Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Ecole Nationale Polytechnique
Département d'Automatique

Projet de Fin d'Etudes pour l'obtention du Diplôme

d'Ingénieur d'Etat en Automatique

Thème :

Commande par DSP d'UMAC de DELTA TAU d'un robot traceur de type SCARA

Elaboré par:

BEN CHABANE Sofiane

DOUAIDI Oualid

Proposé et dirigé par:

Mr. H. CHEKIREB

Mr. O. STIHI

ENP 10, Avenue Hassen BADI, 16200 EL-Harrach, Alger

Remerciements :

Le travail présenté dans ce mémoire a été mené au laboratoire de commande des processus du département d'Automatique de l'Ecole Nationale polytechnique.

Nous tenons à remercier vivement Monsieur H.CHEKIREB professeur à l'Ecole Nationale Polytechnique. On lui exprime toute notre reconnaissance et notre gratitude pour son soutien, la confiance qu'il nous a donné, ses conseils judicieux et son suivi régulier qui a permis l'accomplissement de ce travail.

Nous tenons également à remercier Monsieur O.STIHI qui nous a accompagnés tout au long de ce travail, et qui nous a facilité l'accès au laboratoire.

Nous remercions les membres du jury d'avoir accepté d'examiner notre travail.

Nos remerciements vont aussi à tous les enseignants de l'Ecole Nationale Polytechnique, et spécialement à ceux du Département d'Automatique pour leur apport de connaissances durant notre cursus.

Enfin, tous nos remerciements vont à toutes les personnes qui ont contribué de près ou de loin à l'aboutissement de ce travail.

Dédicace :

Je dédie ce modeste travail :

A mes chers parents, qui par leurs sacrifices, leur amour, leur patience et leurs encouragements m'ont permis de toujours persévérer, de viser haut et de donner le meilleur de moi-même

A mon frère Hachimi et ma Sœur Taous

A mes cousins : Chabane, Koceila, Cefax, Mehana, Lahlou, Omar et Ouahsen

A mes oncles, mes tantes et mes cousines et à toute ma famille

A tous mes amis

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

Et à toute la promotion de l'automatique 2011

Sofiane

Dédicace :

Je dédie ce travail:

A mes chers parents pour tous leurs sacrifices et sans qui jamais je n'aurais pu m'en sortir

A mes frères, sœurs, ma belle sœur, mes beaux frères, mes neveux et mes nièces

A mes cousins et mes cousines

A mes amis notamment ceux de l'ENPEI

A mon ami et partenaire Sofiane

Et à toute la promotion de l'ENP 2011

Oualid

TABLE DES MATIERES

CHAPITRE III : Régulateur PID et PID implémenté dans l'UMAC

CHAPITRE IV : Présentation du logiciel PEWIN32 PRO

CHAPITRE V : Application

LISTE DES FIGURES

CHAPITRE I :

CHAPITRE II :

CHAPITRE III :

CHAPITRE IV :

CHAPITRE V :

LISTE DES TABLEAUX

INTRODUCTION GENERALE :

Lors des dernières décennies, les industriels ont sans cesse recherché la meilleure rentabilité de leurs chaînes de production. C'est pourquoi l'automatisation, et en particulier la robotisation ont pris une place importante dans les unités de production. En effet, les robots sont utilisés afin de réaliser des tâches pénibles et dangereuses pour l'homme et ont largement contribué dans l'augmentation de la productivité (soudage, peinture, assemblage … etc.). De ce fait, la robotique constitue un marché porteur et très lucratif. Les premiers robots utilisés pour ces applications firent leur apparition dans les années 70. Ces robots, appelés SCARA (Selective Compliant Assembly Robot Arm), 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 sophistiquées. Ceci est rendu possible grâce à l'évolution de l'informatique et du traitement numérique 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 l'amélioration de la commande des moteurs grâce à leur vitesse de traitement de l'information d'un côté, et une meilleure interactivité avec l'utilisateur grâce à un ensemble de logiciels accompagnant cette interface, d'un autre côté.

Ce document est la synthèse de notre projet qui consiste à commander un robot de type SCARA par cette interface. Il s'articule autour de cinq chapitres. Le premier chapitre présente la structure existante au niveau de notre laboratoire, ainsi que sa modélisation et les actionneurs utilisés. Le second chapitre contient une description des fonctionnalités de l'UMAC afin de mettre en évidence sa puissance lors de l'exécution simultanée de nombreuses tâches de manière autonome et flexible. Plusieurs stratégies de commandes ont été développées pour la commande des bras manipulateurs, entre autre la commande décentralisée, qui est la commande implémentée dans l'UMAC, fait l'objet du troisième chapitre où on explique également l'algorithme de commande utilisé par l'interface. Quand au quatrième chapitre, le logiciel de configuration et de programmation de l'interface y est

présenté. Enfin, le dernier chapitre présente le fruit de notre travail. Dans ce chapitre, on expose les différentes tâches réalisées par notre robot ainsi que les résultats obtenus.

CHAPITRE I :

Présentation et modélisation du robot.

I.1. Introduction :

Un robot est un système mécanique composé de corps mobiles reliés par des actionneurs qui lui donnent des capacités de mouvement dans l'espace. Les structures mécaniques utilisées sont de types très divers, qu'il s'agisse de robots manipulateurs constitués d'un bras terminé par un outil ou un organe de préhension (pince, main multi-doigts), ou encore de robots mobiles avec des principes de locomotion adaptés à divers environnements.

Un robot est équipé de capteurs multiples pour la mesure de la position et de la vitesse de l'effecteur (organe terminal). Les mesures obtenues permettent de construire des représentations de l'état du système et de l'environnement bien adaptés à la tâche assignée, par la mise en œuvre de techniques de fusion de données multi-capteurs [1].

Dans ce chapitre, nous allons présenter la structure dont nous disposons afin de pouvoir lui apporter des améliorations par la suite. Puis nous allons présenter ses différents modèles qui seront implémentés plus tard dans l'UMAC. Une interface graphique permettant la visualisation du mouvement et les différents profils sera réalisée. Ensuite, nous présenterons les actionneurs, qui sont dans notre cas des moteurs à courant continu, chargés de générer le couple nécessaire pour effectuer le mouvement souhaité.

I.2. Description de la structure :

Nous disposons d'un robot, au niveau du laboratoire LCP, qui est de type SCARA. Ce robot est un manipulateur à trois degrés de liberté, ayant une structure RRP (figure 1.1).

Figure 1.1: *Structure existante*.

Cette structure présente plusieurs imperfections :

- Des jeux mécaniques au niveau des articulations.
- Fléchissement de segments dû à une grande charge (moteur 3 et la vis sans fin).
- Frottements dus à l'absence de roulements.

Notre but est d'apporter des améliorations à cette structure afin de pallier à ces problèmes, et permettre d'avoir une commande plus précise.

Une bonne modélisation de notre robot est requise pour pouvoir, par la suite, le commander avec précision et avoir de bons résultats

I.3. Modélisation du robot :.

On se contentera des deux articulations RR au niveau du coude et de l'épaule, car la troisième articulation (déplacement vertical) sera remplacée, par la suite, par un dispositif électromécanique commandé séparément (chapitre 5).

Donc, on aura à modéliser un robot planaire à deux degrés de liberté, qui a une structure *RR*. Nous allons présenter les différents modèles qui seront programmés par la suite.

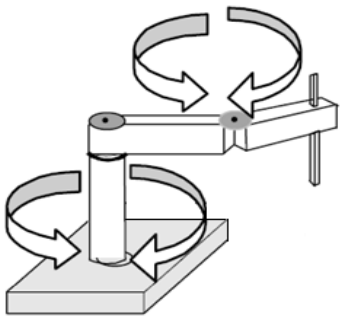


Figure 1.2 : *Schéma de la structure*.

I.3.1. Modèle géométrique direct :

Le modèle géométrique direct permet d'exprimer la position et l'orientation de l'effecteur terminal relativement à un repère fixe (par exemple celui de la base R_0), en fonction des variables articulaires du mécanisme, sans prendre en considération les forces produisant le mouvement [4].

La figure 1.3 présente un schéma de notre robot planaire :

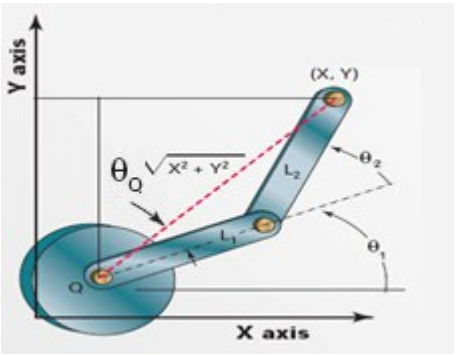


Figure 1.3 : *Robot SCARA dans l'espace articulaire*.

Les équations du modèle géométrique direct sont données par :

$$\begin{cases} X = L_1 \cos(\theta_1) + L_2 \cos(\theta_1 + \theta_2) \\ Y = L_1 \sin(\theta_1) + L_2 \sin(\theta_1 + \theta_2) \end{cases} \quad (1.1)$$

I.3.2. Modèle géométrique inverse :

Le modèle géométrique inverse permet d'exprimer les variables articulaires du mécanisme en fonction de la position et l'orientation de l'effecteur terminal dans l'espace cartésien [4].

En se limitant à des valeurs positives de l'angle du coude $\theta_2$qui donne une configuration droite, les équations géométrique inverses sont données par :

$$\begin{cases} \theta_2 = cos^{-1}\left(\frac{X^2+Y^2-{L_1}^2-{L_2}^2}{2L_1L_2}\right) \\ \theta_1 + \theta_0 = \arctan\left(\frac{Y}{X}\right) \\ \theta_0 = cos^{-1}\left(\frac{X^2+Y^2+{L_1}^2-{L_2}^2}{2L_1\sqrt{X^2+Y^2}}\right) \\ \theta_1 = (\theta_1 + \theta_0) - \theta_0 \end{cases} \qquad (1.2)$$

Le détail du calcul est donné dans l'annexe A.

I.3.3. Modèle cinématique inverse :

Le modèle cinématique inverse d'un robot manipulateur décrit les vitesses articulaires en fonction des vitesses des coordonnées cartésiennes. Nous prêtons plus d'importance au modèle cinématique inverse, car il sera implémenté par la suite lors de la génération de mouvements [4].

Les équations des vitesses articulaires sont données par :

$$\begin{pmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{pmatrix} = \begin{pmatrix} \frac{L_2\cos(\theta_1+\theta_2)}{L_1L_2\sin(\theta_2)} & \frac{L_2\sin(\theta_1+\theta_2)}{L_1L_2\sin(\theta_2)} \\ \frac{-X}{L_1L_2\sin(\theta_2)} & \frac{-Y}{L_1L_2\sin(\theta_2)} \end{pmatrix} \begin{pmatrix} \dot{X} \\ \dot{Y} \end{pmatrix} \qquad (1.3)$$

Le détail du calcul est donné dans l'annexe A.

I.4. Points singuliers et espace de travail :

Les points singuliers sont des configurations qui peuvent endommager le robot. Ils peuvent provoquer la déviation du robot de son parcours désiré, ainsi que l'usure des actionneurs (balais des moteurs à courant continu) en atteignant des vitesses trop élevées. Dans notre cas, et en analysant le modèle cinématique inverse, on voit bien que lorsque le terme $\sin(\theta_2)$est nul, les vitesses articulaires tendent vers l'infini.

Ce qui impose d'éviter les deux configurations suivantes :

$$\theta_2 = 0 \text{ et } \theta_2 = \pi \text{ (bras tendu, bras plié)}$$

L'espace de travail est l'environnement dont lequel notre robot peut évoluer. Il définit les points accessibles dans l'espace cartésien. Il est régi par les contraintes suivantes :

$$\begin{cases} |X| < L_1 + L_2 \\ |Y| < L_1 + L_2 \\ X^2 + Y^2 \leq L_1 + L_2 \\ 0 < \theta_2 < 140° \end{cases} \text{ Avec } L_1 = 186{,}64\ mm \text{ et } L_2 = 176{,}64\ mm.$$

Tout calcul fait, l'espace opérationnel obtenu est un disque creux de centre (0,0), de rayon interne de 124,60 mm et de rayon externe de 363,28 mm.

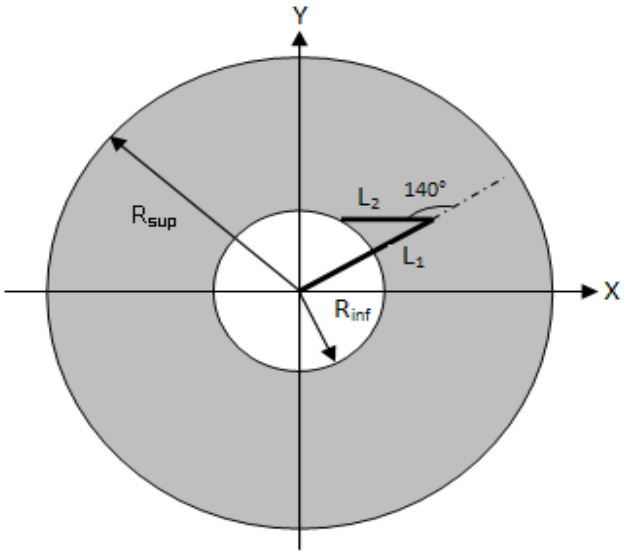


Figure 1.4 : *l'espace de travail du robot.*

I.5. Présentation des actionneurs utilisés [5]:

Nous avons utilisé des moteurs à courant continu à aimants permanents, munis d'encodeurs incrémentaux optiques avec une résolution de 500 *CPR* (comptes par tour) et d'un réducteur avec un rapport de réduction de 65,5. Leur référence est GM9236S027 du fabriquant américain PITTMAN.

Figure 1.5 : *Moteurs PITTMAN.*

Les caractéristiques du moteur PITTMAN sont mentionnées dans le tableau 1.1 :

paramètre	**Unité**	**Valeur**
Constante du couple	N.m	$48.8 * 10^{-3}$
Constante de la force contre électromotrice	V.s/rad	$48.8 * 10^{-3}$
Inertie du moteur	$kg.m^2$	$7.06 * 10^{-6}$
Constante de temps électrique	ms	1.06
Constante de temps mécanique	ms	8.5
Masse du moteur	g	391.2
Résistance	Ω	2.49
Inductance	mH	2.63
Tension d'alimentation	V	24
Rapport de réduction	-	65.5

Tableau 1.1 : *Caractéristique du moteur PITTMAN GM9236027.*

I.6. Génération de mouvements :

Les robots manipulateurs sont appelés à exécuter des tâches, qui consistent généralement en un déplacement d'un point initial vers un point final suivant une certaine trajectoire. Ceci nous amène à générer leurs mouvements.

La génération de mouvements est une étape très importante pour la commande des robots, elle consiste à calculer les consignes de référence nécessaires au régulateur afin d'effectuer le mouvement désiré.

La génération de mouvements peut se faire dans l'espace articulaire comme dans l'espace cartésien.

I.6.1. Génération de mouvements dans l'espace articulaire :

La génération de mouvements dans l'espace articulaire nécessite moins de calculs. Dans cette méthode le mouvement n'est pas affecté par le passage par les singularités du robot, et les contraintes de vitesses et de couples maximaux sont directement déduites des limites physiques des actionneurs. Par contre, la géométrie de la trajectoire de l'organe terminal est imprévisible, il y a donc risque de collision lorsque le robot évolue dans un environnement encombré [2].

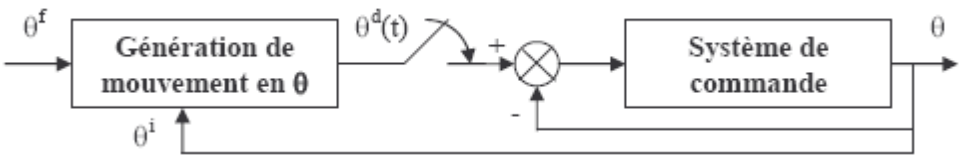


Figure 1.6 : *Génération de mouvements dans l'espace articulaire*.

I.6.2. Génération de mouvements dans l'espace cartésien :

La génération de trajectoires dans l'espace cartésien donne les profils de la position, de la vitesse et de l'accélération de l'organe terminal en fonction du temps. Dans cette méthode, les coordonnées articulaires sont converties en coordonnées cartésiennes et vis versa en utilisant le modèle géométrique direct et le modèle géométrique inverse du robot [2].

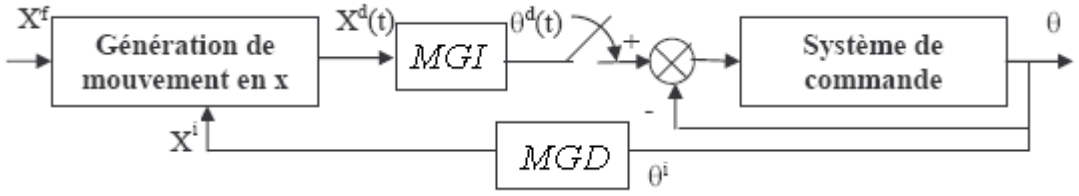


Figure 1.7 : *Génération de mouvements dans l'espace cartésien*.

La forme du parcours décrite par l'organe terminal peut être une ligne droite, circulaire, sinusoïdale…etc.

La génération de trajectoires dans l'espace opérationnel permet de contrôler la géométrie de la trajectoire de l'organe terminal. Ceci avec un temps de calcul plus important que la méthode précédente, car elle exige la transformation en coordonnées articulaires.

La planification de la trajectoire dans l'espace articulaire bien qu'elle soit optimale en temps de calcul et évite les singularités, la trajectoire de l'outil est imprévisible. En revanche, la planification dans l'espace opérationnel est très coûteuse en termes de calcul, elle assure la précision de la trajectoire de l'effecteur.

I.7. Réalisation d'une interface visuelle sous MATLAB :

Afin de pouvoir visualiser les différents profils de vitesse et de position ainsi que le comportement du robot, nous avons réalisé une interface graphique ergonomique sous MATLAB.

Cette interface est présentée à la figure 1.8 :

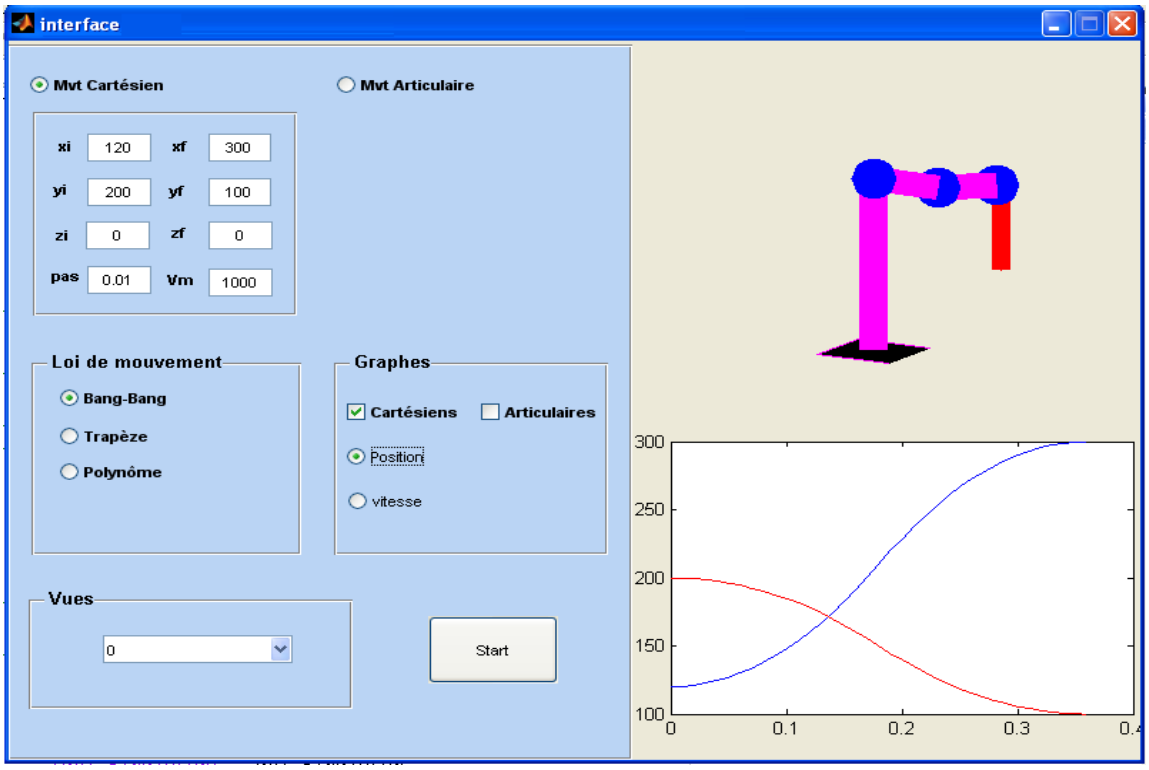


Figure 1.8 : *Interface graphique sous MATLAB.*

Comme le montre cette figure ci-dessus, si on veut, par exemple, générer une trajectoire dans l'espace cartésien, il suffit d'introduire les positions initiale et finale et de choisir la loi de mouvement désirée. En appuyant sur le bouton « Start », une animation montrant le

comportement du robot apparait sur l'écran. L'onglet « Vues » permet de choisir l'angle de vision (3 angles sont programmés). On peut aussi visualiser les différents profils de vitesse et de position à partir de l'onglet « Graphes ».

