27
CALL9
CALL14
CALL7
CALL5
CALL14
CALL9
CALL5
```

```
CALL21
CALL18
CLOSE
```

Les profils de position et de vitesse sont donnés sur les figures 4.18 et 4.19.

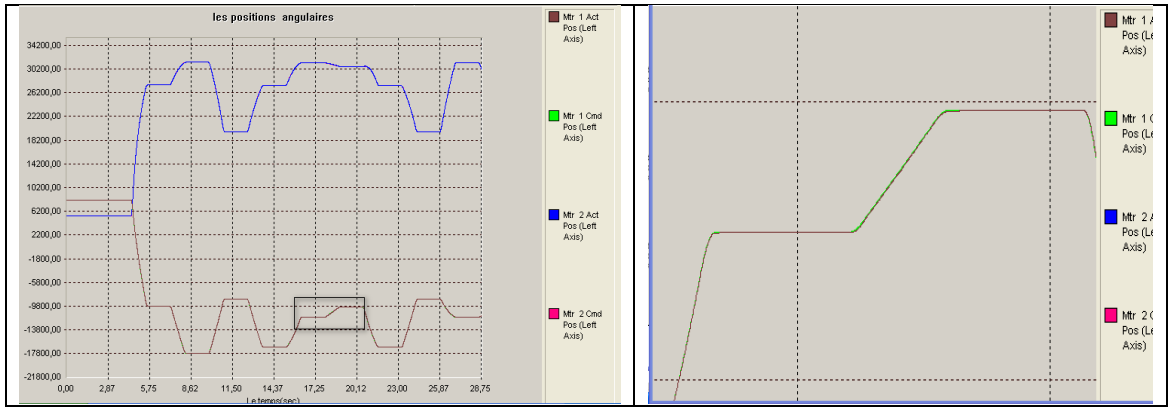


Figure 4.18: Les positions angulaires des moteurs

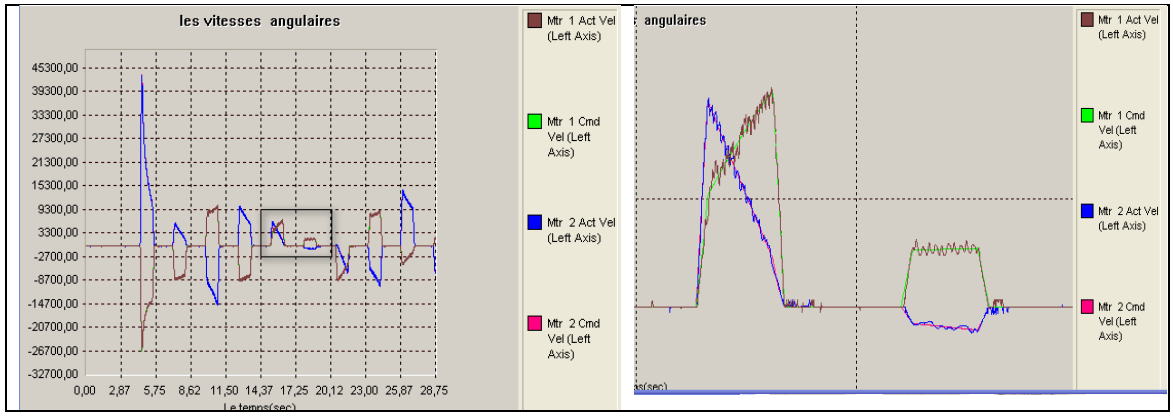


Figure 4.19: Les vitesses angulaires des moteurs

IV.4.4 Pick and Place

IV.4.4.a Planification de la tâche

L'une des tâches les plus simples à programmer est la tâche *Pick and place* expression anglo-saxonne qui veut dire saisir et poser. C'est en fait une tâche de positionnement où le robot doit passer par un certain nombre de points dans l'espace opérationnel. Le but est de déplacer des objets d'une position à une autre. La trajectoire entre les deux points peut être quelconque en cas d'absence d'obstacles, dans ce type de tâche, l'outil employé est un dispositif servant à saisir des objets [2] :

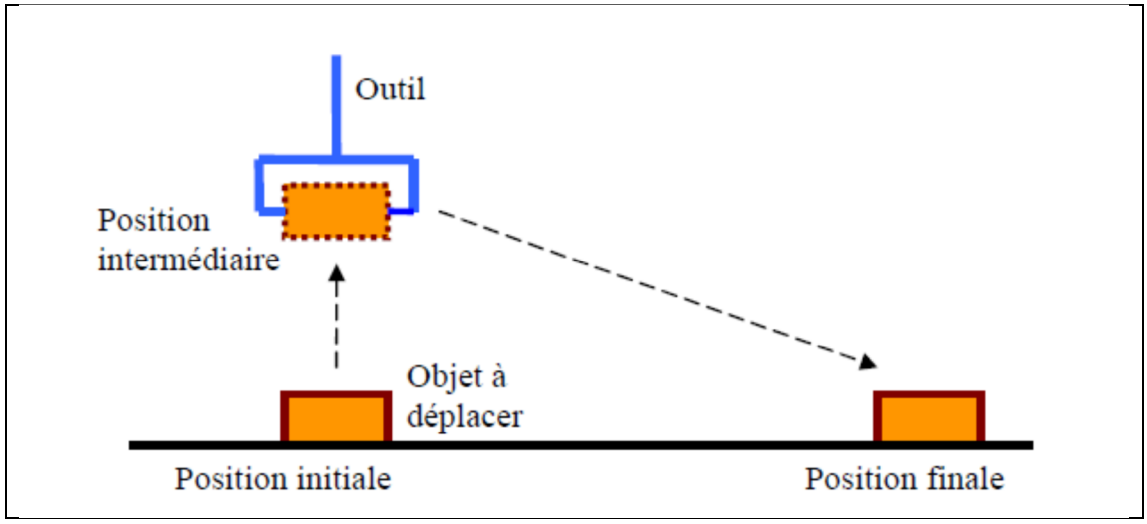


Figure 4.20: La tâche Pick and place

Pour déplacer un objet posé sur un plan horizontal d'une position initiale à une position finale donnée, on doit nécessairement passer par deux étapes :

- Déplacer verticalement, à l'aide du vérin pneumatique, l'objet sachant que la course du vérin qu'on dispose est de 5 cm donc le vérin assure l'action Pick au début et Place à la fin.