I.8. Conclusion :

Dans ce chapitre, nous avons présenté les différents modèles du robot SCARA à deux degrés de liberté. Nous avons aussi présenté les différentes méthodes de génération de mouvements, dans les deux espaces articulaire et cartésien. Ceci est nécessaire pour pouvoir les implémenter par la suite dans l'interface UMAC. Cette interface fait l'objet du prochain chapitre.

CHAPITRE II :

Description générale de l'interface UMAC

II.1. Introduction :

L'évolution de l'informatique et du traitement numérique du signal a eu un grand impact sur les techniques de la commande et de l'asservissement des moteurs électriques. Parmi ces techniques, on trouve la commande par DSP (*digital signal processor*), qui est très répandue grâce à l'architecture de ce dernier qui le rend très performant dans le domaine de l'automatique [6].

L'interface Turbo UMAC de la firme Américaine DELTA TAU, qui s'appuie sur cette technique, est conçue spécialement pour la robotique. Cette interface est programmable à partir d'un ordinateur via un port série. Elle peut contrôler huit moteurs simultanément avec une possibilité d'extension jusqu'à trente deux moteurs.

II.2. Description matérielle de l'UMAC [7] :

L'UMAC « Universal Motion and Automation Controller » est constitué d'un système modulaire en format 3U, combinant la puissance et la flexibilité de la famille des contrôleurs PMAC « Programmable Multi Axis Controller ». L'unité centrale de traitement du système UMAC est la carte Turbo PMAC2-3U CPU. Elle est reliée aux cartes d'entrée/sortie DSPGATEs et IOGATE qui sont reliées à leur tour aux amplificateurs par l'intermédiaire de l'UBUS « UMAC BUS ».

Figure 2.1 : *Le système Turbo UMAC.*

II.2.1. Les différentes cartes constituant l'UMAC :

II.2.1.1. Turbo PMAC2-3U CPU :

C'est l'unité centrale de traitement, elle peut commander jusqu'à huit axes, elle est constituée de :

Processeur DSP 56303 de MOTOROLA : dispose de toutes les fonctionnalités nécessaires pour développer des applications numériques complexes comme le contrôle des moteurs. Il a une architecture Harvard caractérisée par la séparation entre mémoire données et mémoire programmes. Cette structure offre la possibilité d'accéder aux données et aux instructions du programme en même temps (en un cycle d'horloge), grâce à deux bus distincts de données et deux bus distincts d'adresses, cela permet d'avoir de grandes vitesses d'exécution.

Mémoires **:** Pour profiter du parallélisme d'adressage du DSP 56303, le Turbo PMAC2-3U CPU dispose de trois types de mémoire :

- *Mémoire Flash* : c'est une mémoire non volatile contenant les micro-programmes, les variables d'installation, les programmes utilisateurs, les tables de conversion et les algorithmes de commande. La mémoire flash est située dans le champ mémoire **P**.
- *Mémoire de programmes actifs* : c'est une mémoire SRAM (Static RAM) située dans le champ mémoire **P.** Elle contient les résultats et les données intermédiaires lors de la compilation/assemblage des micro-programmes et les différents programmes élaborés par l'utilisateur.
- *Mémoires de données **X/Y*** : C'est un arrangement de trois circuits de mémoire SRAM utilisé pour la sauvegarde des données et des différents programmes. Ces mémoires sont situées dans les champs mémoire **X** et **Y**.

Whatchdog Timer (chien de garde) **:** s'occupe de la détection de toute anomalie (une sous tension ou une faible fréquence d'exécution) pouvant altérer le fonctionnement de la carte. En cas de dysfonctionnement, il se déclenche et arrête la carte.

Registres de contrôle **:** Ils sont répartis dans des zones mémoires séparées du CPU et des cartes DSPGATEs/IOGATE (leur représentation dans un seul bloc sur la figure 2.2 n'est que symbolique). Ces registres définissent l'état des entrées/sorties ainsi que le fonctionnement global du système.

Un connecteur série **RS-232/422 DB25** assure la communication séquentielle entre l'ordinateur principal et le Turbo PMAC2-3U CPU.

Un schéma regroupant les différentes parties du Turbo PMAC2-3U CPU est illustré sur la figure 2.2 :

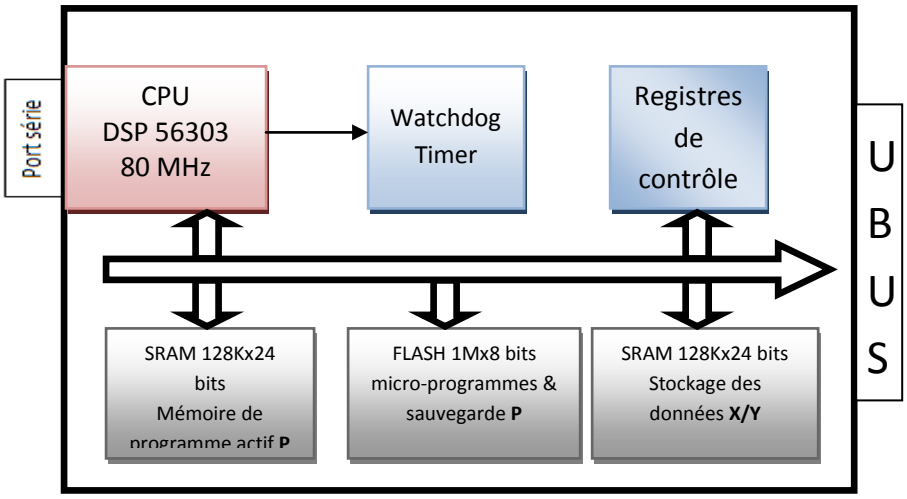


Figure 2.2 : *Schéma bloc du Turbo PMAC2-3U CPU*

II.2.1.2. DSPGATE :

Elle assure la communication entre l'unité centrale du traitement (Turbo PMAC2-3U CPU) et l'ensemble des amplificateurs et des capteurs. Elle contient 4 canaux (pour commander 4 axes), elle se charge de :

- La conversion Analogique/Numérique ;
- Recevoir les signaux des capteurs des moteurs ;
- Noter les fins de course des moteurs.

Elle permet trois types de commande, à savoir :

- Commande en modulation de largeurs d'impulsions (PWM) pour les moteurs triphasés ;
- Commande analogique de couple (DAC) pour les moteurs à courant continu ;
- Commande d'impulsion et de directions pour les moteurs pas-à-pas.

Nous disposons de deux DSPGATEs ACC-24E 4A. De ce fait, on peut commander jusqu'à huit axes.

Figure 2.3 : *DSPGATE*.

II.2.1.3. IOGATE :

Nous disposons d'une carte IOGATE (input/output gate) ACC12E. Elle permet l'adaptation des signaux d'entrées/sorties digitaux avec isolation optique. Elle contient 24 lignes d'entrées et 24 lignes de sorties à tension élevée. La lecture et l'écriture des données entrées/sorties se font par des pointeurs de zone mémoire.

Figure 2.4 : *IOGATE*.

II.2.1.4. Amplificateurs AMP-2 :

Nous disposons de huit amplificateurs arrangés sur deux cartes AMP-2 connectés à la DSPGATE, ayant pour rôle de rendre les signaux de commande délivrés par celle-ci exploitables par les moteurs. Dans notre cas, pour la régulation du couple des moteurs à courant continu, ces amplificateurs sont dotés d'une boucle de régulation analogique du courant. La tension maximale d'alimentation de ces cartes (AMP-2) est de 40 VDC pour un courant maximum de 3 A correspondant à une puissance maximale de 120 W.

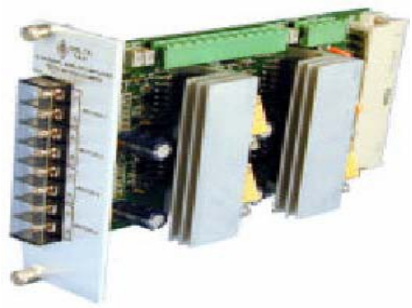


Figure 2.5 : *Amplificateur AMP-2*.

II.2.1.5. UBUS (UMAC BUS) :

C'est une interface conçue pour connecter les différents modules de l'UMAC, considérée comme une surface arrière commune. Cette interface permet au processeur de contrôler simultanément toutes les cartes qui lui sont connectées.

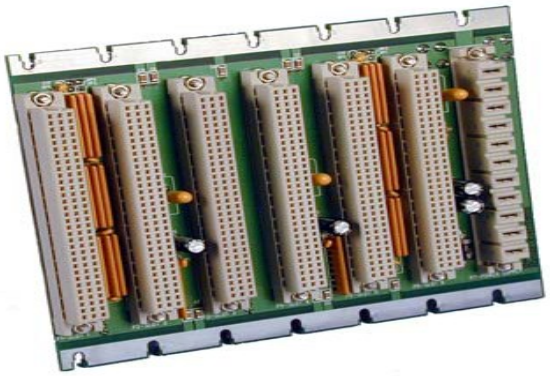


Figure 2.6 : *L'UBUS (La surface arrière)*.

II.2.1.6. Alimentation électrique :

C'est une carte qui fournit une alimentation de ±15 V avec un courant maximal de 1.5A pour les circuits analogiques, et une alimentation de 5 V avec un courant maximal de 1.4A pour les circuits numériques.

Figure 2.7 : *UMAC Alimentation*.

II.2.2. Fonctionnement général de l'UMAC :

Le fonctionnement de l'UMAC se fait suivant ces différentes étapes :

- Les différents programmes sont élaborés sur un ordinateur grâce à un logiciel puis chargés dans une zone mémoire de la carte TURBO PMAC2-3U CPU via le port série RS 232/422.
- A l'exécution des programmes, le CPU et la DSPGATE s'échangent les signaux de commande et de mesure à travers l'UBUS.
- La DSPGATE envoie les signaux de commande (après CNA) aux amplificateurs et compte les impulsions venant des capteurs.
- L'amplificateur génère un courant proportionnel au signal de commande délivré par la DSPGATE.
- Si des signaux d'entrées/sorties digitaux sont utilisés, l'IOGATE les conditionne et les sauvegarde dans un registre pour que le CPU puisse les utiliser.

Le schéma synoptique 2.8 illustre le fonctionnement général de l'interface UMAC :

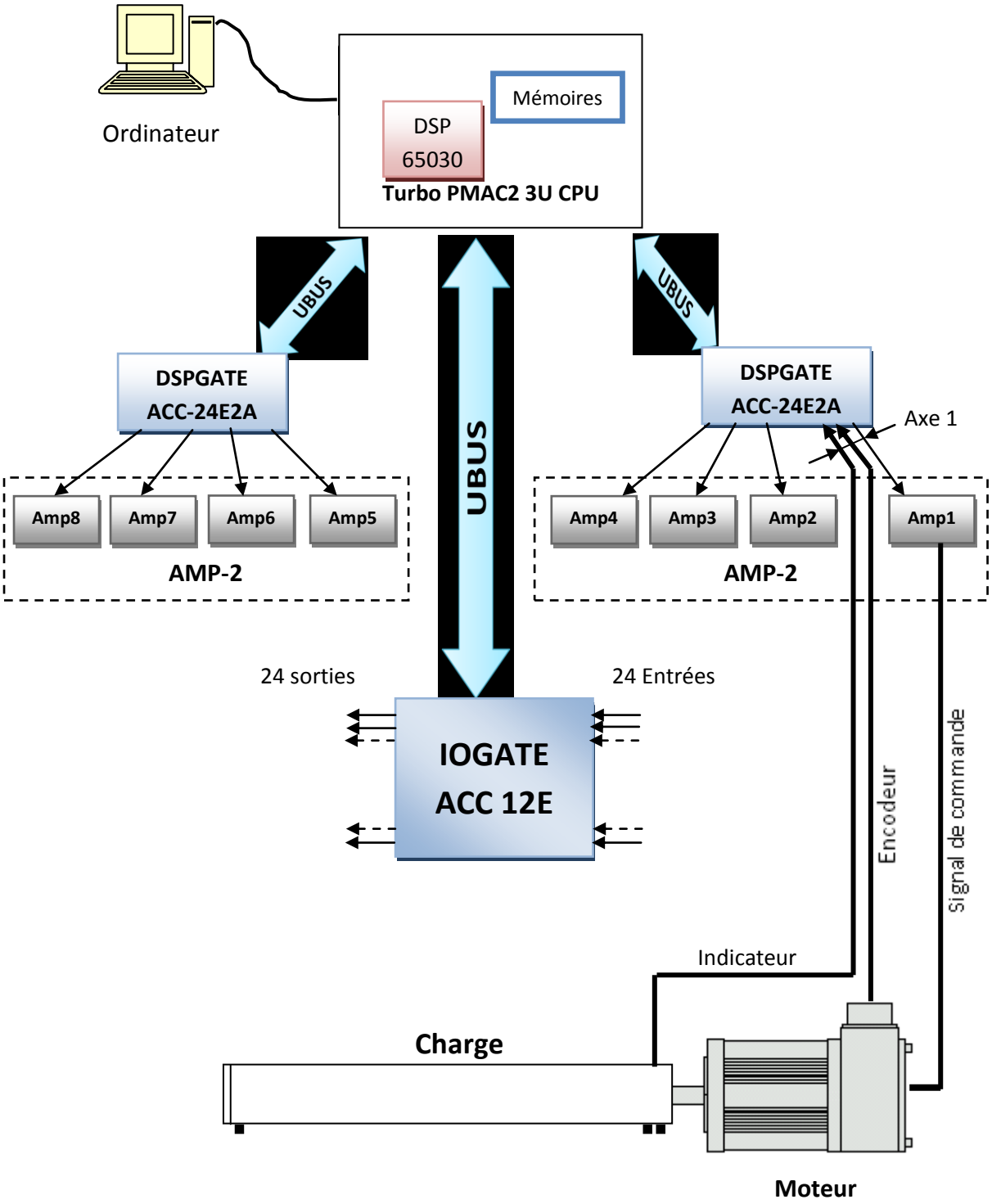


Figure 2.8 : *Schéma synoptique du fonctionnement général de l'UMAC.*

Il est à signaler que dans notre application, l'IOGATE n'est pas utilisée. Le câblage de ses entrées/sorties n'est pas représenté.

II.3. Câblage de l'UMAC :

II.3.1. branchement de l'encodeur du moteur à la DSPGATE [11] :

Nous disposons de moteurs dotés d'encodeurs de haute résolution. Chaque encodeur a cinq lignes. Trois lignes, comme sorties, pour envoyer à la DSPGATE le code de la position, et deux lignes, comme entrée, pour assurer son alimentation.

Le branchement de l'encodeur à la DSPGATE est illustré sur la figure 2.5 :

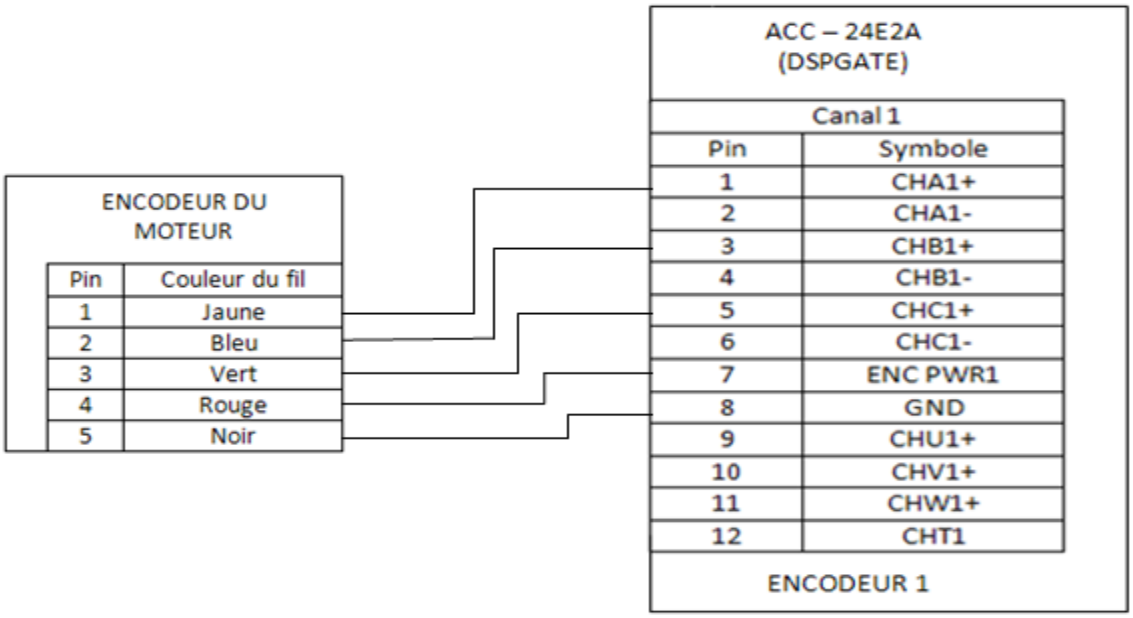


Figure 2.9 : *Branchement de l'encodeur à la DSPGATE.*

Quand le moteur effectue un mouvement, l'encodeur envoie sur les entrées CHA, CHB et CHC des impulsions à la DSPGATE pour la mesure de la position comme le montre la figure 2.6 :

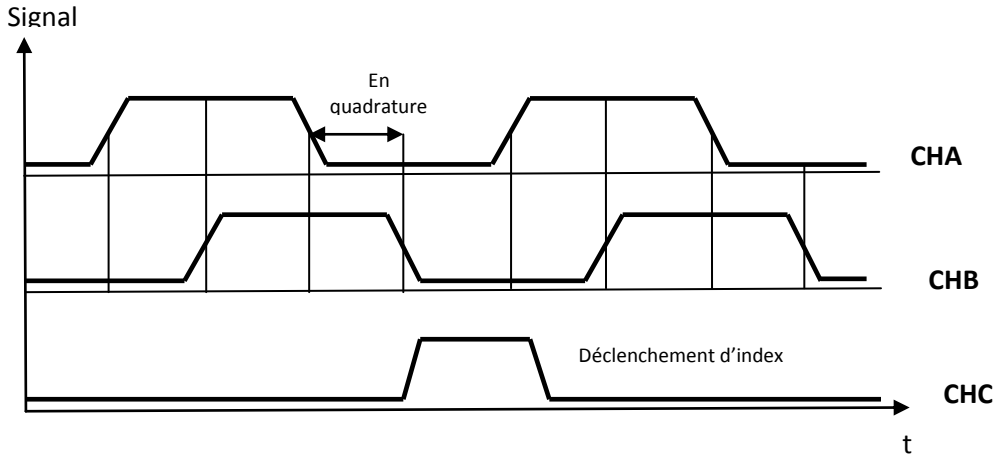


Figure 2.10 : *Signaux délivrés par l'encodeur du moteur lors d'un mouvement.*

- **CHA :** Quand le moteur tourne, le disque de l'encodeur tourne aussi et la DSPGATE reçoit sur CHA des impulsions. Ces impulsions vont être acheminées vers un compteur et deux timers pour déterminer la position du disque et le temps correspondant.
- **CHB :** De même, la DSPGATE reçoit des impulsions sur CHB. Ces impulsions sont en quadrature par rapport à celles reçues sur CHA, elles sont importantes pour déterminer le sens de la rotation. En effet, si ces impulsions sont en retard par rapport aux impulsions reçues sur CHA, alors le disque tourne dans le sens positif. Dans le cas contraire, le disque tourne dans le sens négatif. La détermination du sens de rotation est faite matériellement par le décodeur installé dans la DSPGATE, qui est formé de deux bascules D montées en parallèle :

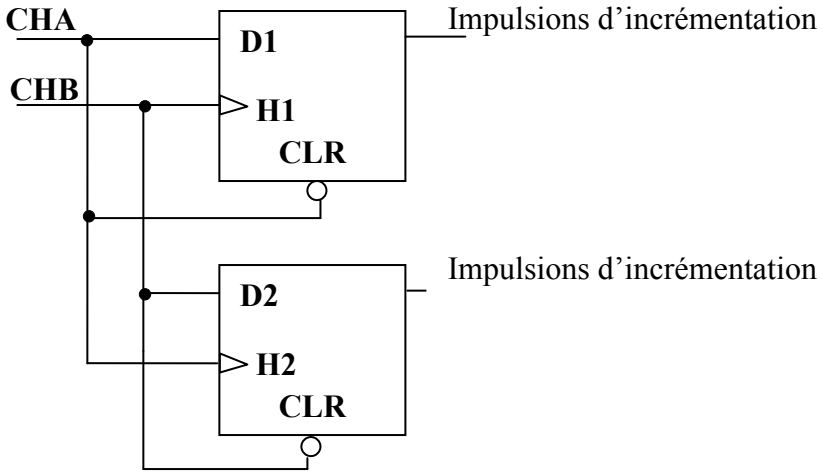


Figure 2.11 : *Décodeur du sens de rotation implémenté dans la DSPGATE*.

Si le sens de rotation est positif, on reçoit des impulsions sur la sortie de la première bascule. Dans le cas contraire, les impulsions sont sur la sortie de la deuxième bascule.

- **CHC :** Cette ligne représente l'index de l'encodeur qui reçoit une impulsion à chaque tour du disque de l'encodeur. Elle sert à déterminer le nombre de tours effectués.

II.3.2. Câblage nécessaire pour la mise en marche de l'UMAC [11] :

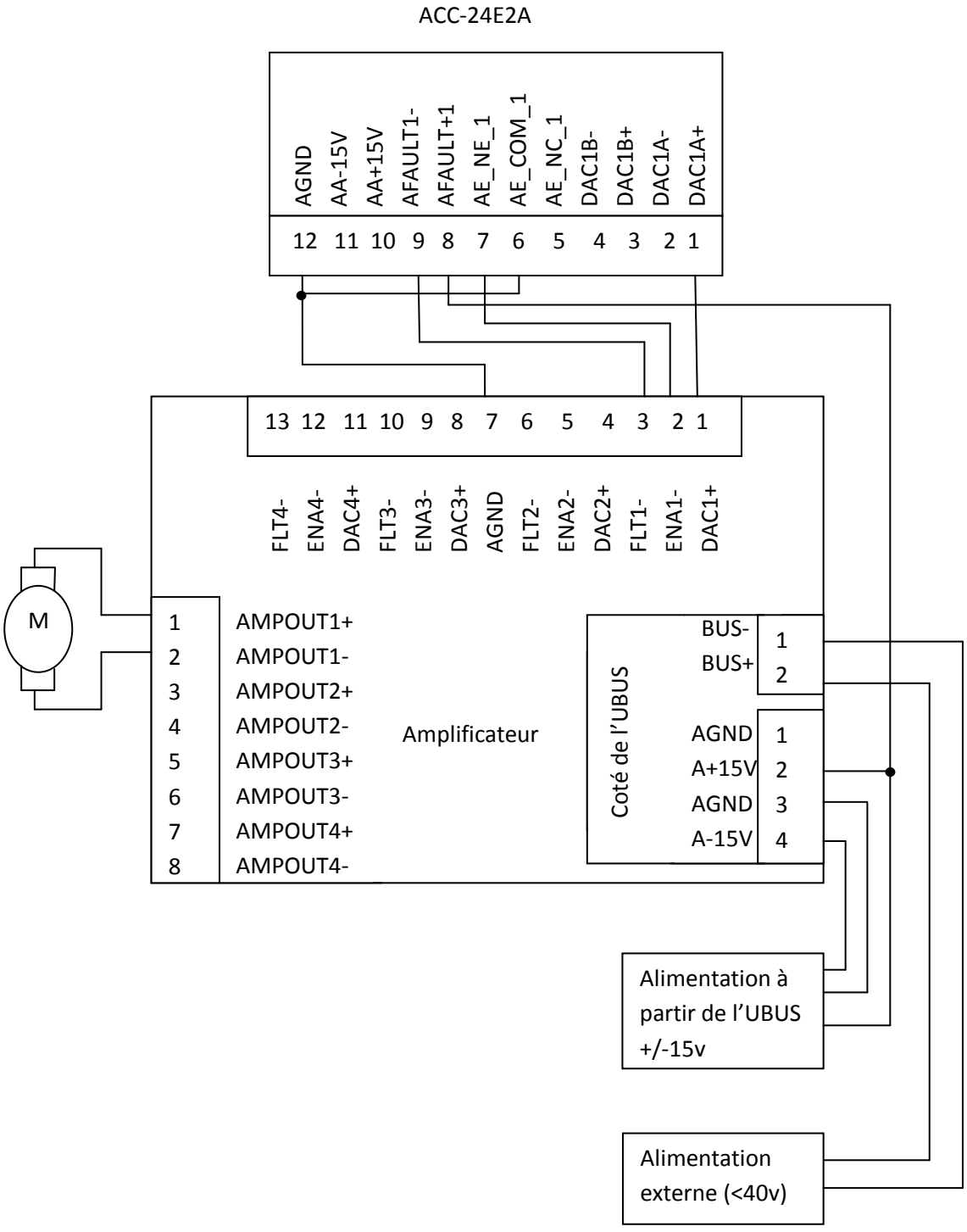


Figure 2.12 : *Câblage de la DSPGATE et l'Amplificateur.*

Les différents signaux mis en jeu :

- **FAULT (AFAULT_n^{+}, AFAULT_n^{-})** *Amplifier Fault*: ces lignes sont utilisées par l'amplificateur pour envoyer un signal d'erreur au CPU à travers la DSPGATE en cas de dysfonctionnement (défaut d'amplification).
- **AENA (AE_NO_n, AE_NC_n, AE_COM_n)** *Amplifier Enable*: commandées par le CPU, ces sorties sont utilisées pour activer ou désactiver les axes de l'amplificateur. Ces opérations sont importantes pour des raisons de sûreté.
- **Sorties DACn** : ces lignes permettent d'envoyer les commandes au moteur.
- **AA-, AA+** : se sont des alimentations analogiques qui peuvent être configurées comme entrées ou comme sorties.
- **AGND** : C'est la référence analogique.
- **AMPOUTn+, AMPOUTn-** : sorties d'alimentations pour le moteur n.

II.4. Programmation de l'UMAC [8] :

La disposition matérielle de l'interface UMAC est idéale pour les applications de commande de mouvement. Sa conception autonome et flexible lui permet d'exécuter simultanément de nombreuses tâches. Pour l'exploiter convenablement, une bonne connaissance logicielle est requise.

II.4.1. L'exécutable principal PEWIN32 Pro :

C'est un logiciel comprenant plusieurs applications dans un environnement permettant la configuration et la programmation de l'interface. Il est développé par la firme DELTA TAU. Il contient un éditeur de programmes et une suite d'outils pour configurer et mettre en œuvre l'UMAC :

- *Turbo Setup Pro* : permet la configuration des différents paramètres d'une manière interactive, tout en visualisant les différentes valeurs de ces derniers.
- *PMAC Tuning Pro* : utilisé pour la configuration des paramètres des boucles d'asservissement. Il permet d'optimiser les gains numériques du régulateur PID implémenté dans l'interface tout en visualisant les réponses des moteurs.
- *PMAC Plot Pro* : permet l'acquisition et la représentation graphique des données.

L'installation et la configuration de ce logiciel est détaillé dans le chapitre IV.

Pour permettre à l'utilisateur de travailler dans un environnement adéquat, le logiciel PEWIN32 Pro utilise des variables facilitant la programmation et la configuration des applications sans avoir à s'encombrer avec la spécification des adresses. Ces variables se divisent en quatre classes :

- *Variables **I** (variables d'installation et d'initialisation)*: permettent d'accéder aux registres de contrôle indirectement et de configurer l'interface pour les applications désirées. Elles sont stockées dans la mémoire Flash (le champ P). Elles ont des significations prédéfinies. Ces variables sont classées par catégories pour un total de 8191 variables. Un tableau contenant les variables I usuelles est présenté dans l'Annexe B.
- *Variables **P** (variables globales)*: variables réelles stockées dans des cases mémoire (SRAM), elles sont utilisées dans le traitement des données.
- *Variables **Q** (variables locales)* : utilisées comme arguments pour les fonctions.
- *Variables **M** (Pointeurs)* : permettent un accès facile la mémoire interne de l'interface et aux registres d'entrées/sorties.

II.4.2. Les différents programmes de l'UMAC :

II.4.2.1. Les programmes de mouvements :

Dans ces programmes, les mouvements à effectuer sont décrits par une suite d'instructions. A l'exécution d'un programme de mouvement, ces instructions sont traduites en une série de positions de références pour les moteurs. L'interface peut supporter jusqu'à 256 programmes de mouvement.

Un programme de mouvement peut faire appel à n'importe quel autre programme de mouvement comme sous programme, avec ou sans arguments. La programmation des mouvements peut être considérée comme un couplage entre la programmation évoluée et le langage machine.

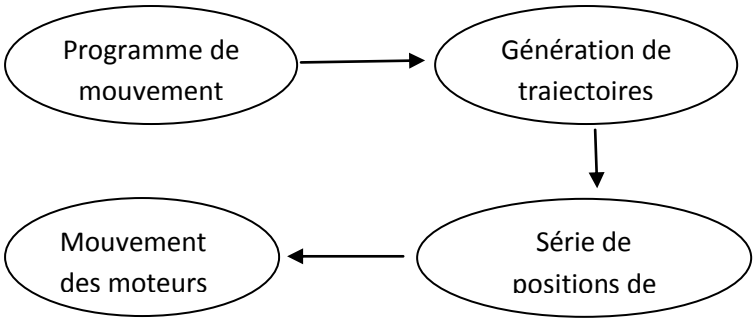


Figure 2.13 : *interprétation de programmes de mouvement*.

II.4.2.1.1. Elaboration d'un programme de mouvement :

Les commandes usuelles pour l'élaboration d'un programme de mouvement sont :

- *OPEN PROG {constante}* : indique qu'il s'agit d'un programme de mouvement. La constante représente le numéro du programme.
- *CLEAR* : Efface le contenu du programme ouvert.
- *LINEAR*, *RAPID*, *CIRCLE*, *PVT* et *SPLINE* : définissent les trajectoires du mouvement (elles seront détaillées dans le prochain paragraphe).
- *INC* et *ABS* : indique le mode du mouvement. Dans la mode incrémental (*INC*), le mouvement est défini par la position initiale et le déplacement relatif à celle-ci, tandis que le mouvement dans le mode absolu (*ABS*) est défini par les deux positions initiale et finale.
- *Commande de mouvement* : un mouvement est défini par la spécification d'un axe suivi d'un déplacement. Les mouvements spécifiés sur la même ligne sont exécutés simultanément. S'ils le sont sur des lignes consécutives, ils s'exécutent l'un après l'autre.
- *Instructions usuelles* : dans un programme de mouvement, on peut utiliser les instructions d'itération condition telles que WHILE … ENDWHILE, IF … ELSE … ENDIF, ainsi que les commandes de branchement telles que GOTO et GOSUB. On peut aussi faire appel à un sous programme avec la commande CALL.
- *CLOSE* : ferme le buffer ouvert.

Dans ce qui suit, nous allons présenter les mouvements principaux que l'UMAC peut effectuer, ainsi que la façon avec laquelle ils sont générés.

II.4.2.1.2. Mouvement linéaire :

C'est un mouvement ayant un profil de vitesse linéaire. Les figures suivantes illustrent les différents mouvements linéaires qu'on peut programmer avec l'UMAC :

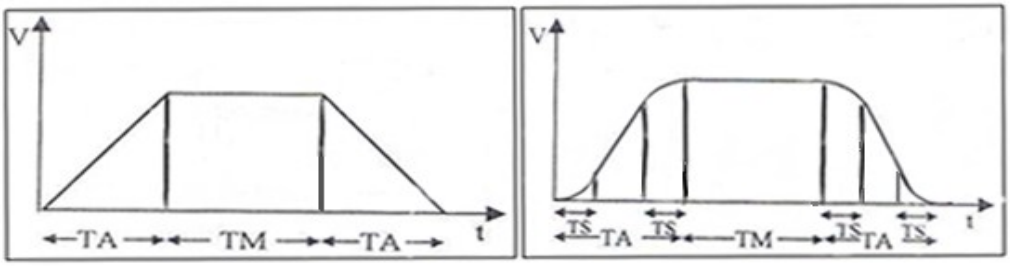


Figure 2.14 : *Mouvement linéaire sans S-curve (à gauche), et avec S-curve (à droite).*

Avec TA est la durée de l'accélération, TM est la durée du mouvement et TS est la durée de la courbure (mode S-curve).

Exemple :

```
CLOSE                ; fermer tous les buffers ouverts
DELETE GATHER        ; effacer toutes les données intermédiaires
UNDEFINE ALL         ; effacer toutes les définitions liées aux SC
#1->363.889X         ; lier le moteur 1 à l'axe X avec une graduation d'axe
                     en  degré
OPEN PROG 7 CLEAR
LINEAR                ; profil de vitesse linéaire
INC                   ; mode incrémental
TA 250                ; temps d'accélération de 250 ms
TS 50                 ; temps de courbure de 50 ms
TM 1000               ; temps de mouvement de 1000 ms
X 180                 ; effectuer un mouvement de 180°
CLOSE                 ; fermer le programme 7
```

II.4.2.1.3. Mouvement SPLINE :

Il permet le raccordement d'une série de mouvements avec une interpolation cubique (un polynôme de degré 3), pour qu'il n'y ait pas de changements brusques de vitesse et d'accélération.

Le mouvement est subdivisé en segments de durées TA égaux avec la commande SPLINE1, et en segments de durées TA variables avec la commande SPLINE2.

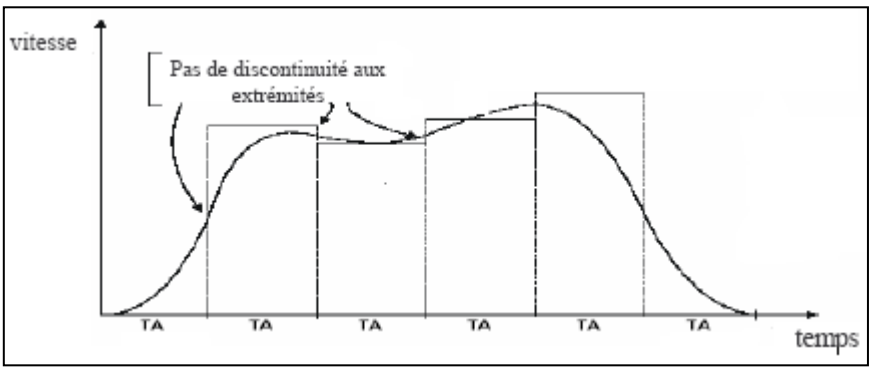


Figure 2.15 : *Le mouvement SPLINE.*

Exemple :

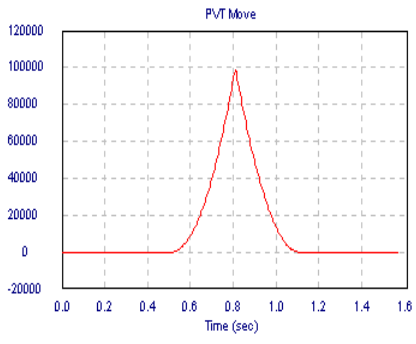
```
CLOSE
DELETE GATHER
UNDEFINE ALL
#1->363.889X
OPEN PROG 8 CLEAR
LINEAR
INC
SPLINE1     ; raccorder les mouvements avec une interpolation cubique
TA 150
TS 30
TM 1000
X 145
X -120
CLOSE
```

II.4.2.1.4. Le mode PVT (Position Velocity Time) :

On l'utilise pour un contrôle plus direct de la trajectoire. Dans ce mode, l'utilisateur indique la position et la vitesse finales, ainsi que le temps de mouvement. Le processeur génère automatiquement le profil de vitesse optimum. Cela nécessite plus de calcul, mais permet une commande plus précise de la trajectoire.

Exemple :

```
CLOSE
&1 #1->2000X
OPEN PROG 1 CLEAR
INC
PVT300      ; temps de mouvement de 300 ms
X5:50       ; déplacement de 5 unités avec une
            vitesse finale de 50 unités/I5190*ms
X5:0        ; déplacement de 5 unités avec une
            vitesse finale nulle
CLOSE
```



II.4.2.1.5. Le mouvement circulaire :

Utilisé pour effectuer un mouvement entre deux points en suivant une trajectoire circulaire (arc de cercle). Le plan de l'arc circulaire est spécifié par la commande NORMAL (NORMAL K-1 : plan XY, NORMAL J-1 : plan ZX et NORMAL I-1 : plan YZ). Le centre est spécifié par le vecteur (I, J, K) = (la composante parallèle X, Y, Z), la position finale est indiquée par les valeurs des axes dans le programme sous un mode absolu ou incrémental.

La différence entre CIRCLE1 et CIRCLE2 est le sens de rotation : CIRCLE1 sens horaire et CIRCLE2 sens trigonométrique.

La syntaxe du mode circulaire est donnée par :

CIRCLE {1 ou 2}

TM (temps de mouvement) ou F (la vitesse)

X {data} Y {data} R {data} ; donner la position finale (mode ABS ou INC) et le rayon du raccordement

Ou bien

X {data} Y {data} I {data} J {data} ; donner la position finale (mode ABS ou INC) et le centre du cercle (mode ABS ou INC)

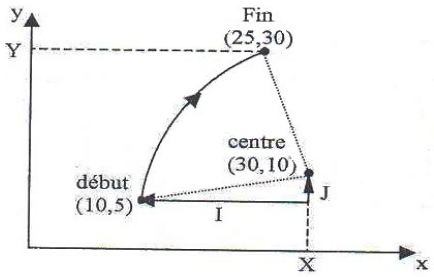
Exemple :

```
NORMAL K-1
ABS (XY)
INC (R)
CIRCLE1
F10
X25 Y30 I20 J5
```

ou bien

```
NORMAL K-1
ABS (XY)

CIRCLE1
F10
X25 Y30 R20.615
```



II.4.2.2. Les programmes PLC et PLCC (PLC Compilé) :

L'UMAC peut supporter 64 programmes PLC (32 PLC et 32 PLCC) qui fonctionnent indépendamment des mouvements avec une exécution répétitive et très rapide. Ils ont les mêmes constructions logiques que les programmes de mouvements, mais n'exécutent aucun déplacement.