- Après avoir soulevé l'objet et pour atteindre la position finale, le robot va suivre soit une trajectoire donnée par le programmeur soit une trajectoire quelconque.

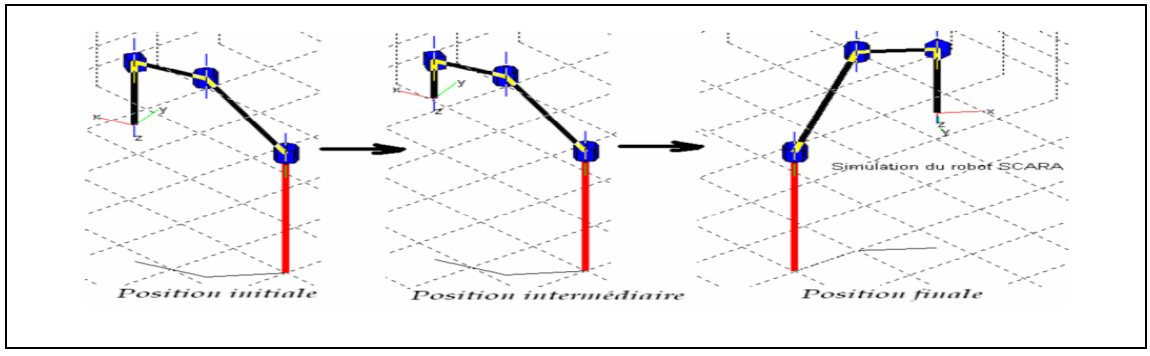


Figure 4.21: les positions de Pick and place

IV.4.4.b Commande de la pince

Pour la commande de la pince, c'est une des sorties de la DSP réservée à la commande d'axe qui est utilisée pour le pilotage du moteur relatif à la pince en plus des programmes PLC prévus pour cette opération. Ce moteur est actionner dans un sens de rotation pour l'ouverture de la pince et en sens inverse pour sa fermeture) et il est mis dans un état de repos (au moment du déplacement horizontal).

Le PLC utilisé pour la gestion du sens de rotation du moteur est le suivant :

```
Open PLC 1 Clear
            IF (P50=1)              ; vérifier dans le programme principale P50
CMD"#3J+"                           ; pour ouvrir la pince

Else
CMD"#3k"                            ; arrêter le moteur 3

            ENDIF
            IF (P51=1)              ; vérifier dans le programme principale P51
CMD"#3J-"                           ; pour fermer la pince

Else
CMD"#3k"                            ; arrêter le moteur 3

            ENDIF

Disable PLC 1
Close
```

Ce PLC sera lancé en parallèle avec l'exécution d'un programme de mouvement et à chaque variation des variables P dans le programme, le PLC va commander le sens de rotation du moteur.

IV.4.4.c Programmation de la tâche :

La trajectoire à suivre est illustré sur la figure 4.22.

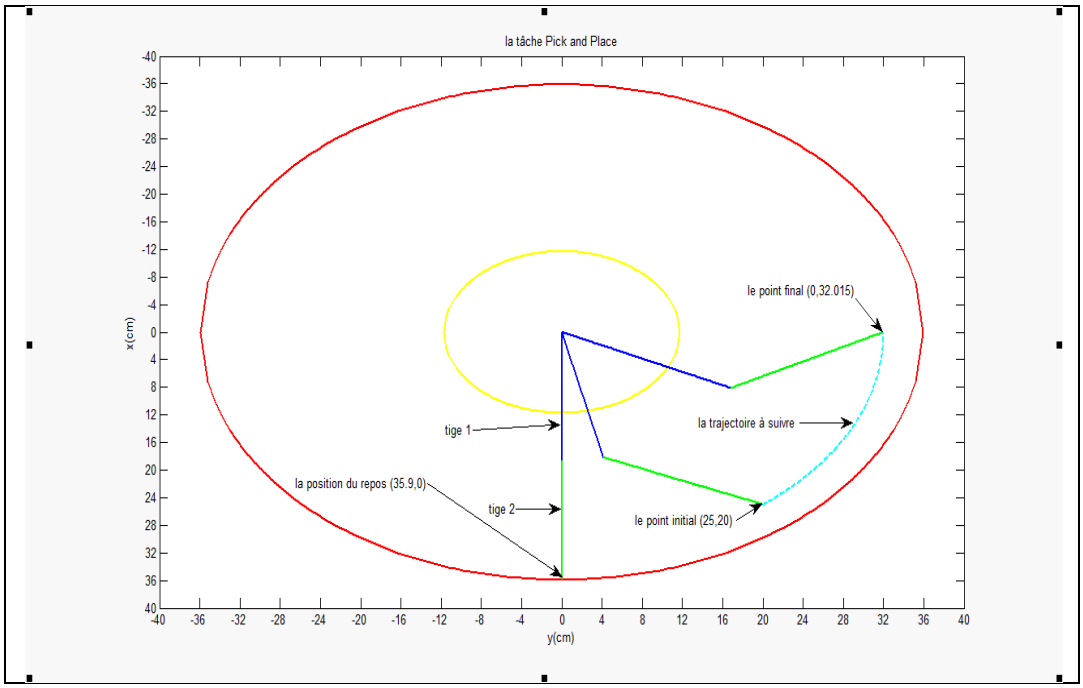


Figure 4.22 : Planification de la tâche Pick and Place

Le programme implémenté est le suivant :

```
DELETE GATHER                ; effacer toutes les données des buffers
UNDEFINE ALL                 ; effacer toutes les définitions liées à l'axe
OPEN PROG 102 CLEAR
#1->I
#2->I
LINEAR                       ; interpolation linéaire
ABS                          ; le mode absolu
TA 150                       ; le temps d'accélération linéaire est de 150ms
TS 0                         ; Aucune accélération de S-curve
TM 1000                      ; le temps de mouvement est de 1000ms
P40=0                        ; pour soulever le vérin
    DWELL 2000               ; arrêter le mouvement  2 s pour s'assurer que le vérin est
                                 complètement rentré
X 359 Y 0