II.4.2.2.1. Les PLC (Programmable Logic Controllers):

Généralement utilisés pour la surveillance des entrées, l'envoi des messages, la surveillance des paramètres de mouvement, le changement des gains et pour démarrer et arrêter les programmes de mouvement. Les PLC constituent un outil très puissant de gestion des programmes de mouvement grâce à leur accès aux variables de travail de l'interface et aux entrées/sorties.

La structure d'un PLC est la suivante :

```
CLOSE
DELETE GATHER                    ; effacer les données contenues
                                   dans la mémoire
DELETE TRACE                     ; effacer les programmes
                                      précédents
OPEN PLC n CLEAR
                      {Les fonctions PLC}
CLOSE
ENABLE PLC n
```

Puisque tous les programmes PLC sont activés (ENABLE) lors d'un *RESET*, il est recommander de mettre la variable I5 = 0 pour les désactiver.

II.4.2.2.2. Les PLCC (PLC Compilés) :

Les PLCC s'exécutent plus rapidement que les PLC grâce à l'élimination du temps d'interprétation et la possibilité d'exécuter des opérations arithmétiques et logiques sur des nombres entiers de 24-bits (variables L). Les opérations en virgule flottante dans un PLCC s'exécutent 2 à 3 fois plus rapidement que dans les PLC ; les opérations sur des nombres entiers s'exécutent 20 à 30 fois plus rapidement dans les PLCC que dans les PLC.

La compilation se fait dans l'ordinateur, puis le code machine résultant est chargé dans l'UMAC. Tous les codes des PLCC doivent appartenir au même fichier texte ASCII, car l'interface n'en supporte qu'un seul fichier.

Les variables L sont utilisées dans les PLCC pour pouvoir exécuter des opérations sur des nombres entiers au lieu des opérations en virgule flottante. L'utilisateur définit ces variables de la même manière que les variables M, mais elles ne peuvent être combinées avec d'autres types de variables (toutes les variables dans la commande de part et d'autre, doivent être des variables L ou des constantes entières).

La structure d'un PLCC est la suivante :

```
CLOSE
DELETE GATHER       ; effacer les données contenues dans la
                     mémoire
DELETE TRACE        ; effacer les programmes précédents
OPEN PLCC n CLEAR
        {Les fonctions PLCC}
CLOSE
ENABLE PLCC n
```

II.4.2.3. Programmes géométriques :

Les différents modèles du robot sont implémentés dans l'UMAC grâce à des programmes spéciaux *forward-kinematic* (programme géométrique direct) et *inverse-kinematic* (programme géométrique inverse). Ils sont considérés comme des sous-programmes des programmes de mouvements, ils permettent le passage de coordonnées cartésiennes aux coordonnées articulaires et vis versa. Ils sont exécutés automatiquement aux temps appropriés une fois qu'ils sont activés (en affectant la valeur 1 à la variable Isx50).

On trouve également le programme géométrique inverse pour le mode PVT qui permet l'utilisation des calculs cinématiques (modèle cinématique).

II.4.2.3.1. Elaboration des programmes géométriques directs :

Ils ont la structure suivante :

```
&n OPEN FORWARD

CLEAR

      {Les instructions et les fonctions du
modèle}

CLOSE
```

Avant l'exécution d'un programme géométrique direct, UMAC place les positions actuelles de chaque moteur xx dans les variables globales Pxx. Ces variables seront utilisées comme entrées pour le calcul.

Les coordonnées cartésiennes obtenues après calcul pour chaque axe seront placées automatiquement dans les variables Q1-Q9, comme suit :

Variable Q	Axe correspondant	Variable Q	Axe correspondant	Variable Q	Axe correspondant
Q1	A	Q4	U	Q7	X
Q2	B	Q5	V	Q8	Y
Q3	C	Q6	W	Q9	Z

Tableau 2.1 : *Correspondance des variables Q avec les axes.*

II.4.2.3.2. Elaboration des programmes géométriques inverses :

Ils ont la structure suivante :

```
&n OPEN INVERSE

CLEAR

      {Les instructions et les fonctions du
modèle}

CLOSE
```

Avant l'exécution d'un programme géométrique inverse, UMAC place les positions actuelles de chaque axe dans les variables Q1-Q9. Ces variables seront utilisées comme entrées pour le calcul.

Les coordonnées articulaires obtenues après calcul pour chaque moteur xx seront placées automatiquement dans les variables Pxx.

II.4.2.3.3. Programmes géométriques inverses pour le mode PVT :

L'UMAC peut également faire la conversion des vitesses de l'espace cartésien à l'espace articulaire dans le programme géométrique inverse, pour permettre l'utilisation du mode PVT qui exige des calculs cinématiques.

Avant l'exécution du mode PVT, l'UMAC placera automatiquement les valeurs des vitesses commandées des axes dans les variables Q11-Q19. Le tableau suivant montre les variables utilisées pour chaque axe :

Variable Q (vitesse de l'axe)	Axe correspondant	Variable Q (vitesse de l'axe)	Axe correspondant	Variable Q (vitesse de l'axe)	Axe correspondant
Q11	A	Q14	U	Q17	X
Q12	B	Q15	V	Q18	Y
Q13	C	Q16	W	Q19	Z

Tableau 2.2 : *Correspondance des variables Q avec les vitesses des axes.*

Les vitesses articulaires obtenues après calcul pour chaque moteur xx seront placées automatiquement dans les variables P1xx.

Pour activer les calculs cinématiques, il faut mettre la variable Q10 à 1.

II.4.3. Les systèmes en coordination (SC):

L'exécution d'un programme de mouvement nécessite la définition d'un système en coordination. Ce dernier regroupe plusieurs moteurs afin qu'ils aient des mouvements synchronisés.

Les systèmes en coordination peuvent exécuter plusieurs programmes de mouvements indépendamment ou simultanément, mais aussi d'exécuter le même programme à des moments différents. Ils sont définis en affectant un moteur à chaque axe avec un rapport de définition d'axe (graduation d'axe). Il doit y avoir au moins un moteur attaché à un axe pour pouvoir exécuter des programmes de mouvements.

Il est nécessaire de spécifier le système en coordination à l'exécution d'un programme de mouvement et dans les programmes géométriques. Et ce grâce à la commande « &n », où « n » désigne le numéro du SC.

II.4.4. Graduation d'axe :

Elle définit l'unité de déplacement des moteurs qui est équivalente à un certain nombre d'incréments de l'encodeur. Dans notre cas, nous disposons d'un encodeur de 500 comptes par tour avec un rapport de réduction $\eta = 65,5$. Comme la résolution est augmentée 4 fois par l'interface, ce qui signifie qu'on a 2000 comptes par tour, alors une rotation complète de l'arbre correspond à : $\eta*2000 = 131000$ comptes. Ce qui fait que l'affectation « #1->131000X » permet de travailler avec une unité de déplacement du moteur 1 de 1 tour. Si on

veut travailler avec le degré, il suffit de diviser 131000 par 360°, ce qui correspond à 363,889 comptes.

II.5. Conclusion :

La description matérielle et logicielle de l'UMAC nous a permis de mettre en évidence la puissance qu'il peut atteindre lors de l'exécution simultanée de nombreuses tâches de manière autonome et flexible. La programmation de l'interface se fait grâce au logiciel PEWIN32PRO.

Nous avons vu qu'à l'exécution d'un programme de mouvement, l'UMAC génère une série de positions de références que le moteur doit suivre de manière précise afin d'obtenir le mouvement désiré. Pour cela, une commande robuste des moteurs est nécessaire.

Dans le prochain chapitre, nous allons présenter la commande décentralisée qui est la commande utilisée par l'UMAC, en s'appuyant sur un régulateur de type PID + Feed Forward.

CHAPITRE III :
Régulateur PID et PID implémenté dans l'UMAC

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 ce genre de systèmes (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, pneumatiques ou hydrauliques) [5].

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 de type *SISO* où l'effet des couplages est considéré comme des perturbations. La commande décentralisée est également la commande utilisée par l'interface UMAC. Son asservissement est assuré par le régulateur PID implémenté dans cette interface. Ce dernier est associé à une boucle de compensation feedforward et un filtre éjecteur afin de rejeter les fréquences de résonances.

Dans ce chapitre, nous présentons, dans un premier lieu, la théorie du PID appliqué aux moteurs électriques (moteurs à courant continu pour notre cas). Puis, nous présentons l'algorithme de commande PID implémenté dans l'interface UMAC.

III.2. La théorie du PID appliqué aux moteurs électriques [9] :

III.2.1. Modélisation mathématique d'un moteur à courant continu :

Un moteur électrique à courant continu est régit par les équations physiques découlant de ses caractéristiques électriques, mécaniques et magnétiques. En utilisant le théorème du moment cinétique et des équations d'électromagnétique, on obtient un système d'équations différentielles linéaires :

$$\begin{cases} u(t) = e(t) + R.i(t) + L.\frac{di(t)}{dt} \\ e(t) = k_e.\omega(t) \\ C_m(t) = k_c.i(t) \\ C_m(t) - C_r(t) = J_T.\frac{d\omega(t)}{dt} \end{cases} \quad (3.1)$$

Avec :

u : Tension appliquée au moteur.

e : Force électromotrice.

i : Intensité traversant le moteur.

ω : Vitesse de rotation du moteur.

C_m : Couple moteur généré.

C_r : Couple résistant.

On en déduit le modèle suivant dans le domaine de Laplace :

$$\Omega(P) = \frac{k_c}{k_e k_c + RJ_T P + LJ_T P^2} U(P) - \frac{R + LP}{k_e k_c + RJ_T P + LJ_T P^2} C_r(P) \quad (3.2)$$

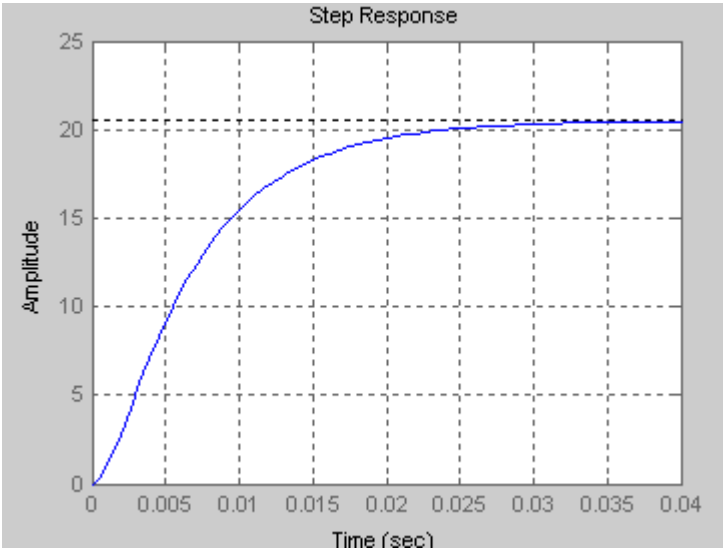


Figure 3.1 : *Réponse à un échelon de tension.*

On observe bien que le système modélisé est du second ordre, lorsque l'inductance interne est négligeable devant la résistance interne, il s'apparente à un système du premier ordre.

III.2.2. Boucle de retour :

Étant donné que l'asservissement d'un moteur utilise une boucle fermée, il est nécessaire de disposer d'un capteur de position angulaire ou de vitesse angulaire.

Les moteurs utilisés dans notre application (Pittman GM 9236S027) sont munis d'encodeurs incrémentaux optiques assurant la mesure de la position. L'UMAC détermine la vitesse par dérivation numérique de la position.

III.2.3. Asservissement et influence des coefficients :

III.2.3.1. Action proportionnelle (P):

Il s'agit d'appliquer une correction proportionnelle à l'erreur corrigeant de manière instantanée tout écart de la grandeur à régler :

$$consigne(t) = K_p.\varepsilon(t) \xrightarrow{L} consigne(P) = K_p.\varepsilon(P) \qquad (3.3)$$

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.

Elle permet de vaincre les grandes inerties du système et diminue le temps de montée en donnant de la puissance au moteur (plus l'erreur est grande, plus on donne de puissance au moteur). 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 démesuré.

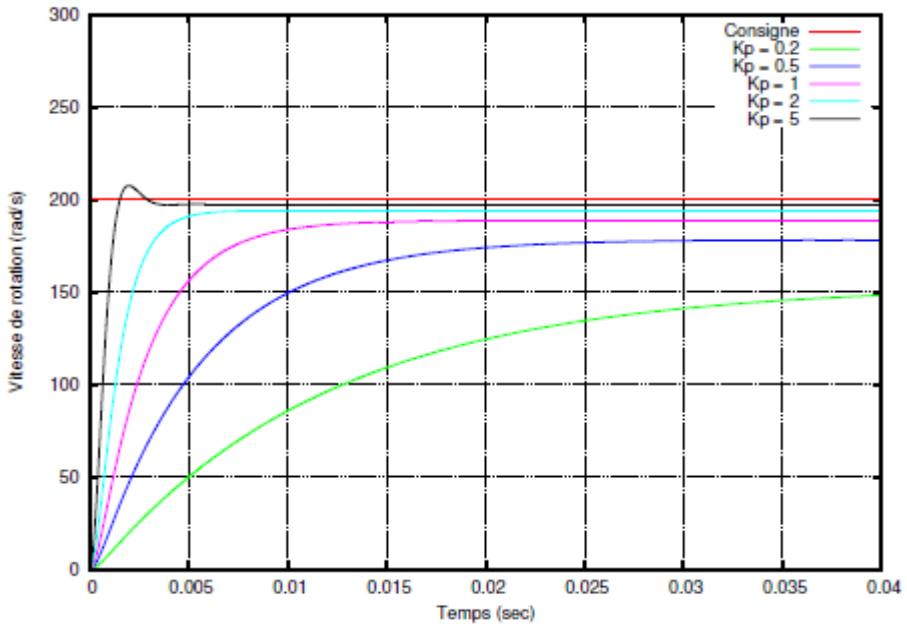


Figure 3.2 : *Influence du paramètre Kp sur la réponse à un échelon dans un asservissement de vitesse.*

Néanmoins, il faut toujours avoir une certaine tension aux bornes du moteur pour que celui-ci puisse tourner. C'est pourquoi, lorsque la réponse s'approche de la valeur demandée, l'erreur n'est plus assez grande pour faire avancer le moteur, ce qui fait que cette réponse n'atteint jamais vraiment la valeur demandée. Il subsiste alors une erreur statique, qui est d'autant plus faible que K_P est grand.

III.2.3.2. Action Proportionnelle Intégrale (PI) :

Il s'agit d'appliquer, en plus de l'action proportionnelle, un terme intégral :

$$consigne(t) = K_p.\varepsilon(t) + K_i.\int_0^t \varepsilon(\tau)d\tau \xrightarrow{L} consigne(P) = \left[K_p + K_i.\frac{1}{P}\right].\varepsilon(P) \qquad (3.4)$$

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 réponse se rapproche de la valeur demandée, 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. En plus, l'intégrale agissant comme un filtre sur le signal intégré, permet de diminuer l'impact des perturbations (bruit, parasites), et il en résulte alors un système plus stable. Malheureusement, un terme intégral trop important peut lui aussi entraîner un dépassement de la consigne, une stabilisation plus lente, voire même des oscillations divergentes.

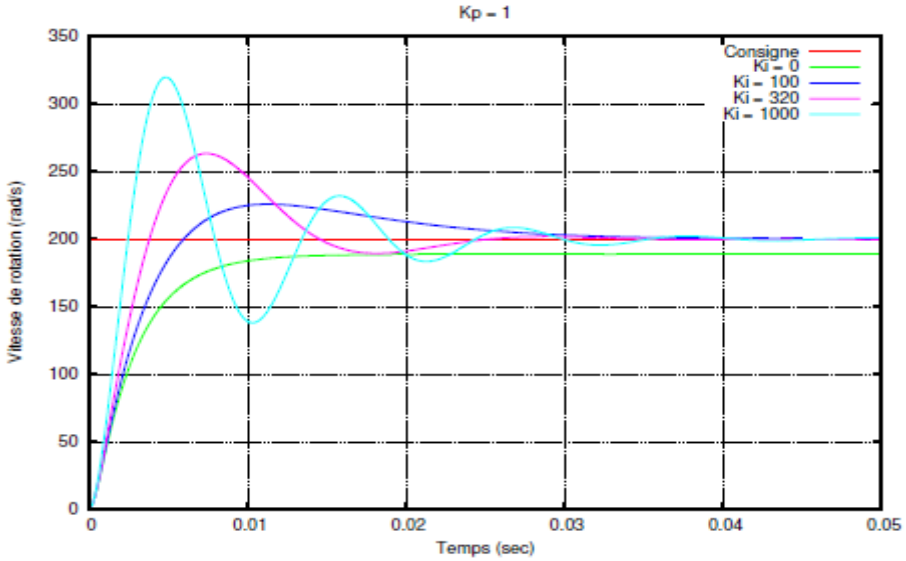


Figure 3.3 : *Influence du paramètre Ki sur la réponse à un échelon dans un asservissement de vitesse.*

III.2.3.3. Action Proportionnelle Intégrale Dérivée (PID) :

Les termes proportionnel et intégral peuvent engendrer un dépassement de la consigne et des oscillations. Cela implique pour le moteur des inversions de polarité, ce qui est loin d'être idéal. Pour limiter ce phénomène indésirable, on introduit un troisième élément : le terme dérivé. Son action va dépendre du signe et de la vitesse de variation de l'erreur, et sera opposée à l'action proportionnelle. Elle devient prépondérante aux abords de la valeur demandée 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.

$$consigne(t) = K_p.\varepsilon(t) + K_i.\int_0^t \varepsilon(\tau)d\tau + K_d\frac{d\varepsilon(t)}{dt} \qquad (3.5)$$

$$consigne(P) = \left[K_p + K_i.\frac{1}{P} + K_d.P\right].\varepsilon(P) \qquad (3.6)$$

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

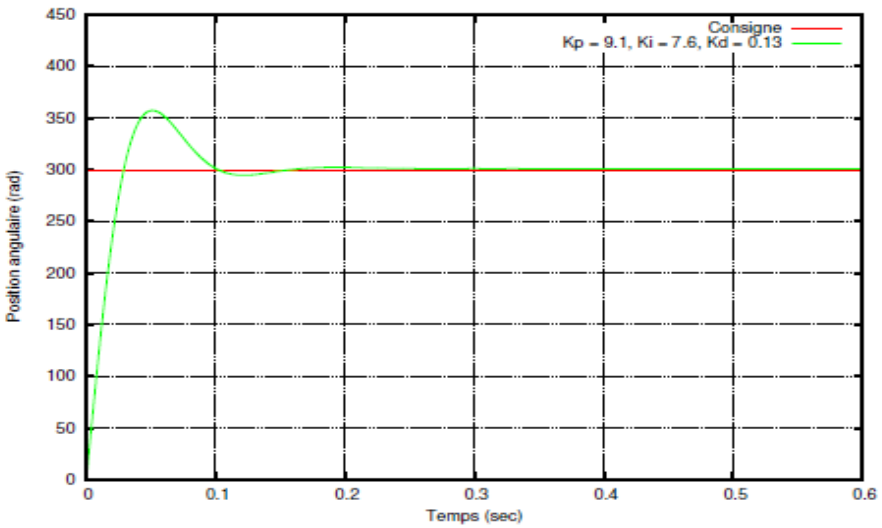


Figure 3.4 : *Réponse à un échelon dans un asservissement de position.*

III.2.4. Récapitulatif de l'action des coefficients :

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 : *récapitulatif de l'action des coefficients du régulateur PID.*

III.3. Algorithme PID implémenté dans l'interface UMAC [10] :

Un algorithme de commande PID en association avec une boucle de compensation Feedforward et d'un filtre éjecteur, est implémenté dans l'interface. Le schéma fonctionnel de ce correcteur est illustré sur la figure 3.5 :

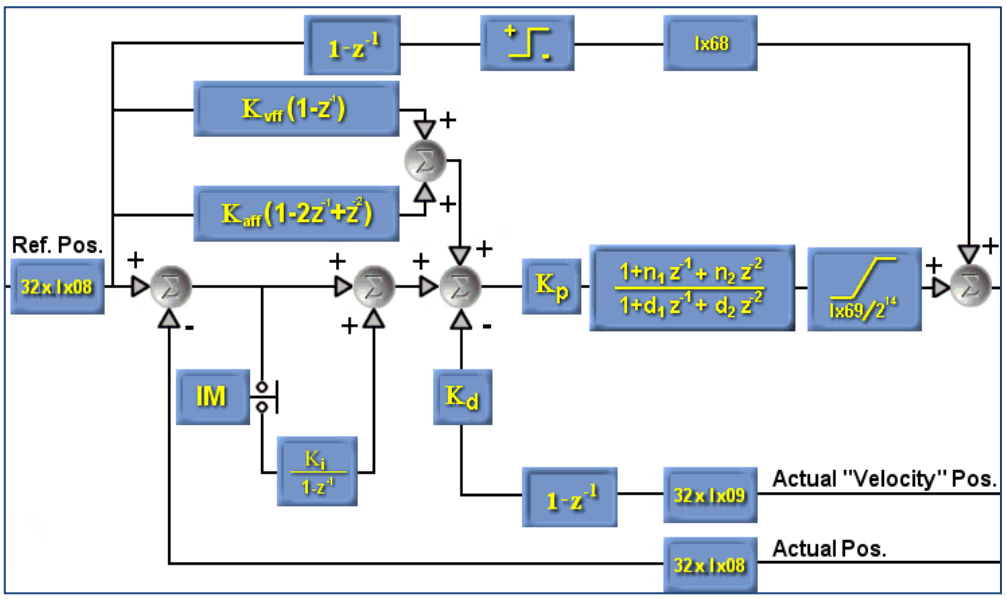


Figure 3.5 : *L'algorithme PID implémenté dans l'UMAC.*

En plus des gains du régulateur PID : le gain proportionnel K_p(**Ixx30**), le gain dérivé K_d(**Ixx31**) et le gain intégral K_i(**Ixx32**), on trouve les termes feedforward suivants :

- *Gain de vitesse feedforward* $Kvff$ (**Ixx32**) : 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* $Kaff$ (**Ixx35**) : réduit ou élimine les erreurs de poursuite dues à l'inertie du système qui sont proportionnelles à l'accélération.
- *Gain feedforward de frottement* (**Ixx68**) : 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.

On trouve également les paramètres de sécurité suivants :

- *DAC limite* (**Ixx69**) : limite le courant pour ne pas dépasser les valeurs maximales acceptables par l'amplificateur ou le moteur.
- *Limite fatal de l'erreur* (**Ixx11**) : quand l'erreur dépasse cette limite, le moteur xx est désactivé.

Le filtre éjecteur (Notch filter) est un filtre d'anti-résonnance ayant pour rôle de contrecarrer une résonance physique. Il est placé à la sortie du régulateur PID pour rejeter une bande de fréquence susceptible de contenir la fréquence de résonance. Le numérateur $N(Z)$ de ce filtre est de deuxième ordre, il constitue la partie anti-résonnance. Le dénominateur $D(Z)$ est aussi de deuxième ordre, il constitue la partie passe-bas. Les paramètres n1, n2, d1, d2 sont fixés par les variables **Ix36**, **Ix37**, **Ix38** et **Ix39** respectivement, ils prennent des valeurs entre $[-2, 2]$. Ils peuvent être déterminés automatiquement en utilisant le Pmac Tuning Pro.

Le paramètre *IM* (**Ixx34**) est mis à 0 pour activer l'action intégrale, et il est mis à 1 pour la désactiver.

III.4. Conclusion :

Nous avons vu dans ce chapitre que l'UMAC utilise la commande décentralisée pour le contrôle des robots. Cette méthode consiste à commander chaque moteur séparément et l'effet de couplage est vu comme une perturbation. Pour cela, un régulateur de type PID est implémenté dans l'interface.

Ce régulateur est associé à une boucle de compensation feedforward afin de diminuer ou d'éliminer l'effet des couplages (perturbations), et à un filtre éjecteur pour rejeter la fréquence de résonnance mécanique.

Le réglage et l'ajustement de ces paramètres se fait à l'aide de l'application Pmac Tuning Pro accompagnant le PEWIN32 Pro.

CHAPITRE IV :

Présentation du logiciel PEWIN32 PRO

IV.1. Introduction :

Nous avons vu, dans les précédents chapitres, que l'interface UMAC est dotée d'un logiciel permettant sa configuration, sa programmation et sa mise en œuvre. Il s'agit de l'exécutable principal PEWIN32 PRO.

Dans ce présent chapitre, nous présentons les différentes fonctionnalités de ce logiciel et les configurations nécessaires pour sa mise en marche.

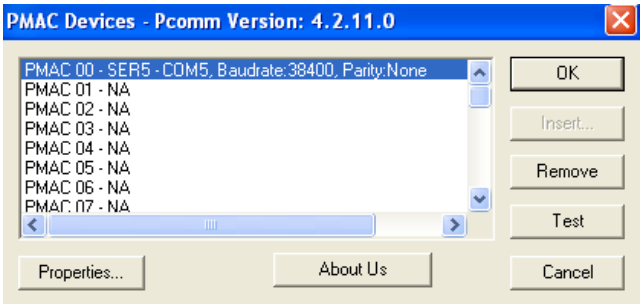
IV.2. Installation du logiciel et communication avec un ordinateur :

L'installation du logiciel se fait comme l'installation de n'importe quelle application sous Windows. Quand à l'établissement de la communication entre l'ordinateur et l'interface UMAC se fait par ajout d'un nouveau matériel dans le panneau de configurations selon le cheminement suivant :

- Ajout d'un nouveau matériel dans le panneau de configuration.
- Sélectionner « Non, le périphérique ne figure pas dans la liste ».
- Sélectionner « Non, je veux choisir le matériel à partir d'une liste ».
- Sélectionner « **Delta Tau Data Systems, INC**».

Si le périphérique est bien installé, on redémarre l'ordinateur.

A présent, il est nécessaire d'établir la communication entre l'UMAC et le PEWIN32 Pro. Pour ce faire, on utilise l'application **UmacConfigPro** accompagnant le PEWIN32 Pro. En cliquant sur « select », on aperçoit la fenêtre suivante :



On clique sur «insert » et on choisit le port utilisé, puis on configure les paramètres de ce dernier dans le volet « propriétés… » :

- Vitesse de transmission : 38400 bits/sec.
- Parité : non.
- Sélectionner CTS (Clear To Send).
- Dans le contrôle de RTS (Ready To Send), sélectionner Enable.
- Pour le temps de sortie (Timeouts) : Character 2000 ms, Flush 30 ms.

L'application **Turbo Setup Pro** est utilisée pour configurer et initialiser automatiquement les différents paramètres suivant le matériel disponible et les moteurs utilisés.

IV.3. Configurations nécessaires pour les moteurs PITTMAN 9136027 [8] :

Après avoir branché les moteurs et leurs encodeurs, il faut configurer l'interface selon les moteurs utilisés et la commande appropriée en utilisant les **variables I** facilement accessibles par le PEWIN 32 PRO.

Les configurations nécessaires pour les moteurs PITTMAN sont les suivantes :

- Ix00=1 : activer le moteur x ;
- Ix01=0 : désactiver la commutation du moteur x ;
- I72x6=1 : mettre le canal x de la DSPGATE en sortie DAC (commande en courant) ;
- I72x0= 3 (ou 7) : si la limite de l'erreur fatale (Ixx11) est atteinte, inverser le sens de comptage ;
- Ix24=$200001 : si +LIM et –LIM ne sont pas à la masse, il est nécessaire de faire cette configuration pour ne pas déclencher les fins de courses.

IV.4. Utilisation de l'application Pmac Tunning Pro :

Le Pmac Tunning Pro est utilisé pour la détermination des paramètres des différentes boucles d'asservissement et de régulation des moteurs. Et ce, suivant ces différentes étapes :

IV.4.1. Calibrage de la sortie du convertisseur DAC :

Le but de cette étape est de déterminer les valeurs extrémales à la sortie du DAC et affecter les variables Ixx29 et Ixx79.

IV.4.2. Test en boucle ouverte :

On peut visualiser la réponse en boucle ouverte du moteur à un signal de référence carrée périodique.

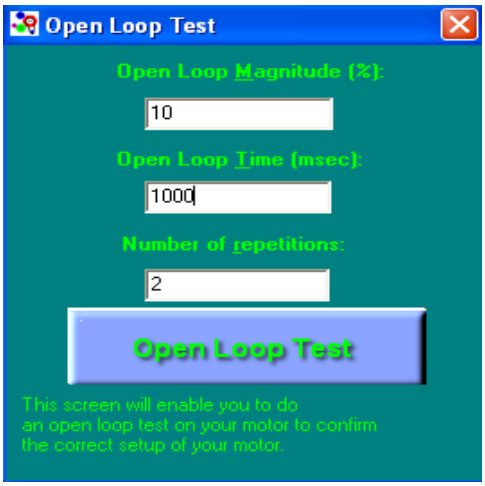


Figure 4.1 : *Test en boucle ouverte.*

On obtient la réponse présentée à la figure 4.2 :

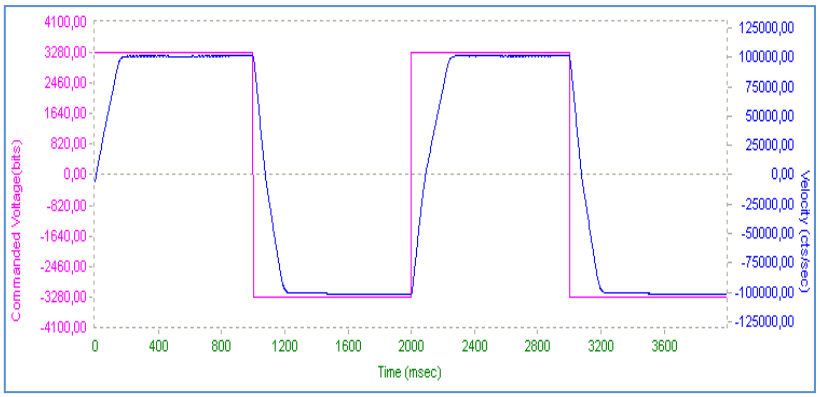


Figure 4.2 : *Réponse en boucle ouverte.*

L'objectif de cette étape, en plus de la réponse en boucle ouverte, et de vérifier le bon branchement des moteurs ainsi que la bonne configuration matérielle et logicielle de l'interface.

IV.4.3. Auto estimation des paramètres du régulateur PID :

On peut déterminer automatiquement les paramètres du régulateur PID en utilisant la rubrique « Position Loop\ Regular PID\ Auto Tuning » :

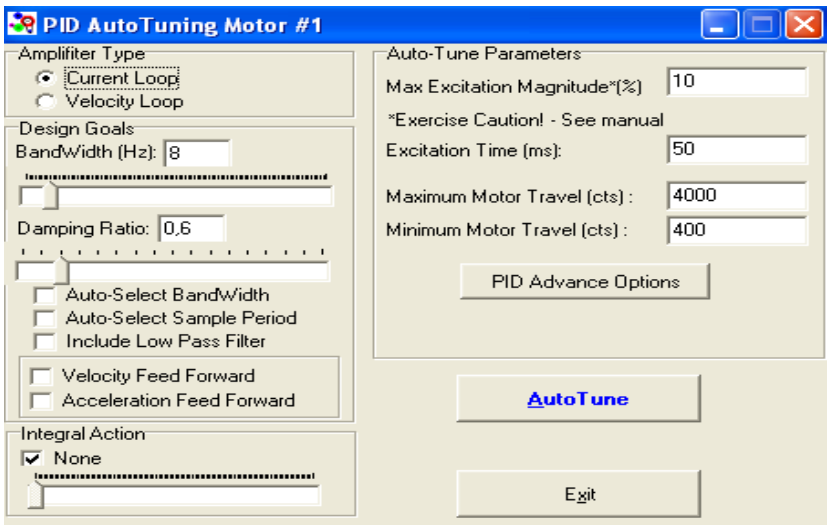


Figure 4.3 : *Auto ajustement du régulateur PID.*

- Sélectionner « Auto-select Bandwich ».
- Sélectionner un « Damping Ratio » entre 0.8 et 1.
- Appuyer sur « Auto Tune » pour obtenir une estimation des paramètres optimaux du régulateur PID.

La figure 4.4 présente les paramètres obtenus :

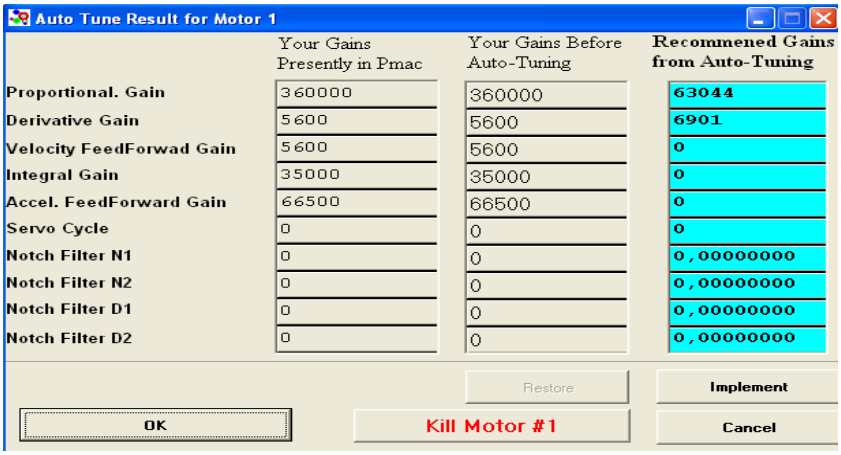


Figure 4.4 : *Paramètres obtenus après auto ajustement du PID.*

- Appuyer sur « Implement » pour implémenter les paramètres obtenus.

Ensuite, On visualise la réponse en boucle fermée :

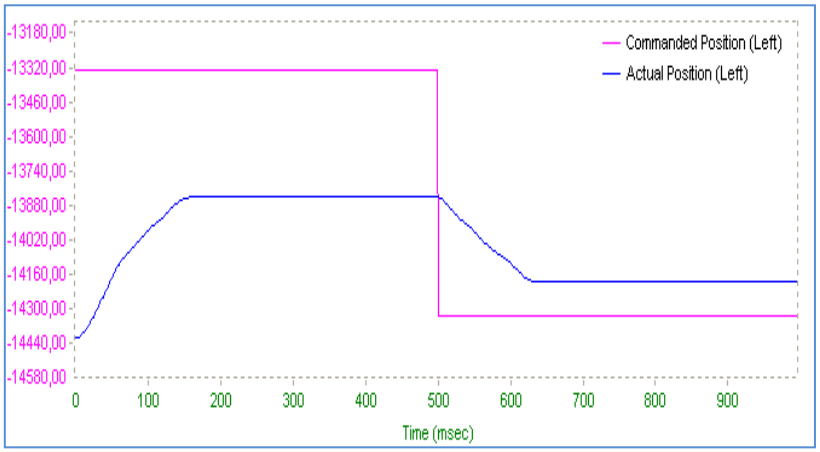


Figure 4.5 : *Réponse en boucle fermée après auto ajustement.*

On remarque que la réponse obtenue ne suit pas exactement la consigne. Pour améliorer cette réponse, il faudrait jouer sur les paramètres du régulateur PID, en utilisant la rubrique « PID interactive Tuning », jusqu'à l'obtention d'une réponse satisfaisante.

IV.4.4. Utilisation de l'estimation interactive des paramètres PID :

Dans le menu « Position Loop\ Regular PID\ Interactive Tuning »

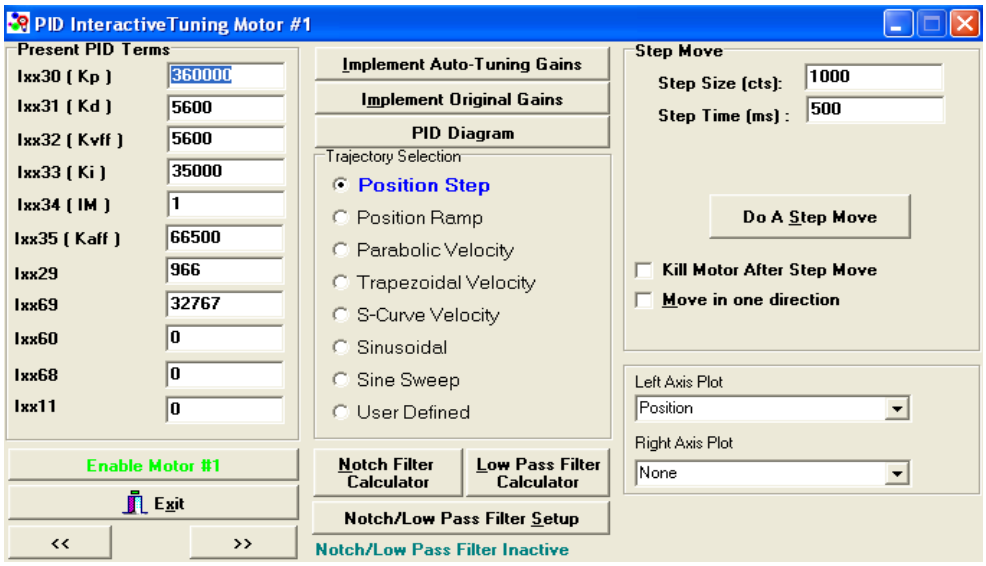


Figure 4.6 : *Ajustement interactif des paramètres du PID.*

Cette fenêtre nous permet de changer les paramètres du régulateur PID et de la boucle de compensation feedforward jusqu'à l'obtention de résultats satisfaisants (la méthode suivie pour la détermination des paramètres est détaillée dans l'annexe D).

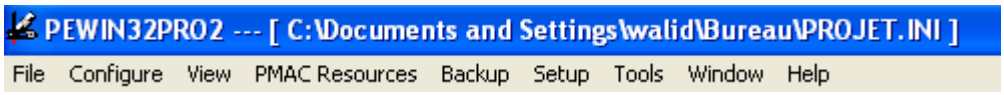
On peut également déterminer les paramètres des filtres « Notch/Low Pass filter ».

IV.5. Utilisation du PEWIN32 PRO [12] :

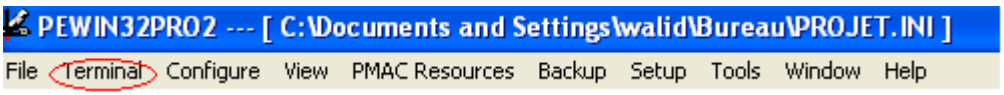
Le PEWIN32 Pro est un logiciel de configuration et de programmation de l'UMAC. Il comporte un terminal, un éditeur de programmes et un espace de travail (Workspace). Il contient aussi une série d'outils facilitant la tâche lors du développement des applications avec le PMAC et ses accessoires. Dans ce qui suit, nous allons présenter l'environnement de travail de ce logiciel.