X 250 Y 200
P50=1                            ; pour  ouvrir la pince
DWELL 2000                   ; arrêter le mouvement  2 s pour s'assurer que la pince est
complètement ouverte
P50=0                            ; couper l'alimentation du moteur 3
P40=1                        ; pour descendre  le vérin
```

```
 DWELL 2000                    ; arrêter le mouvement  2 s pour s'assurer que le vérin est
                                 complètement sorti
P51=1                          ; pour  fermer  la pince
 DWELL 2000                    ; arrêter le mouvement  2 s pour s'assurer que la pince s'est fermé
P51=0                          ; couper l'alimentation du moteur 3

P40=0                          ; pour soulever  le vérin
DWELL 2000                     ; arrêter le mouvement  2 s pour s'assurer que le vérin est
complètement rentré

CIRCLE1
F 50
X0 Y 320.15 R 32.015
P40=1                          ; pour descendre  le vérin
DWELL2000                      ; arrêter le mouvement  2 s pour s'assurer que le vérin est
complètement sorti
P50=1                          ; pour  ouvrir la pince
DWELL2000                      ; arrêter le mouvement  2 s pour s'assurer que la pince est
complètement ouverte
P50=0                          ; couper l'alimentation du moteur 3

X 359 Y 0                      ; pour que le robot revient à sa position du repos
CLOSE
```

Les profils de la position et de la vitesse sont donnés sur les figures 4.23 et 4.24.

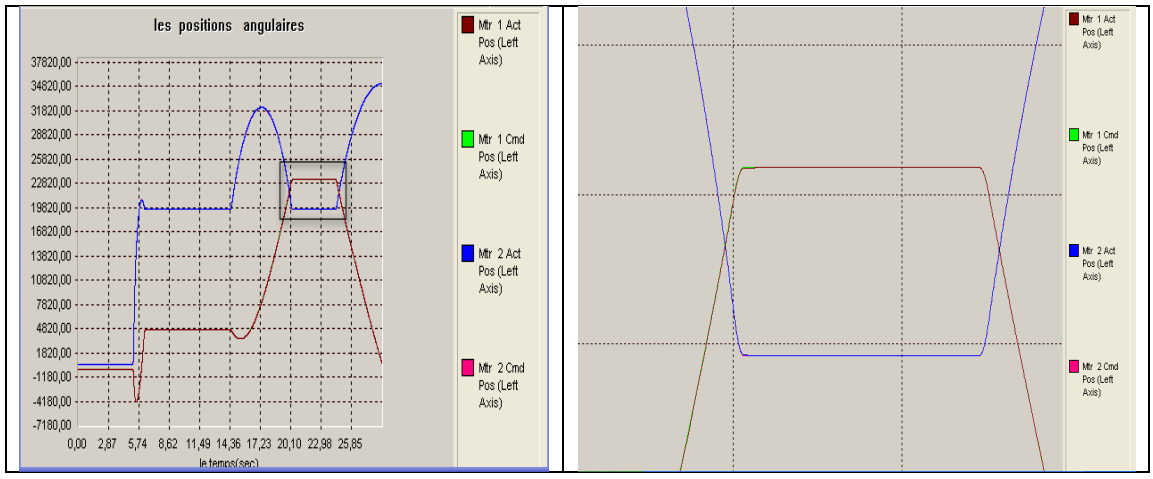


Figure 4.23: Les positions angulaires des moteurs

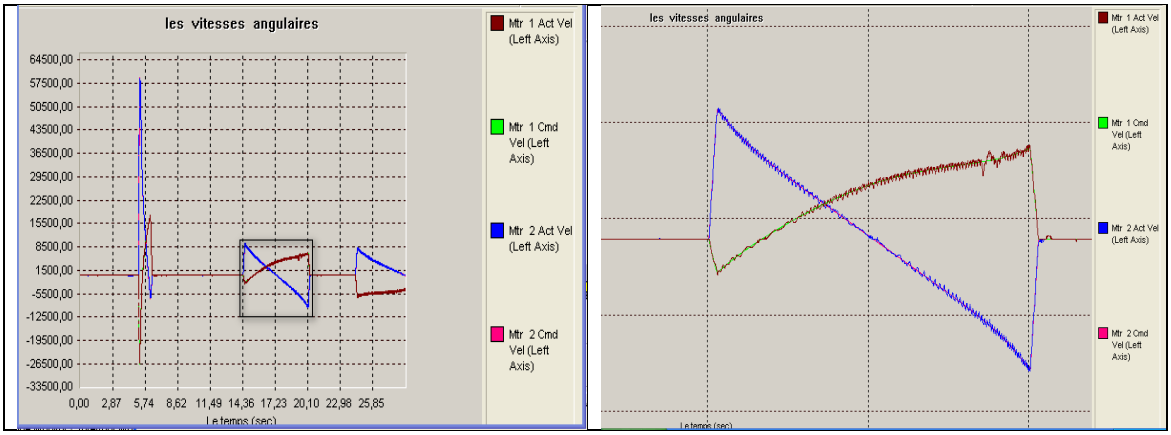


Figure 4.24 : Les vitesses angulaires des moteurs

IV.5 Conclusion

Dans ce chapitre, nous avons planifié et programmé plusieurs tâches pour le robot SCARA modifié en mettant à profit les logiciels de programmation de la DSP UMAC. Ainsi, nous avons pu effectuer avec ce robot les tache suivantes : traçage de courbes, écriture des lettres de l'alphabet et des chiffres, la frappe automatique sur un clavier pour la saisie de texte sur micro-ordinateur et enfin l'opération « Pick and Place ». Cependant, le test de poursuite de trajectoire a révélé les insuffisances de cette structure qui sont lié surtout à :

- une fixation du vérin insuffisamment solidaire à la liaison ;
- le plan de travail n'est pas vraiment horizontal ;
- un jeu au niveau du stylo feutre.