IV.5.1. Le Menu principal :

Il utilise un menu dynamique, qui change en fonction des fenêtres ouvertes. Le menu standard est présenté sur la figure suivante :

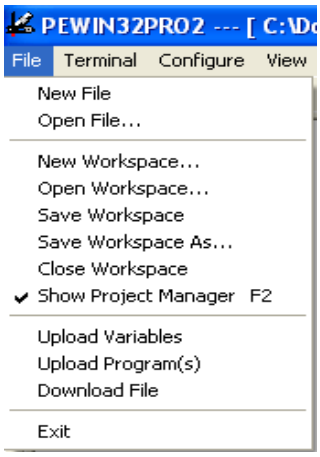


En ouvrant une nouvelle fenêtre (un terminal par exemple), le menu bascule vers l'affichage suivant :



Ce menu comporte les onglets suivants :

File Menu :

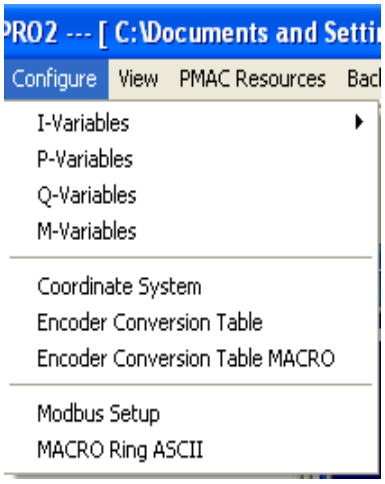


Il permet :

- De créer un nouveau fichier ou d'en ouvrir un déjà existant ;
- La gestion de l'espace de travail (créer, ouvrir, sauvegarder et fermer un Workspace) ;

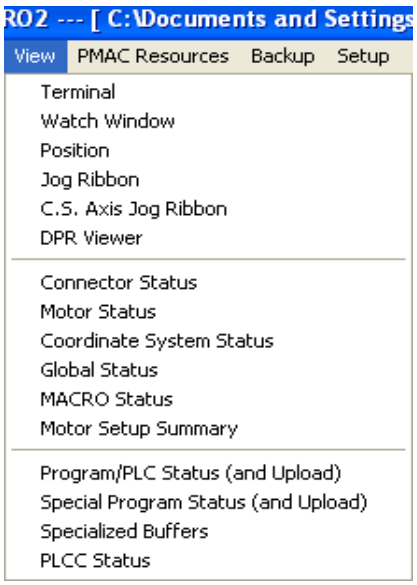
- De charger les différents programmes dans l'UMAC ;

Configure Menu :



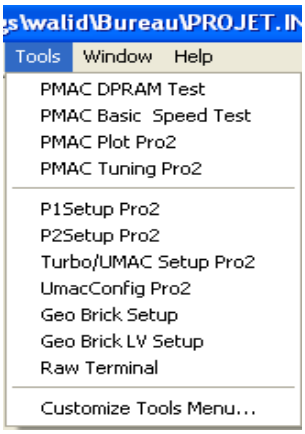
Il permet d'accéder aux différentes variables et de définir des systèmes en coordination. Il permet aussi de modifier, mettre à jour, récupérer et sauvegarder les tables de conversion de l'encodeur.

View Menu :



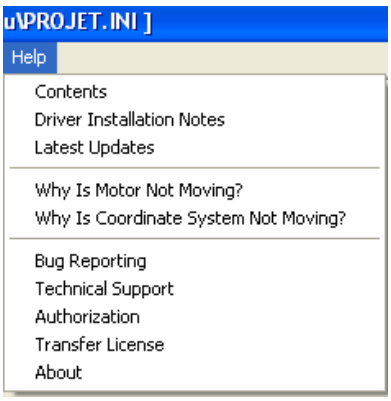
Il permet de visualiser les différentes fenêtres, telles que, le terminal ou la fenêtre position qui indique les positions actuelles des moteurs.

Tools Menu :



A partir de cet onglet, on peut accéder directement aux différentes applications accompagnant le PEWIN32 PRO.

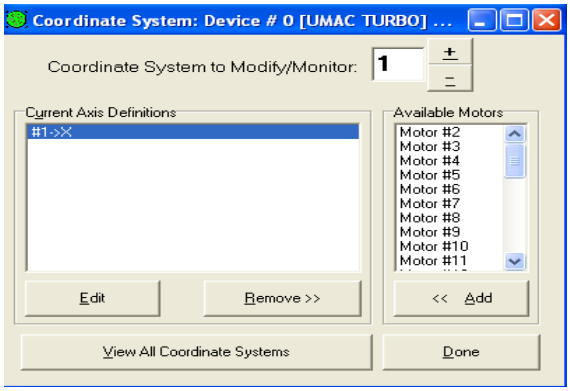
Help Menu :



Le Help Menu permet de récupérer des informations en ligne sur le PMAC, le PEWIN et ses fonctionnalités. On peut également accéder aux routines de diagnostique fournies par le logiciel (Why Is Motor Not Moving ?, Why Is Coordinate System Is Not Moving ?).

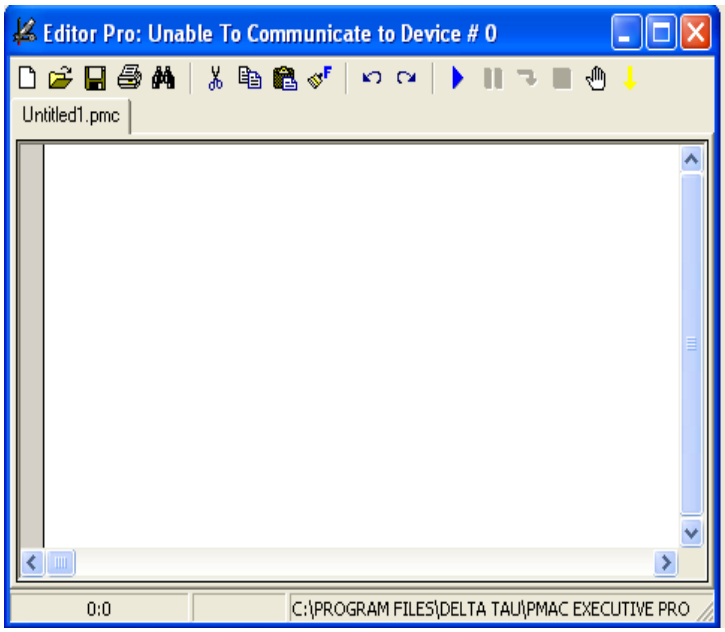
IV.5.2. Les étapes de création d'un programme :

- Définir un système en coordination à partir du « Configure Menu\ Coordinate System »



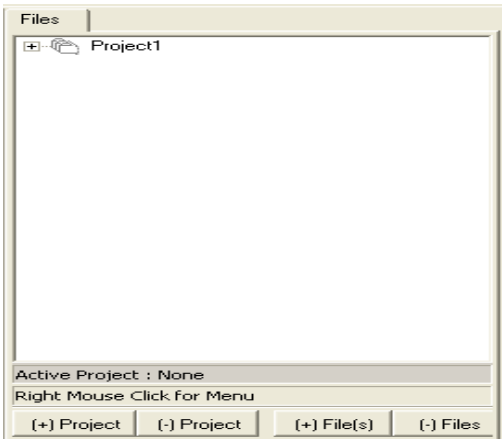
Il faut indiquer le numéro du SC et les moteurs qui lui sont associés ainsi que l'axe lié à chaque moteur.

- Ecrire un programme : l'éditeur de programmes est accessible à partir du « File Menu\ New File».



Il faut sauvegarder le programme à la fin de son écriture.

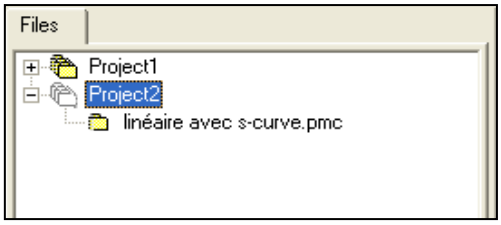
- Créer un nouveau projet :



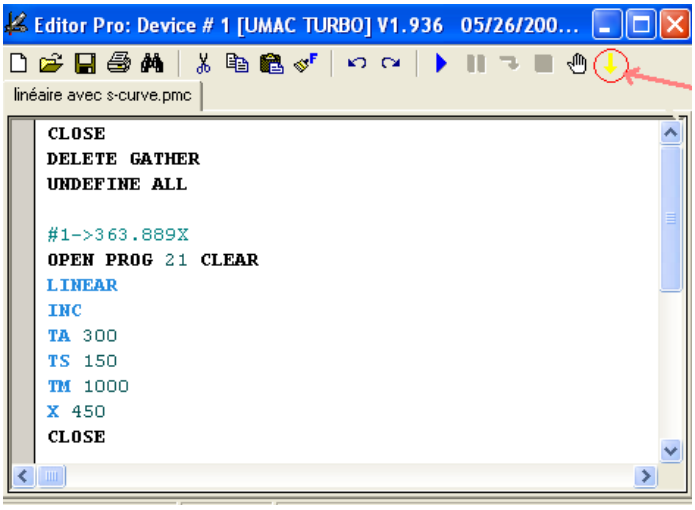
Pour ajouter un nouveau projet (respectivement supprimer un déjà existant), on clique sur « (+) Project » (respectivement « (-) Project »).

Ensuite, on peut ajouter (respectivement enlever) un programme en cliquant sur « (+) File(s) » (respectivement « (-) Files »).

A présent on aura l'affichage suivant :



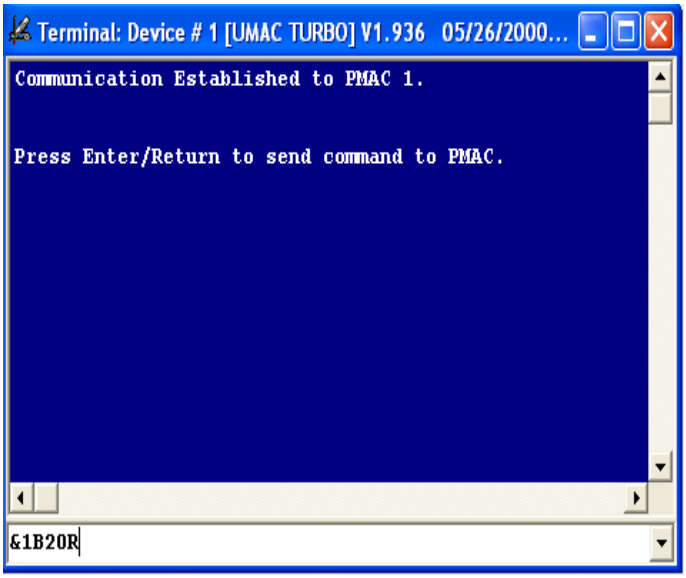
- Ouvrir le programme dans l'éditeur de texte, puis le charger dans l'UMAC en appuyant sur la flèche jaune comme le montre la figure suivante :



- Vérifier que le programme ne contient pas d'erreurs.

IV.5.3. Exécution des programmes :

L'exécution des programmes se fait à partir du terminal : « View Menu\ Terminal ». Ce terminal permet d'exécuter des commandes en ligne.



Pour exécuter un programme de mouvement, il faut taper les commandes suivantes :

- « #xJ/ » : fermer la boucle du moteur x.
- « &n» où n désigne le SC.
- « Bm » : se brancher au début du programme de mouvement m.
- « R » : lancer l'exécution du programme.

On trouve également les commandes : la commande « A » pour décélérer les moteurs, la commande « Q » pour arrêter le programme à la fin de son exécution et la commande « K » pour l'arrêt immédiat des moteurs.

IV.5.4. Utilisation de l'application Pmac Plot32 Pro :

A partir du Menu du PEWIN32 PRO : « Tools\Pmac Plot Pro ».

Cette application est utilisée pour l'acquisition et représentation graphique des données.

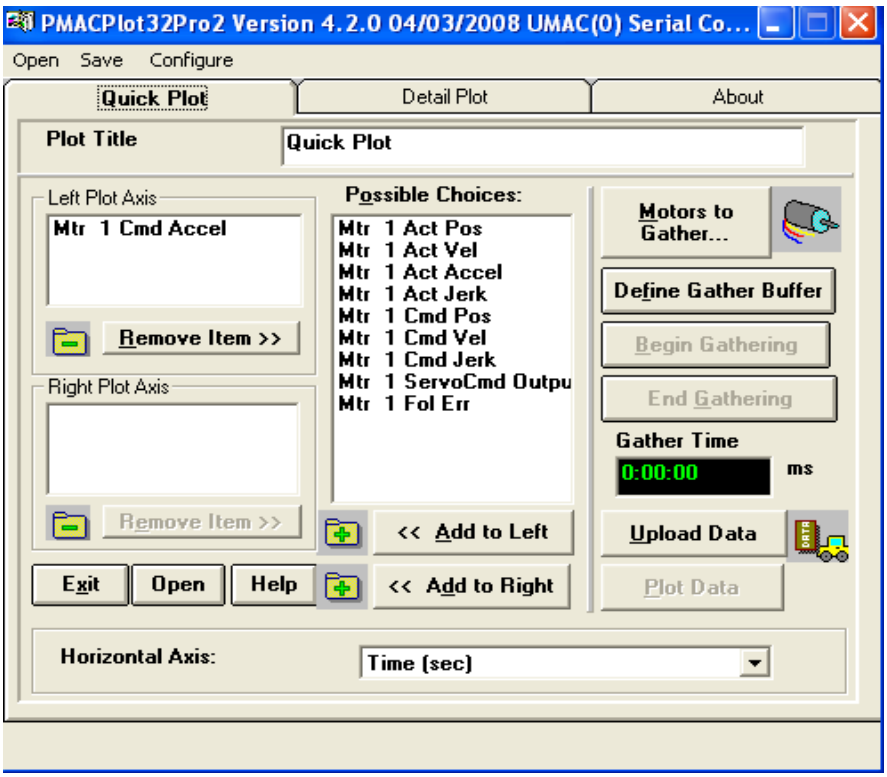


Figure 4.7 : le Pmac Plot32 Pro.

Pour utiliser convenablement cette application, on procède comme suit :

- Sélectionner « Motor to Gather » pour choisir les moteurs effectuant le mouvement.
- Sélectionner le profil à tracer (position, vitesse,…etc.) dans la zone « Possible Choices » et appuyer sur « Add to Left ».
- Selectionner « Define Gather Buffer ».
- Sélectionner « Begin Gathering » pour commencer l'acquisition des données.
- Exécuter un programme de mouvement depuis le terminal de PEWIN32 PRO.
- A la fin du mouvement, sélectionner « End Gathering ».
- Sélectionner « Upload Data » pour charger les données dans l'ordinateur.
- Sélectionner « Plot Data » pour tracer le profil choisi.

IV.6. Conclusion :

Dans ce chapitre, nous avons détaillé les fonctionnalités du logiciel PEWIN32 PRO qui nous offre un environnement adapté pour développer des applications complexes de

manière interactive telles que la commande des moteurs électriques ou encore la commande des robots (en introduisant les différents modèles du robot).

CHAPITRE V :

Application

V.1. Introduction :

Dans ce présent chapitre, nous allons mettre en œuvre ce qui a été traité dans les chapitres précédents. Nous commençons par la détermination des paramètres optimaux du régulateur PID+Feed Forward assurant les performances souhaitées. Par la suite, nous allons exécuter de simples mouvements dans l'espace articulaire afin de mettre en évidence les types de mouvements que l'UMAC peut effectuer. Ces derniers seront utilisés pour la génération de mouvements dans l'espace opérationnel dont le but de planifier quelques tâches pour notre robot. Nous terminons par l'utilisation d'un PLC pour pouvoir gérer les différents programmes de mouvements en facilitant leur exécution.

V.2. Détermination des paramètres optimaux du régulateur PID+Feed Forward :

Comme vu précédemment, l'algorithme de commande implémenté dans l'UMAC est le régulateur PID+Feed Forward dont les paramètres sont déterminés à partir de l'application Pmac Tunning Pro. Ceci est possible soit par une auto estimation en utilisant l'*Auto Tunning*, soit par une estimation interactive en utilisant *PID Interactive Tunning*, qui permet de varier les paramètres jusqu'à l'obtention des performances souhaitées.

La réponse obtenue par l'auto estimation n'est pas satisfaisante (voir le chapitre précédent), d'où le recours à l'utilisation de l'estimation interactive des paramètres.

Pour ce faire, on procède empiriquement jusqu'à l'obtention des paramètres idéaux du régulateur. La méthode suivie se base sur les observations données dans l'annexe D.

Les réponses obtenues après l'implémentation de ces paramètres sont présentées sur les figures suivantes :

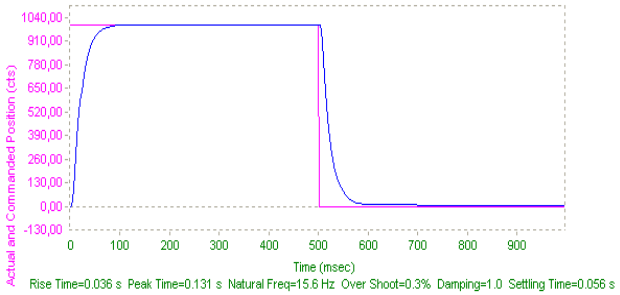


Figure 5.1 : *Réponse à un échelon*

On remarque que la réponse à un échelon (figure 1.5) n'a pas d'erreur statique, et elle présente un faible temps de montée de 0.036 s avec un dépassement de 0.3%.

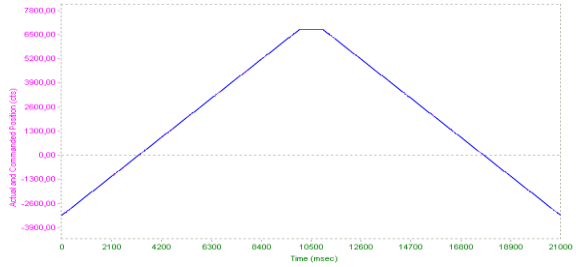


Figure 5.2 : *Réponse à une rampe*

La figure 5.2 montre que les paramètres choisis assurent une bonne poursuite de la référence.

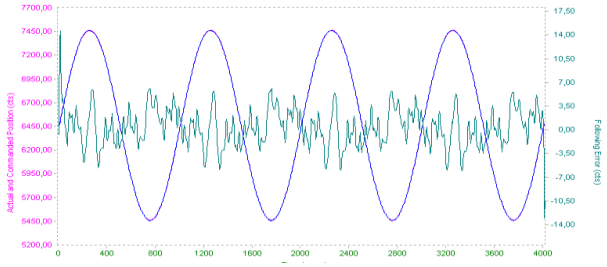


Figure 5.3 : *Réponse à un signal sinusoïdal*

On voit que les paramètres du régulateur assurent également la décorrélation de l'erreur.

Voici quelques réponses en vitesses et en accélérations :

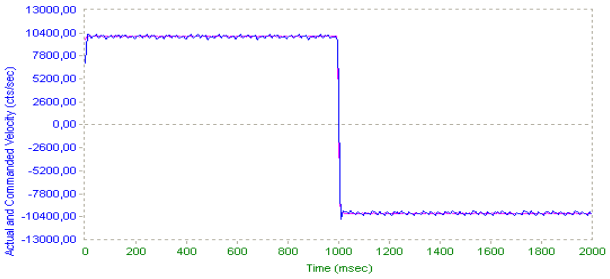


Figure 5.4 : *Réponse en vitesse à une rampe*

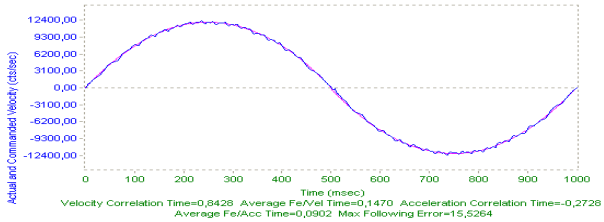


Figure 5.5 : *Réponse en vitesse à une parabole*

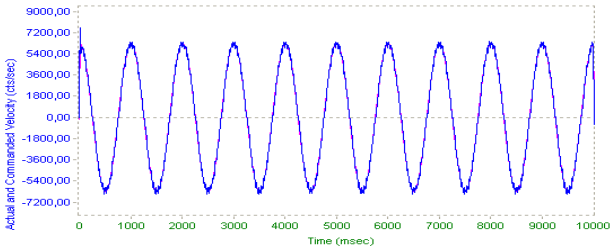


Figure 5.6 : *Réponse en vitesse à une sinusoïde*

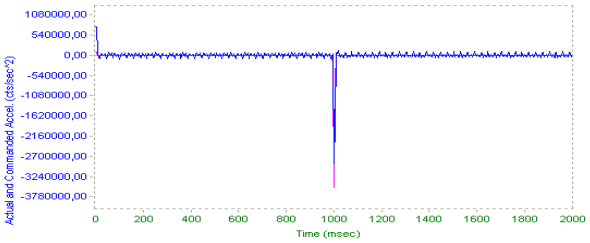


Figure 5.7 : *Réponse en accélération à une rampe*

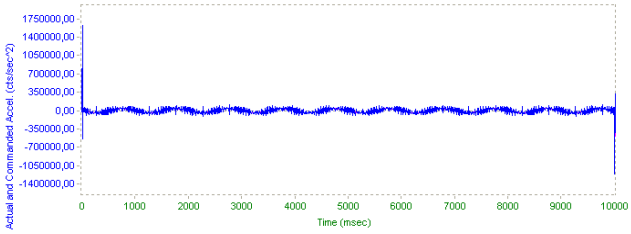


Figure 5.8 : *Réponse en accélération à une sinusoïde*

V.3. modifications apportées à la structure :

Vue que la structure dont nous disposons présente plusieurs défauts, engendrés par les jeux mécaniques, frottements et le fléchissement du premier segment qui ne supporte pas la charge composée par les deux autres moteurs et le deuxième segment, nous avons apporté les modifications suivantes pour remédier à ces problèmes :

- Pour éliminer les jeux mécaniques, nous avons attaché les segments aux moteurs avec des liaisons encastrées très rigides.
- Nous avons également placé des roulements pour diminuer les frottements.
- Pour éliminer le fléchissement, nous avons placé un palier au niveau du coude pour soutenir le premier segment et contrecarrer la charge.
- Pour diminuer la charge, nous avons remplacé le premier segment (en acier) par un autre en aluminium et le deuxième segment par un autre en plastique.
- Nous avons remplacé aussi le troisième moteur (déplacement vertical) par un dispositif électromécanique plus léger. Il consiste en un électro-aimant permettant d'effectuer un petit déplacement vertical (très pratique pour un robot traceur). Sa commande est assurée par l'interface UMAC.

V.4. Génération de mouvements dans l'espace articulaire :

Après l'implémentation du régulateur PID + Feed Forward qui assure les performances désirées, nous allons illustrer les différents mouvements dans l'espace articulaire présentés précédemment, à savoir les mouvements linéaire, Spline et le mode PVT.

Nous rappelons que la trajectoire de l'organe terminal est imprévisible, car les moteurs sont commandés séparément. On ne s'intéresse donc qu'aux mouvements effectués par les moteurs.

Dans ce qui suit, nous allons présenter les programmes de mouvement ainsi que les résultats obtenus sur les profils des position, vitesse et accélération.

V.4.1. Mouvement linéaire sans S-curve :

Le moteur est commandé pour faire un déplacement de 450^0, ce qui est équivalent à 163750,05 incréments. Le programme de mouvement est le suivant :

```
CLOSE
DELETE GATHER
UNDEFINE ALL
#1->363.889X
OPEN PROG 20 CLEAR
LINEAR
INC
TA 300
TS 0
TM 1000
X 450
CLOSE
```

Les profils de position, vitesse et accélération sont comme suit :

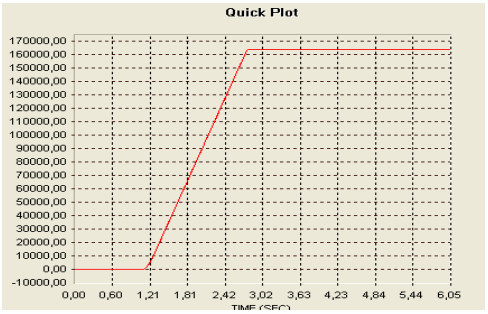


Figure 5.9 : *profil de la position commandée*

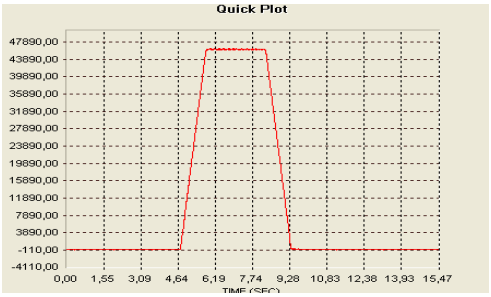


Figure 5.10 : *profil de la vitesse commandée*

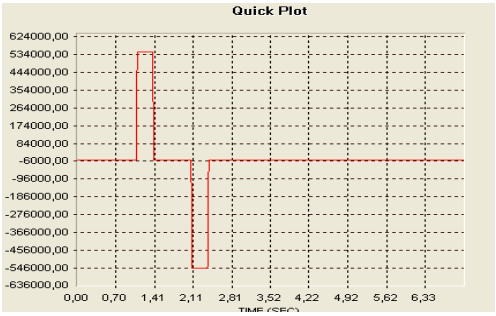


Figure 5.11 : *profil de l'accélération commandée*

V.4.2. Mouvement linéaire avec S-curve :

Nous avons refait la même chose qu'avec le mouvement précédent, avec un temps de courbure TS = 150 ms. Le programme de mouvement est le suivant :

```
CLOSE
DELETE GATHER
UNDEFINE ALL
#1->363.889X
OPEN PROG 21 CLEAR
LINEAR
INC
TA 300
```

```
TS 150
TM 1000
X 450
CLOSE
```

Les profils de position, vitesses et accélération sont comme suit :

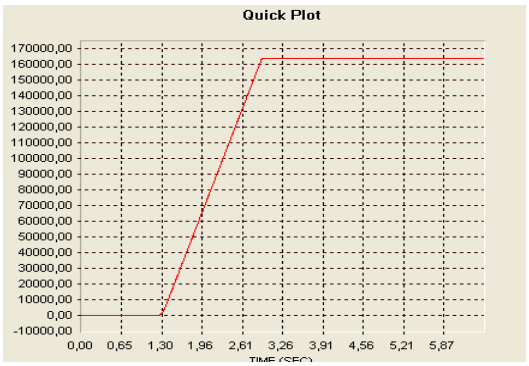


Figure 5.12 : *profil de la position commandée*

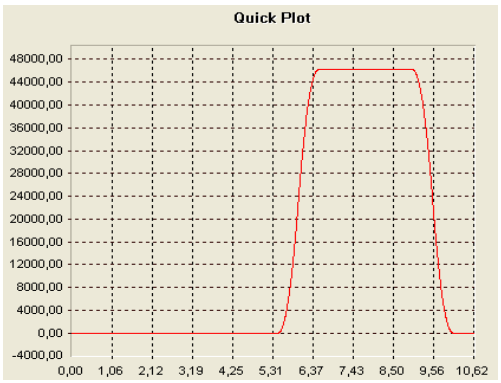


Figure 5.13 : *profil de la vitesse commandée*

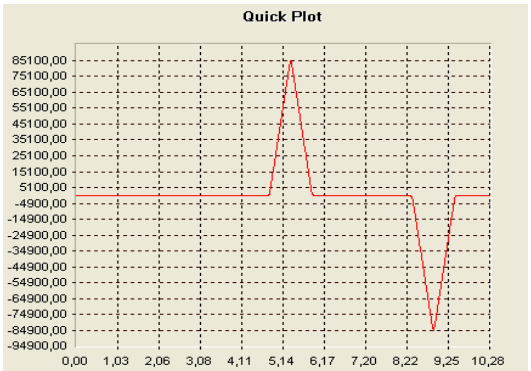


Figure 5.14 : *profil de l'accélération commandée*

V.4.3. Mouvement Spline :

Le moteur doit faire un déplacement de 270^0 (98250,03 incréments) dans le sens positif, et de 145^0 (52763,905 Incréments) dans le sens négatif. Et grâce au mode Spline, UMAC va raccorder les deux mouvements de façon à lisser la vitesse et l'accélération. Ceci va permettre d'éviter les changements brusques de l'accélération. Le programme de mouvement est le suivant :

```
CLOSE
DELETE GATHER
UNDEFINE ALL
#1->363.889X
OPEN PROG 22 CLEAR
LINEAR
INC
SPLINE1
TA 200
TS 30
TM 1000
X 270
X -145
CLOSE
```

Les profils de position, vitesse et accélération sont comme suit :

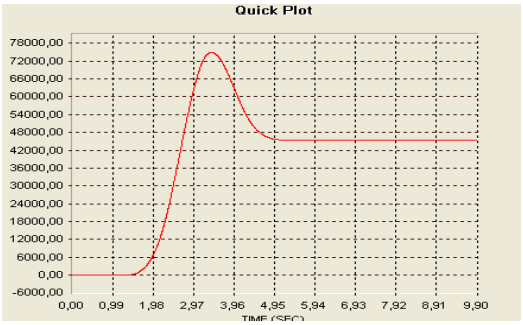


Figure 5.15 : *profil de la position commandée*

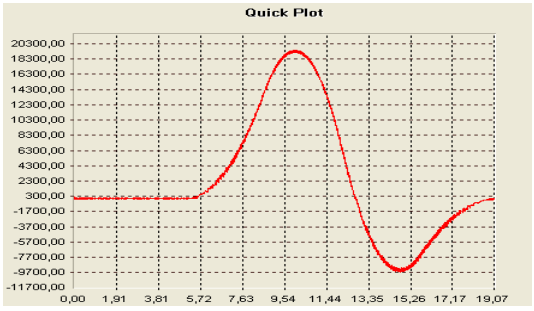


Figure 5.16 : *profil de la vitesse commandée*

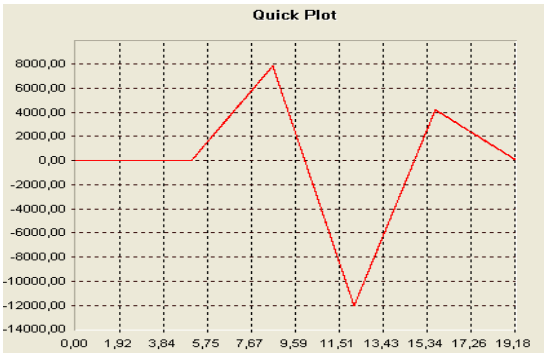


Figure 5.17 : *profil de l'accélération commandée*

V.4.4. Mouvement en mode PVT :

Le moteur doit faire un déplacement de 270^0 (98250,03 incréments) dans le sens positif avec une vitesse finale de 50 incréments/ms, et de 180^0 (65500,02 Incréments) dans le sens négatif avec une vitesse finale nulle. Le programme de mouvement est le suivant :

```
CLOSE
DELETE GATHER
UNDEFINE ALL
#1->363.889X
OPEN PROG 23 CLEAR
INC
PVT 1000
X 270 : 50
X -180 : 0
CLOSE
```

Les profils de position, vitesse et accélération sont comme suit :

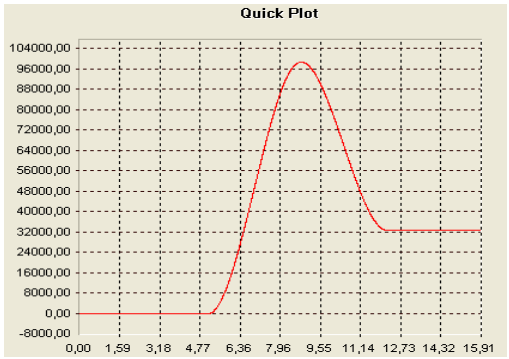


Figure 5.18 : *profil de la position commandée*

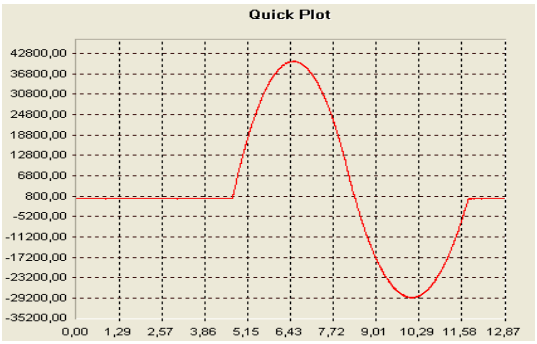


Figure 5.19 : *profil de la vitesse commandée*

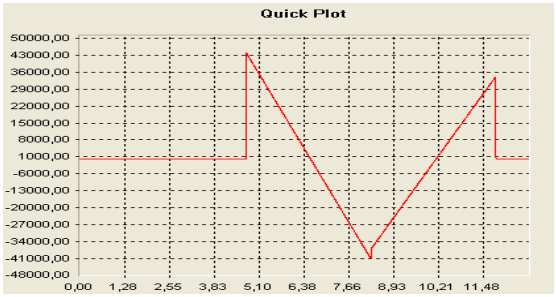


Figure 5.20 : *profil de l'accélération commandée*

La génération de mouvement dans l'espace articulaire permet de prendre en compte les limitations en vitesse et en accélération. On peut éviter les changements brusques de l'accélération en effectuant un mouvement avec S-curve (TS ≠ 0) ou avec le mouvement Spline. Ceci permet de ne pas exciter les fréquences de résonance du système. On peut aussi avoir un contrôle plus direct sur la vitesse avec le mode PVT.

V.5. Génération de mouvements dans l'espace opérationnel :

Dans cette partie, nous présenterons les programmes géométriques qui sont nécessaires pour la planification de tâches dans l'espace opérationnel. Par la suite, nous présenterons les différentes trajectoires réalisées avec leurs programmes de mouvements, ainsi que leurs résultats. Nous finirons par un PLC qui sera chargé de gérer certains programmes de mouvements.

V.5.1. Programmes géométriques :

Avant l'exécution de n'importe quel mouvement et la planification de trajectoires dans l'espace opérationnel, on doit, en premier lieu, implémenter les programmes géométriques direct et inverse afin de prendre en considération la structure du robot. On va aussi implémenter le programme cinématique inverse pour avoir un bon contrôle sur les vitesses.

Nous allons présenter les différents programmes géométriques, tout en regroupant le programme cinématique inverse avec le programme géométrique inverse dans un seul programme.

V.5.1.1. Le programme géométrique direct :

```
I15=0                       ; les calculs trigonométriques en degré
&1                          ; SC 1 adressé
Q91=186.64                  ; la longueur L1 (mm)
Q92=242.64                  ; la longueur L2 (mm)
Q93=363.889                 ; Comptes par degré
&1 OPEN FORWARD
CLEAR
Q7=Q91*COS(P1/Q93)+Q92*COS((P1+P2)/Q93)      ; position X
Q8=Q91*SIN(P1/Q93)+Q92*SIN((P1+P2)/Q93)      ; position Y
CLOSE
```

V.5.1.2. Programme géométrique inverse et cinématique inverse :

```
&1
#1->I                       ; moteur 1 affecté au MGI et au MCI
#2->I                       ; moteur 2 affecté au MGI et au MCI
M5182->Y:$00105C,2          ; bit indicateur du temps de calcul dans le SC 1
Q94=Q91*Q91+Q92*Q92         ; L12+L22
Q95=2*Q91*Q92               ; 2*L1*L2
Q96=Q91*Q91-Q92*Q92         ; L12-L22
&1 OPEN INVERSE
CLEAR
Q20=Q7*Q7+Q8*Q8                          ; X2+Y2
Q21=(Q20-Q94)/Q95                        ; cos(θ2)
IF (Q21<0.9998 AND Q21>-0.76)            ; si l'angle 0 < θ2 < 140°
            Q22=ACOS(Q21)                ; l'angle θ2 en degré
            Q0=Q7                        ; pour utiliser la fonction ATAN2
            Q23=ATAN2(Q8)                ; θ1+ θ0= arctan(Y/X)
            Q24=ACOS((Q20+Q96)/(2*Q91*SQRT(Q20)))     ; l'angle θ0 en degré
            Q25=Q23-Q24                               ; l'angle θ1 en degré
            P1=Q25*Q93          ; consigne pour le moteur 1 en comptes
            P2=Q22*Q93          ; consigne pour le moteur 2 en comptes
            ELSE
                  M5182=1    ; déclenchement de l'erreur du temps de calcul
IF (Q10=1)                    ; activations des calculs cinématiques?
            Q26=SIN(Q22)                  ; sin(θ2)
            IF (Q26>0.0175)               ; s'il n'y a pas de singularité
                        Q27=Q91*Q92*Q26   ; L1*L2* sin(θ2)
                        Q28=COS(Q25+Q22)  ; cos(θ1 +θ2)
                        Q29=SIN(Q25+Q22)  ; sin(θ1 +θ2)
                        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    ; déclenchement de l'erreur du temps de calcul
            ENDIF
ENDIF
ENDIF
CLOSE
```

Ce programme géométrique désactive n'importe quel programme de mouvement dans le cas où le bras du robot serait tendu au maximum ($0 < \theta_2 < 140°$). Il permet aussi un meilleur ajustement sur les vitesses angulaires des moteurs.

Une fois que les MGI et MGD seront implémentés dans l'interface et que les calculs géométriques et cinématiques seront activés (I5150 = 1 et Q10 = 1), n'importe quel programme exécuté par le SC 1 utilisera cette transformation géométrique.

On note que ces programmes géométriques ne tiennent pas compte du déplacement vertical, assuré par l'électro-aimant. Ce dispositif est commandé indépendamment en utilisant un PLC. Ce dernier maintient le stylo à la position haute si la variable « P100 = 0 » et à la position basse si la variable « P100 = 1 ».

```
CLOSE
DELETE GATHER
OPEN PLC 7
CLEAR
IF (P100=1)
            CMD"#3J/"
ELSE
            CMD"#3K"
ENDIF
CLOSE
ENABLE PLC 7
```

V.5.2. Planification des tâches :

Nous allons utiliser notre robot pour l'écriture de caractères, une chaine de caractères et pour la génération de quelques trajectoires afin de réaliser des dessins plus au moins compliqués.

Dans ce qui suit, nous présenterons les différents programmes de mouvements pour chaque tâche, ainsi que le résultat obtenu à partir du Pmac Plot Pro.