Conclusion générale et Perspectives

Dans ce projet, notre objectif consistait en l'amélioration de la structure du robot existant au sein du LCP et la programmation de plusieurs tâches pour le robot modifié.

Pour entamer les travaux, nous avons commencé par un diagnostic des défauts existants, et ceci en faisant appel aux compétences des ingénieurs en mécanique. Le constat était qu'il faut placer des roulements au niveau des articulations et de diminuer la longueur des tiges et la hauteur du robot. Pour assurer un déplacement vertical et une bonne rapidité d'exécution, nous avons opté pour un actionneur pneumatique. Finalement, nous avons installé une pince mécanique pour que le robot puisse effectuer la tâche « Pick and Place ». Une fois les modifications envisagées ont été effectuées, nous avons déterminé le modèle géométrique et le modèle cinématique direct et inverse relatifs à la nouvelle structure.

Ensuite, une description du matérielle et du logiciel de cette interface de commande (DSP UMAC) a été présenté. Dans cette partie, nous avons explicité le fonctionnement général de l'UMAC ainsi que ses différents programmes. Sa programmation est rendu aisée grâce à la disponibilité du logiciel PEWIN32PRO conçu par la firme DELTA TAU.

Une exécution satisfaisante des différents programmes de mouvement nécessite un bon asservissement des actionneurs. Pour cela, nous avons exploité le régulateur PID+Feed Forward implémenté sur la carte DSP (l'interface UMAC). Aussi, nous avons procédé à la détermination des coefficients de ce régulateur conduisant à des performances satisfaisantes.

Ensuite, diverse tâches ont été planifiées, programmées et implémentées sur la carte UMAC. En effet, en exploitant judicieusement ce robot, ce dernier a pu réaliser le tracer des trajectoires, le tracer du LOGO du LCP, l'écriture des lettres de l'alphabet et des neuf chiffres de plus, il a pu réaliser la frappe sur clavier pour la saisie de texte vers un micro-ordinateur. La dernière tâche réalisée et qui est fréquente dans l'industrie, était celle du « Pick and Place »

Cependant, la réalisation de notre travail a souffert de la difficulté majeure représentée par un manque de matériel et sa faible disponibilité sur place. Ceci, nous a empêché d'atteindre tous nos objectifs.

Pour réaliser un asservissement du mouvement vertical dans l'espace opérationnel, nous préconisons d'utiliser un vérin à plusieurs chambres au lieu du vérin à deux positions.

La pince mécanique peut être améliorée en concevant une pince dont on peut maitriser le mouvement des doigts pour assurer une bonne saisie des pièces de différentes formes.

Concevoir un contrôleur spécifique autre que l'interface UMAC ou encore la mise en place d'une interface virtuelle permettant de commander le robot et de connaître sa position en temps réel.

L'ajout, de capteurs plus perfectionnés, peut être envisagé, par exemple, une caméra pour asservissement visuel du robot.

Généralités sur les vérins pneumatiques [10]

- **Principe de fonctionnement**

Les systèmes automatisés qui mettent en œuvre des actionneurs pneumatiques sont nombreux dans les secteurs industriels.

Un vérin pneumatique est un actionneur qui permet de transformer l'énergie de l'air comprimé en un travail mécanique. Un vérin pneumatique est soumis à des pressions d'air comprimé qui permettent d'obtenir des mouvements dans un sens ou dans un autre. Les mouvements obtenus peuvent être linéaires ou rotatifs, Cet actionneur de conception robuste et de simplicité de mise en œuvre est utilisé dans toutes les industries manufacturières.

Figure A.1: vérin pneumatique

- **Constitution d'un vérin pneumatique**

Un vérin se compose d'un piston muni d'une tige se déplaçant librement à l'intérieur d'un tube. Pour faire sortir la tige, on applique une pression sur la face avant du piston, et sur la face arrière pour faire rentrer la tige.

Certains vérins disposent d'amortisseurs afin d'obtenir un ralentissement en fin de mouvement de façon à éviter un choc du piston sur le nez ou le fond du vérin. On pourra également équiper les vérins de dispositifs de contrôle de mouvement tels que régulateur de vitesse et capteurs de position magnétique.

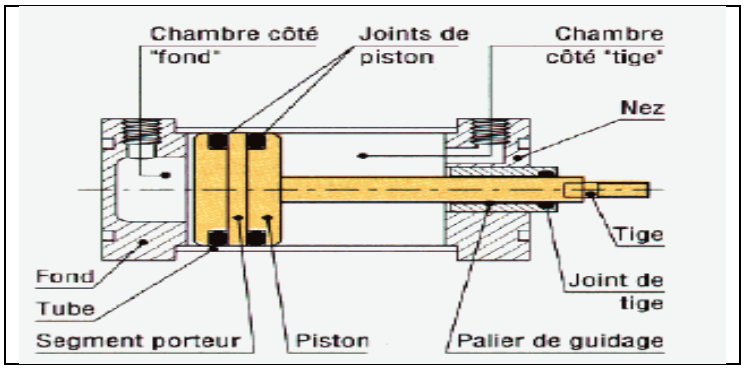


Figure A.2 : vue en coupe d'un vérin pneumatique

Le vérin pneumatique est utilisé avec de l'air_comprimé_entre 2 et 10 bars_dans un usage courant. Simple à mettre en œuvre, il est très fréquent dans les systèmes automatisés industriels.

Les vérins standards varient suivant leurs alésages et leur compacité auxquels on peut adjoindre différentes options : fixation universelle, tige anti-rotation, salle blanche).Les vérins se définissent aussi suivant leur fonction : bloqueur, avec table, guidé, à tige anti-rotation et sans tige. Pour des applications spécifiques, les vérins spéciaux offrent différentes adaptations : bride rotative, stoppeur, sinusoïde, faible vitesse, faible frottement, haute précision, en acier inox, 3 positions, séparateurs.

- **Types des vérins**

  - **Les vérins pneumatiques à simple effet**

Ce sont des vérins qui effectuent un travail dans un seul sens. Ils permettent soit de pousser soit de tirer une charge, exclusivement. Seules les positions extrêmes sont utilisées avec ce type de vérin. Un vérin pneumatique à simple effet n'a qu'une seule entrée d'air sous pression et ne développe un effort que dans une seule direction. La course de retour à vide est réalisée par la détente d'un ressort de rappel incorporé dans le corps du vérin.

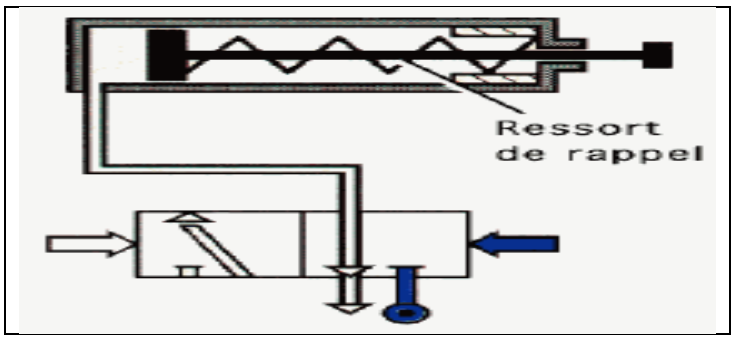


Figure A.3: vérin simple effet avec un distributeur

  - **Vérin double effet**

Un vérin double effet a deux directions de travail. Il comporte deux orifices d'alimentation de pression cette dernière est appliquée alternativement de chaque côté du piston ce qui entraîne son déplacement dans un sens puis dans l'autre. On vérifiera que le vérin ne sera pas soumis aux effets de multiplication de pression qui pourraient le faire éclater du côté de sa tige. Associé à une servovalve ou un distributeur à commande proportionnelle, ainsi qu'un capteur de position ou des capteurs de pression, le vérin

devient alors un servo-vérin. Cet actionneur est utilisé dans tous les servomécanismes, ils sont souvent équipés d'amortisseurs de fin-de-course qui évitent les chocs du piston.

Figure A.4 : vérin double effet avec un distributeur

- **Servo-vérin**

C'est un vérin asservi. Il est équipé soit d'un distributeur à commande proportionnelle soit d'une servovalve et d'un ou plusieurs capteurs. Selon sa qualité, il sera équipé de joints à faible frottement ou de paliers hydrodynamiques, voire de paliers hydrostatiques pour les plus performants. La réponse d'un servo-vérin dépend de ses dimensions, de la masse embarquée, de la qualité de l'organe de commande et de la longueur des circuits d'alimentation. Les régulateurs numériques aujourd'hui permettent d'améliorer la réponse des servo-systèmes hydrauliques. L'effort développé par un servo-vérin dépend des pressions résultantes dans les deux chambres, celles-ci créent des efforts antagonistes et sont dues aux pertes de charge générées par l'organe de contrôle qui laminent le fluide.

Les distributeurs pneumatiques [10]

- **Définition**

Ils ont pour fonction essentielle de distribuer le fluide dans des canalisations qui aboutissent aux chambres des vérins. Comme le contacteur associé à un moteur électrique, le distributeur est le pré-actionneur associé à un vérin pneumatique

- **Représentation schématique des distributeurs**

La représentation d'un distributeur s'effectue à l'aide de cases. Il y a autant de cases que de positions possibles. A l'intérieur des cases, on représente les voies de passage de l'air pour chacune des positions.

Pour caractériser un distributeur, il faut définir le nombre de voies ou d'orifices ainsi que le nombre de positions (exemple : distributeur 3/2 : ce distributeur comprend 3 orifices et 2 positions).

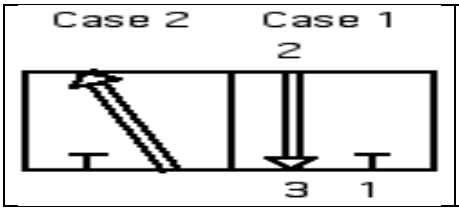


Figure A.5: Distributeur 3/2

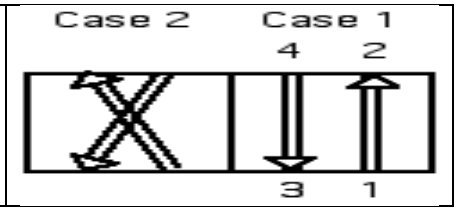


Figure A.6: Distributeur 4/2

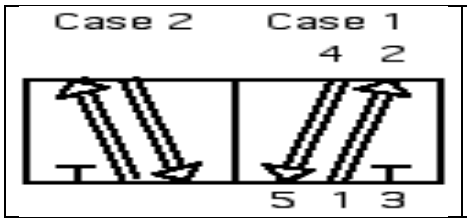


Figure A.7 : Distributeur 5/2

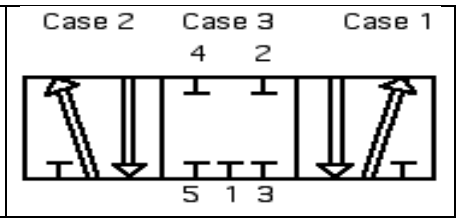


Figure A.8 : Distributeur 5/3 à centre fermé

- Si le vérin est à simple effet et ne comporte donc qu'un seul orifice à alimenter, on utilise alors un distributeur ne comportant qu'un seul orifice de sortie : distributeur 3/2 à trois orifices (pression, sortie, échappement) et à deux positions.
- Si le vérin est à double effet et comporte donc deux orifices sur lesquels il faut alterner les états de pression et d'échappement, on utilise alors un distributeur comportant deux orifices de sortie. Deux possibilités sont offertes :
  - ✓ Distributeurs 4 /2 à quatre orifices (pression, sortie1, sortie2, échappement1, échappement2) et deux positions.
  - ✓ Dans le cas particuliers où il est nécessaire d'immobiliser ou de mettre hors énergie le vérin double effet, on utilise un distributeur 5/3 (cinq orifices, trois positions) à centre fermé ou à centre ouvert.

On distingue plusieurs types de pilotage des distributeurs

Le pilotage monostable (figure A.9) où Le distributeur est rappelé dès la disparition du signal de pilotage par un ressort de rappel.

Le pilotage bistable (figure A.10) où Le distributeur garde sa position en l'absence du signal de pilotage (fonction mémoire).

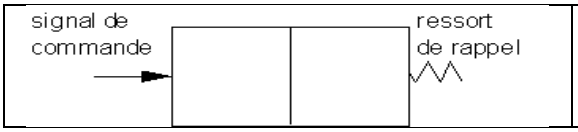


Figure A.9: distributeur monostable

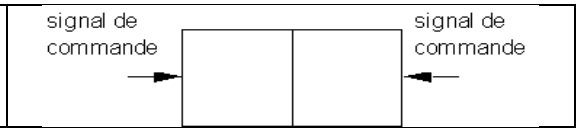


Figure A.10 : distributeur bistable

Les variables I

Variable *I*	**signification**
I0-I99	Installation générale de la carte
Ixx00-Ixx86	Configuration du moteur xx
Ixx87-Ixx99	Configuration du système de coordonnée xx
I900-I979	Encodeur 1-16 installés (dans les groupes de 5)
I980-I1023	Réservé pour de futur usage
I0	Le nombre d'adressage de la carte série
I1	Mode de Communications du Porte série
I2	Panneau de Commande invalidé
I3	Mode de prise de contact d'E/S
I4	La Somme des Communications validée
I5	Le PLC Programme "Marche/Arrêt"
I7	Nombre de cycles en position
I8	Période d'interruption en temps réel
I10	Temps servo d'interruption
I11	Temps de calcul du mouvement programmé
I12	Temps de calcul d'un essai en position
I13	Temps de segmentation du mouvement programmé
I15	Commande pour des fonctions de triglycéride d'utilisateur Degré/Radian
I16	Demande rotatoire d'amortisseur pour un point en marche
I17	Demande rotatoire d'amortisseur pour un point en arrêt
I18	Avertissement si le buffer est plein
I19	La période de réception de donnée
I20	Le choix du masque de réception de donnée
I21-I44	Les adresses 1-24 de la source de réception de données
I45	Le mode et l'endroit du buffer de réception de données
I46	Commande de la fréquence de la CPU
I47	L'adresse de la commande <CTRL-W>
I48	Les données servo d'EPRAM
I49	Les données de fond d'EPRAM
I50	Le mode rapide de la vitesse du mouvement
I51	Le tableau de compensation
Ixx00	Activer le moteur xx
Ixx01	PMAC commute le moteur xx
Ixx02	L'adresse de la sortie de commande (DAC) du moteur xx
Ixx03	L'adresse de la boucle de position du moteur xx
Ixx04	L'adresse de la boucle de vitesse du moteur xx
Ixx05	L'adresse de position principale du moteur xx
Ixx11	Arrêt du moteur xx après la limite d'erreur
Ixx12	Avertissement après la limite d'erreur du moteur xx
Ixx13	La position limite positive du moteur xx
Ixx14	La position limite négative du moteur xx
Ixx15	Le taux de décélération du moteur xx
Ixx16	La vitesse maximale du moteur xx
Ixx17	L'accélération maximale du moteur xx

Ixx19	L'accélération maximale d'essai du moteur xx
Ixx20	Le temps d'accélération du moteur xx
Ixx21	Le temps de S-courbure du moteur xx
Ixx22	La vitesse d'essai du moteur xx
Ixx23	La vitesse et la direction du moteur xx
Ixx30	Le gain proportionnel du régulateur PID pour le moteur xx
Ixx31	Le gain de dérivation du régulateur PID pour le moteur xx
Ixx32	Le gain Feedforward de vitesse du régulateur PID pour le moteur xx
Ixx33	Le gain de l'action intégrale du régulateur PID pour le moteur xx
Ixx34	Le mode d'intégration du moteur xx
Ixx35	Le gain Feedforward d'accélération du régulateur PID pour le moteur xx
Ixx36	Le coefficient N1 du filtre du moteur xx
Ixx37	Le coefficient N2 du filtre du moteur xx
Ixx38	Le coefficient D1 du filtre du moteur xx
Ixx39	Le coefficient D2 du filtre du moteur xx
Ixx40	Le gain du filtre de la position désirée du moteur xx
Ixx57	La limite de courant continu du moteur xx
Ixx58	Le niveau de défaut du courant intégré du moteur xx
Ixx60	La période de prolongation du cycle de servo du moteur xx
Ixx61	Le gain intégral de la boucle du courant du moteur xx
Ixx63	La limite d'intégration du moteur xx
Ixx67	La limite linéaire d'erreur de la position du moteur
Ixx68	Le gain de Feedforward de frottement du moteur xx
Ixx69	La limite de la commande de la sortie (DAC) du moteur xx
Ixx70	Le nombre de cycle de commutation du moteur xx
Ixx71	Comptes d'encodeur par cycles de commutation du moteur xx
Ixx72	L'angle de phase de commutation du moteur xx (xx=00-32)
Ix87	Le temps d'accélération du programme par défaut du système de coordonné x (x=1-8)
Ix88	Le temps de S-courbure du programme par défaut du système de coordonné x
Ix91	Le nombre de programme de fonctionnement par défaut du système de coordonné x
I900-I904	L'encodeur 1
I905-I909	L'encodeur 2
I910-I914	L'encodeur 3
I915-I919	L'encodeur 4
……..	……
I970-I974	L'encodeur 15
I975-979	L'encodeur 16

Les Commandes Utilisées

Les commandes en ligne

Les commandes en ligne sont utilisées directement à partir d'un terminal.

- **A :** pour arrêter tous les programmes de mouvement en cours d'exécution.
- **R :** pour commencer l'exécution d'un programme de mouvement.
- **J= :** pour faire tourner le moteur jusqu'au point final programmé après un l'arrêt s'est produit.
- **ABR {constante}** : pour arrêter le programme fonctionnant actuellement et exécuter l'autre programme spécifié par la constante.
- **ABS** : pour choisir le mode absolue pour les positions sur les axes dans un système de coordonnées.
- **ABS ({axe}, {axe}…, {axe})** : pour choisir le mode absolue pour les axes spécifie, où {axe} est une des lettres (X, Y, Z, A, B, C, U, V, W) représentant un axe ou bien R représentant un rayon.
- **{Axe}= {constante}**: pour donner la position sur cet axe.
- **B {constante}:**pour diriger le système de coordonnées adressé à un programme de mouvement.
- **R :** pour exécuter un programme de mouvement (ex : &1B10R, cette commande va exécuter le programme 10 associé au système de coordonné 1)
- **S :** pour exécuter une seule instruction d'un programme de mouvement (PROG et ROT) (ex : &1B10S, cette commande va exécuter la première instruction du programme 10 associé au système de coordonné 1, et on tape S pour la deuxième fois pour exécuter l'instruction suivante dans le programme …etc.

- **CLEAR** ou bien **CLR** : pour effacer le buffer actuellement ouvert.
- **CLEAR ALL** ou bien **CLR ALL** : pour Effacer tout mouvement fixe, cinématique, et des programmes de PLC non compilés
- **CLEAR ALL PLCS** ou bien **CLR ALL PLCS**: pour effacer les programmes PLC.
- **CLOSE** ou bien **CLS**: pour fermer le buffer actuellement ouvert.
- **CLOSE ALL** ou bien **CLS ALL**: pour fermer le buffer actuellement ouvert sur n'importe quel port.
- **DEFINE GATHER** : créer un buffer pour stocker les données.
- **DELETE ALL** ou bien **DEL ALL** : effacer tous les buffers permanents et provisoires définis.
- **DELETE ALL TEMPS** ou bien **DEL ALL TEMP** : effacer tous les buffers provisoires définis.
- **DELETE GATHER** ou bien **DEL GAT**: pour effacer les données se trouvant dans un buffer

- **DELETE PLCC {constante}** ou bien **DEL PLCC {constant}** : pour effacer le programme PLC compilé spécifié par la constante.
- **DISABLE PLC {constante}, {constante}, DIS PLC {constante}, {constante}, DISABLE PLC {constante}... {Constante}** ou bien **DIS PLC {constante}... {Constant} :** arrêter un programme PLC spécifié par la constante, avec {constante}=0 à 31 représentant le numéro du programme PLC. Ces commandes sont les mêmes que pour les PLCC .
- **ENABLE PLC {Constante}, {Constante}, ENA PLC {Constante}, {Constant}, ENABLE PLC {Constante}... {Constante}, ENA PLC {Constant}... {Constante} :** pour exécuter un programme PLC spécifié par la constante.
- **END GATHER** ou bien **ENDG**: arrêter la réception de données.
- **FRAX ({axe}, {axe}... {axe})**: pour spécifier la vitesse de chaque axe.
- **GATHER** : pour commencer la réception des données.
- **INC**: pour indiquer le mode incrémental pour le mouvement, c.-à-d. la position est donnée toujours par rapport au dernier point du mouvement.
- **INC ({axe},{axe}...):** pour indiquer le mode incrémental pour chaque axe spécifie par la lettre de l'axe
- **J!:** pour ajuster la position commandée du moteur au plus proche entier compte d'encodeur
- **J+** : pour tourner le moteur dans le sens positif sans arrêt
- **J-** : pour tourner le moteur dans le sens négatif sans arrêt
- **J/ :** pour arrêter le moteur
- **J:{constante}**: pour tourner le moteur, la constante donne le nombre d'incrément d'encodeur à compter avant que le moteur ne s'arrête
- **J=:** pour que le moteur revient à la position précédente
- **J=*** : pour tourner le moteur à une position donnée dans une variable M
- **K** : pour arrêter les moteurs et couper l'alimentation de ces derniers
- **LIST:** pour lister le contenu du buffer ouvert
- **LIST FORWARD, LIST INVERSE, LIST PROG {constante}, LIST PLC {constante} :** pour lister le contenue d'un programme géométrique direct, inverse, ou autre programme et un PLC respectivement. La constante indique le numéro du programme
- **LIST GATHER [{début}] [,{longueur }]:** pour rapporter les données qui sont dans un buffer
- **O {constante}** : pour ouvrir la boucle du moteur
- **PAUSE PLC {constant}:** pour faire pause à un programme PLC
- **Q :** pour arrêter un programme
- **SAVE :** pour sauvegarder les valeurs de toutes les variables dans la mémoire flash
- **UNDEFINE ALL** : pour effacer toutes les définitions dans les systèmes de coordonnées

Les commandes utilisées dans un programme

- **{axe}{donnée}…… {axe}{donnée} :** pour indiquer le déplacement sur chaque axe (ex : X 20 Y 12, X 2000, X (P1+P3)…etc.)
- **{axe}{donnée}:{donnée} …{axe}{donnée}:{donnée} :** pour indiquer le déplacement et la vitesse sur chaque axe (ex : X 100 :5, déplacement =100 et vitesse=5, X (P1) :(P2)…etc.),
- **ABS ({axe},{axe},..{axe}):** pour choisir le mode absolue lors du mouvement sur les axes (ex : ABS(X, Y), ABS(R), ABS (U, V, W)…etc.)
- **INC ({axe},{axe},..{axe}):** pour choisir le mode incrémental lors du mouvement sur les axes (ex : INC(X, Y), INC(R), INC (U, V, W)…etc.)
- **ADIS {constante}:** déplacement sur les axes X, Y et Z **(**ex : Q20=7.5, Q21=12.5, Q22=25, ADIS 20 équivalent à X 7.5 Y 12.5 Z 25)
- **AND ({condition}):** et conditionnel (ex : AND (M12=1))
- **CALL {donnée}:** pour appeler un sous-programme avec un return (ex :CALL 34…etc.)
- **CIRCLE1** ou bien **CIR1:** pour commander un mouvement circulaire dans le sens horaire
- **CIRCLE2**ou bien**CIR2:** pour commander un mouvement circulaire dans le sens antihoraire
- **CMD "{commande}", CMDS "{commande}", CMDP "{command}", CMDR "{command}" :** pour donner une commande dans un programme PLC(ex : CMD"&1B5R", CMD"#1J+"…etc.)
- **DISABLE PLC {constante}… {constante} :** pour arrêter l'exécution d'un programme PLC (ex : DISABLE PLC 1, DISABLE PLC 1,4…etc.)
- **ENABLE PLC {constante}… {constante} :** pour exécuter un programme PLC (ex : ENABLE PLC 1, ENABLE PLC 1,4…etc.)
- **DWELL {donnée}**ou bien **DWE {donnée} :** pour arrêter le mouvement d'une durée égal à la donnée (ex : DWELL 1000 arrêter le mouvement 1s)
- **IF ELSE ENDIF :** pour tester les variables pour exécuter d'autres instructions (ex : IF (P5=7) X 1000 ELSE X 2000 ENDIF)
- **WHILE ENDWHILE :** la boucle tant que
- **LINEAR** ou bien **LIN :** pour choisir un mode avec interpolation linéaire
- **N {constante} :** pour numéroter les lignes d'un programme, la constante est un entier de 0 à 262143
- **OR :** le ou conditionnel
- **PVT {donnée} :** pour choisir la durée de mouvement, la vitesse et le point final (ex : PVT 500 X 100 :1500, le temps de mouvement = 500ms, le point final 100 incrément d'encodeur et la vitesse final égal à 1500 incrément/s)
- **R {donnée} :** pour imposer le rayon d'un cercle
- **RAPID** ou bien **RPD :** pour choisir le mode rapide lors du mouvement
- **ADDRESS &{constante}#{constante}** ou bien **ADR &{constant}#{constant} :** pour adresser un moteur dans un système de coordonné (ex : ADR &1#2),elle est utilisée dans un programme PLC

- **F {donnée} :** pour indiquer la vitesse du mouvement dans un programme de mouvement (PROG et ROT) (ex : F 100, F (Q10), F (SIN (P8*P9))…etc.)

- **GOSUB {donnée}:** pour aller à une ligne spécifie par la donnée avec un return pour continuer l'exécution du programme de mouvement (ex : GOSUB 34, cette commande va aller à la ligne N 34 ensuite si elle rencontre un return elle revient à l'instruction qui vienne après GOSUB), elle est utilisée seulement dans un programme de mouvement (PROG)

- **GOTO {donnée} :** cette commande est comme GOSUB sauf ici sans return (ex : GOTO 5)

REFERENCES BIBLIOGRAPHIQUES

[1] Bernard BAYLE Université Louis Pasteur de Strasbourg Master Ingénierie et Technologie «Robotique» 2007/2008

[2] CHIKH LOTFI et ZIOUI NADJET, « Conception, réalisation et commande d'un

Robot manipulateur », Projet de fin d'études, ENP, 2006.

[3] E.DOMBRE ET W.KHALIL « Robot Manipulators Modeling, Performance Analysis and Control »

[4] DELTA TAU DATA SYSTEMS, INC «Hardware Reference Manual» janvier 2003

[5] Digital DNA from Motorola «DSP56303 User's» janvier 2001

IV.5.1 [6] DELTA TAU DATA SYSTEMS, INC «UMAC» Products Selection Guide Fall 2003

IV.5.2 [7] DELTA TAU DATA SYSTEMS, INC «User Manual» Turbo PMAC Septembre 2008

[8] BEN CHABANE Sofiane et DOUAIDI Oualid « Commande d'un robot manipulateur de type SCARA par l'interface UMAC », projet de fin d'études, ENP, 2011

[9] DELTA TAU DATA SYSTEMS, INC« Quick Reference» octobre 2003

[10] B. HEMICI cours 4ème année automatique «mesure et technologie des systèmes asservis » 2010-2011