- **Ecriture d'un caractère :**

Le programme :

```
DELETE GATHER
UNDEFINE ALL
OPEN PROG 100 CLEAR
LINEAR
ABS
TA 600
TS 450
TM 3000
X -140 Y 280
DWELL 500
P100=1
LINEAR
INC
TA 300
TS 550
TM 2000
X -40 Y 0
X 0 Y -80
X 40 Y 0
DWELL 500
P100=0
LINEAR
INC
X -40 Y 40
DWELL 500
P100=1
LINEAR
INC
X 45 Y 0
DWELL 500
P100=0
CLOSE
```

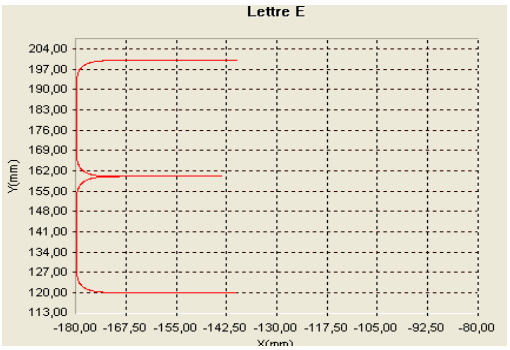


Figure 5.21 : *Ecriture d'un caractère*

- **Ecriture d'une chaine de caractères :**

  Le programme :

```
CLOSE
DELETE GATHER
UNDEFINE ALL

OPEN PROG 101 CLEAR
LINEAR
ABS
TA 600
TS 450
TM 3000
X -140 Y 200
DWELL 500
P100=1
LINEAR
INC
TA 300
TS 550
TM 2000
X -40 Y 0
X 0 Y -80
X 40 Y 0
DWELL 500
P100=0
LINEAR
INC
X -40 Y 40
DWELL 500
P100=1
LINEAR
INC
X 40 Y 0
DWELL 500
P100=0
LINEAR
INC
X 8 Y -40
DWELL 500
P100=1
LINEAR
INC
X 0 Y 80
X 40 Y -80
X 0 Y 80
DWELL 500
```

```
P100=0
LINEAR
INC
X 10 Y -80
DWELL 500
P100=1
LINEAR
INC
X 0 Y 80
X 15 Y 0
CALL 40
LINEAR
INC
TA 200
TM 2000
TS 200
X -15 Y 0
DWELL 500
P100=0
LINEAR
ABS
X 100 Y 200
CLOSE
```

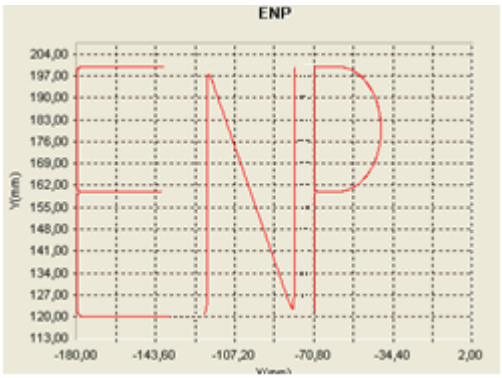


Figure 5.22 : *Ecriture d'une chaine de caractères*

Remarque :

Dans ce programme nous avons fait appel à un autre programme de mouvements qui est le suivant :

```
CLOSE
DELETE GATHER
UNDEFINE ALL
OPEN PROG 40 CLEAR
NORMAL K-1
CIRCLE1
ABS
X -60 Y 160 R 20
RETURN
```

- **Trajectoire spirale :**

Le programme :

```
CLOSE
DELETE GATHER
UNDEFINE ALL
I13=10
OPEN PROG 102 CLEAR
LINEAR
ABS
TA 600
TS 450
TM 3000
X -160 Y 220
DWELL 500
P100=1
ABS
INC (R)
NORMAL K-1
CIRCLE1
P32=0
WHILE (P32<80)
                  X (-P32-100) Y 250 I (P32)
                  P32=P32+20
ENDWHILE
DWELL 500
P100=0
LINEAR
ABS
X 100 Y 200
CLOSE
```

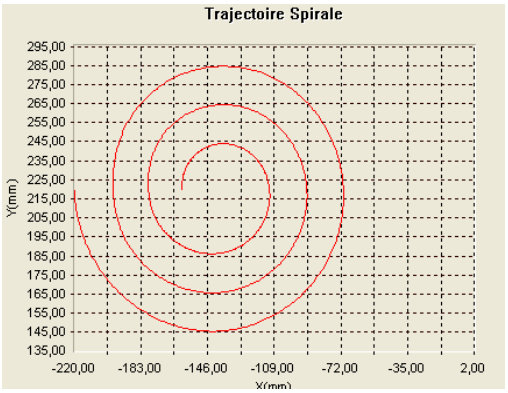


Figure 5.23 : *Trajectoire spirale*

- **Dessin du logo de l'Ecole Nationale Polytechnique :**

Le programme :

```
CLOSE
DELETE GATHER
UNDEFINE ALL
I13=10
OPEN PROG 38 CLEAR
LINEAR
INC
TA 200
TM 1000
TS 200
X 13.33 Y 40
NORMAL K-1
ABS
CIRCLE2
X -175.2 Y 240.2 R 42.72
X -222.5 Y 155.2 R 100
LINEAR
INC
TA 200
TM 700
TS 150
X 5 Y 0
NORMAL K-1
ABS
CIRCLE2
X -203.35 Y 169.34 R 20
NORMAL K-1
ABS
CIRCLE1
X -217.5 Y 160.2 R 20
LINEAR
INC
TS 200
TM 700
TS 150
X 3 Y 5
NORMAL K-1
ABS
CIRCLE2
X -200.35 Y 174.34 R 20
NORMAL K-1
ABS
CIRCLE1
X -214.5 Y 170.2 R 20
X -175.2 Y 225.2 R 80
X -140 Y 220 R 45
```

```
LINEAR
INC
TA 200
TM 1000
TS 200
X -22.46 Y -62.8
NORMAL K-1
ABS
CIRCLE2
X -117.26 Y 167 R 42.72
X -69.96 Y 252 R 100
NORMAL K-1
ABS
CIRCLE1
X -68.32 Y 242 R 6
NORMAL K-1
ABS
CIRCLE2
X -59.96 Y 249 R 20
X -69.96 Y 252 R 40
NORMAL K-1
ABS
CIRCLE1
X -117.26 Y 182 R 80
X -150.19 Y 194 R 45
CLOSE
```

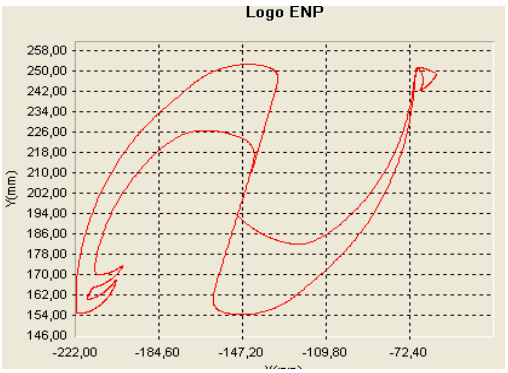


Figure 5.24 : *Logo de l'ENP*

V.6. Utilisation des PLC :

On utilise les PLC pour la gestion des programmes de mouvements. Dans notre application, nous avons d'abord élaboré un schéma synoptique décrivant le fonctionnement de notre PLC. Ensuite, nous avons traduit ce schéma en un PLC.

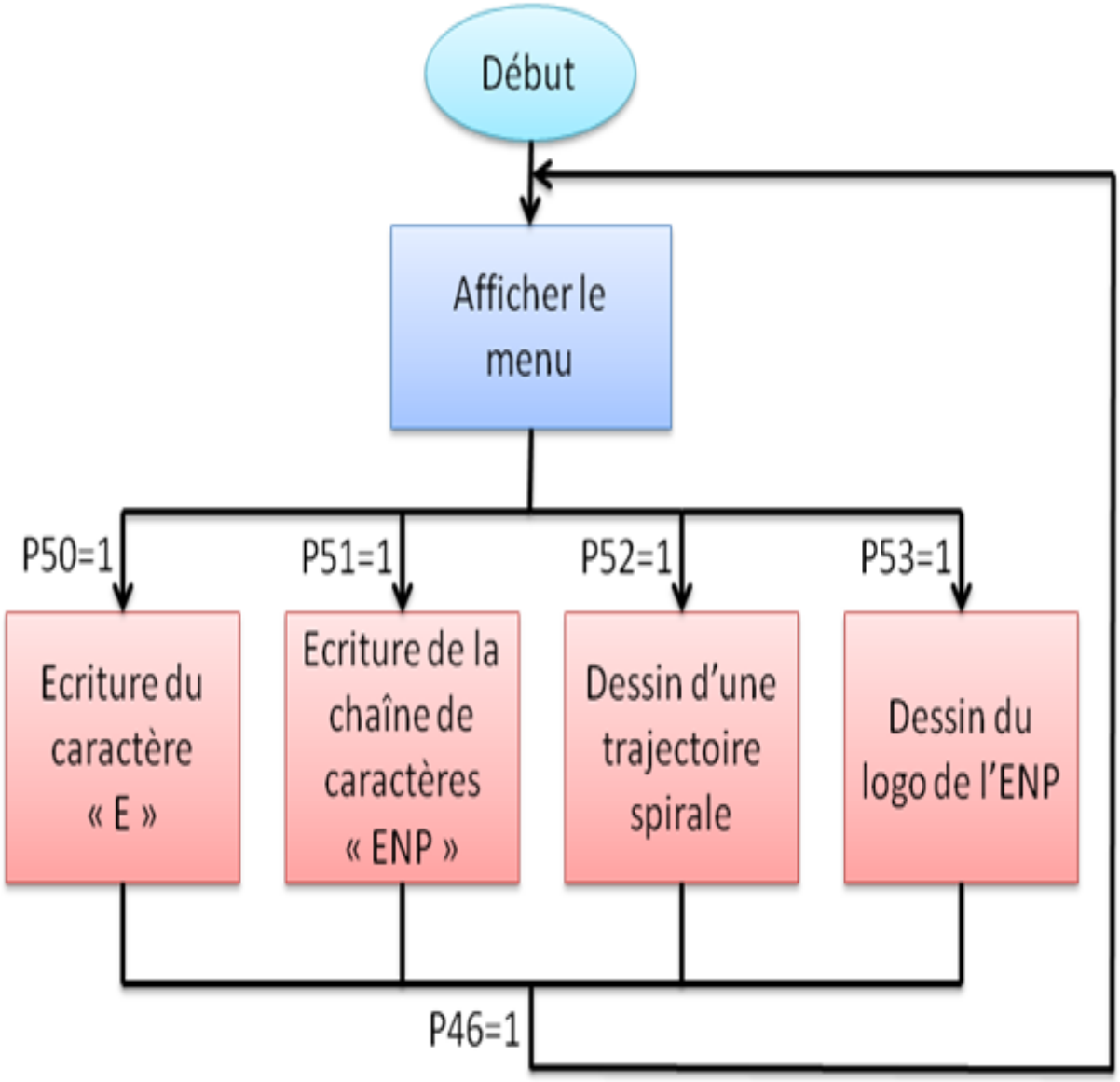


Figure 5.25 : *Schéma Synoptique du PLC*

Le programme (PLC) associé :

```
CLOSE
DELETE GATHER
OPEN PLC 6
CLEAR
IF (P40=0)
            CMD"#1J/"
            CMD"#2J/"
            I5150=1
            Q10=1
```

```
            P40=1
ENDIF
WHILE (P47=0 AND P46=0)
IF (P42=0)
       SENDS"SI VOUS VOULEZ ECRIRE (E) TAPEZ P50=1"
       SENDS"SI VOUS VOULEZ ECRIRE (ENP) TAPEZ P51=1"
       SENDS"SI VOUS VOULEZ EFFECTUER UNE TRAJECTOIRE SPIRALE TAPEZ P52=1"
       SENDS"SI VOUS VOULEZ DESSINER LE LOGO ENP TAPEZ P53=1"
       P42=1
ENDIF
WHILE (P48=0)
IF (P50=1)
       SENDS"TAPER P43=1 POUR COMMENCER L'ECRITURE"
       WHILE (P46=0)
                              IF (P43=1)
                                          CMD"&1B101R"
                                          P46=1
                                          P43=0
                              ENDIF
       ENDWHILE
       P50=0
       P48=1
       P49=1
ENDIF
IF (P51=1)
       SENDS"TAPER P43=1 POUR COMMENCER L'ECRITURE"
        WHILE (P46=0)
                              IF (P43=1)
                                          CMD"&1B100R"
                                          P46=1
                                          P43=0
                              ENDIF
       ENDWHILE
       P51=0
       P48=1
       P49=1
ENDIF

IF (P52=1)
       SENDS"TAPER P43=1 POUR COMMENCER L'ECRITURE"
       WHILE (P46=0)
                              IF (P43=1)
                                          CMD"&1B102R"
                                          P46=1
                                          P43=0
                              ENDIF
       ENDWHILE
       P52=0
       P48=1
       P49=1
ENDIF
IF (P53=1)
```

```
       SENDS"POSITIONNEMENT: A LA FIN DU POSITIONNEMENT TAPER P43=1 POUR
COMMENCER L'ECRITURE"
       CMD"&1B10R"
       WHILE (P46=0)
                              IF (P43=1)
                                          CMD"&1B38R"
                                          P46=1
                                          P43=0
                              ENDIF
       ENDWHILE
       P50=0
       P48=1
       P49=1
ENDIF
ENDWHILE
P47=1
ENDWHILE
IF (P49=1)
            SENDS"A LA FIN DU MOUVEMENT TAPER P46=0 POUR EXECUTER UN AUTRE
MOUVEMENT"

      SENDS"++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++"
            P42=0
            P47=0
            P48=0
            P49=0
ENDIF
CLOSE
```

Pour lancer l'exécution de ce PLC on tape « ENABLE PLC6 » dans le terminal et pour l'arrêter on tape « DISABLE PLC6 ».

CONCLUSIONS ET PERSPECTIVES:

Dans ce travail, nous nous sommes fixés comme objectif, la planification de trajectoires pour le robot SCARA en prenant en considération les singularités et les limites physiques des actionneurs utilisés. Nous avons commencé par la détermination des différents modèles de notre robot ainsi que la génération de mouvements dans les deux espaces articulaire et cartésien. Ceci est nécessaire pour pouvoir les implémenter par la suite dans l'interface UMAC.

Ensuite, une description matérielle et logicielle de cette interface a été faite. Dans cette partie, nous avons expliqué le fonctionnement général de l'UMAC ainsi que ses différents programmes. Sa programmation est possible grâce au logiciel PEWIN 32 PRO.

Les programmes de mouvements décrivent le mouvement à exécuter à l'aide d'une suite d'instructions. A leur exécution, ces instructions sont traduites en une série de positions de références que les moteurs doivent suivre avec une grande précision, afin d'obtenir le mouvement désiré. Pour cela, un régulateur PID+Feed Forward est implémenté dans l'interface.

Les programmes géométriques permettent d'implémenter les différents modèles du robot. Ils sont considérés comme des sous-programmes des programmes de mouvements, ils assurent le passage de coordonnées cartésiennes aux coordonnées articulaires et vis versa.

Les PLC constituent un outil très puissant pour la gestion des programmes de mouvements grâce à leur accès aux variables de travail de l'interface et aux entrées sorties.

Nous avons terminé par une application où on a apporté quelques modifications à la structure existante pour avoir un contrôle meilleur. Ensuite, nous avons exécuté plusieurs mouvements dans l'espace articulaire comme dans l'espace opérationnel. Nous avons aussi élaboré un PLC pour pouvoir gérer de manière ergonomique ces programmes de mouvements.

En perspectives, nous proposons d'adapter le robot à d'autres utilisations (Pick and Place par exemple) puisque l'UMAC nous offre la possibilité de planifier aisément n'importe quelle tâche. Nous proposons aussi d'utiliser le logiciel « Pmac HMI » qui permet des développer des interfaces ergonomiques afin de commander le robot d'une manière interactive.

ANNEXE A :

Calcul du modèle géométrique inverse :

En se basant sur le schéma du robot, une démarche analytique procédant par substitution, permet de déterminer le modèle géométrique inverse :

$$X^2 + Y^2 = {L_1}^2 + {L_2}^2 + 2L_1L_2\cos(\theta_2)$$

Ce qui donne :

$$\cos(\theta_2) = \frac{X^2 + Y^2 - {L_1}^2 - {L_2}^2}{2L_1L_2}$$

D'où :

$$\theta_2 = cos^{-1}(\frac{X^2 + Y^2 - {L_1}^2 - {L_2}^2}{2L_1L_2})$$

Nous allons calculer θ_1 :

$$\tan(\theta_1 + \theta_0) = \frac{Y}{X}$$

Alors :

$$\theta_1 + \theta_0 = \arctan(\frac{Y}{X})$$

Nous avons aussi :

$$\cos(\theta_0) = \frac{L_1 + L}{\sqrt{X^2 + Y^2}}$$

Tel que :

$$L = L_2\cos(\theta_2)$$

$$L = L_2\frac{X^2 + Y^2 - {L_1}^2 - {L_2}^2}{2L_1L_2}$$

En remplaçant L dans l'expression de $\cos(\theta_0)$, on aura :

$$\cos(\theta_0) = \frac{L_1 + L_2\frac{X^2 + Y^2 - {L_1}^2 - {L_2}^2}{2L_1L_2}}{\sqrt{X^2 + Y^2}}$$

Ce qui donne :

$$\theta_0 = cos^{-1}(\frac{X^2 + Y^2 + {L_1}^2 - {L_2}^2}{2L_1\sqrt{X^2 + Y^2}})$$

Calcul du modèle cinématique inverse :

Calcul de $\dot{\theta}_1$:

En dérivant les équations du modèle géométrique direct on obtient :

$$\dot{X} = -L_1\dot{\theta}_1 \sin(\theta_1) - L_2(\dot{\theta_1} + \dot{\theta}_2)\sin(\theta_1 + \theta_2)$$

$$\dot{Y} = L_1\dot{\theta}_1 \cos(\theta_1) + L_2(\dot{\theta_1} + \dot{\theta}_2)\cos(\theta_1 + \theta_2)$$

En multipliant les termes $\dot{X}$ (respectivement $\dot{Y}$) par $L_2 \cos(\theta_1 + \theta_2)$ (respectivement $L_2 \sin(\theta_1 + \theta_2)$), on obtient :

$$L_2 \cos(\theta_1 + \theta_2)\dot{X} = -L_1L_2\dot{\theta}_1 \sin(\theta_1)\cos(\theta_1 + \theta_2) - {L_2}^2(\dot{\theta_1} + \dot{\theta}_2)\sin(\theta_1 + \theta_2)\cos(\theta_1 + \theta_2)$$

$$L_2 \sin(\theta_1 + \theta_2)\dot{Y} = L_1L_2\dot{\theta}_1 \cos(\theta_1)\sin(\theta_1 + \theta_2) + {L_2}^2(\dot{\theta_1} + \dot{\theta}_2)\cos(\theta_1 + \theta_2)\sin(\theta_1 + \theta_2)$$

En faisant la somme des deux équations, on trouve :

$$L_2 \cos(\theta_1 + \theta_2)\,\dot{X} + L_2 \sin(\theta_1 + \theta_2)\,\dot{Y} = L_1L_2\dot{\theta}_1(\cos(\theta_1)\sin(\theta_1 + \theta_2) - \sin(\theta_1)\cos(\theta_1 + \theta_2))$$

On a :

$$\cos(\theta_1)\sin(\theta_1 + \theta_2) - \sin(\theta_1)\cos(\theta_1 + \theta_2) = \sin\big((\theta_1 + \theta_2) - \theta_1\big) = \sin(\theta_2)$$

D'où :

$$L_2 \cos(\theta_1 + \theta_2)\,\dot{X} + L_2 \sin(\theta_1 + \theta_2)\,\dot{Y} = L_1L_2\dot{\theta}_1\sin(\theta_2)$$

En final, on écrit :

$$\dot{\theta}_1 = \frac{L_2 \cos(\theta_1 + \theta_2)\,\dot{X} + L_2\sin(\theta_1 + \theta_2)\dot{Y}}{L_1L_2\sin(\theta_2)}$$

Calcul de $\dot{\theta}_2$ **:**

Dans ce qui précède (calcul du modèle géométrique inverse), nous avons l'expression suivante :

$$\cos(\theta_2) = \frac{X^2 + Y^2 - {L_1}^2 - {L_2}^2}{2L_1L_2}$$

En dérivant cette expression, on trouve :

$$-\dot{\theta}_2 \sin(\theta_2) = \frac{2\dot{X}X + 2\dot{Y}Y}{2L_1L_2}$$

Ce qui donne :

$$\dot{\theta}_\mathbf{2} = -\frac{\dot{X}X + \dot{Y}Y}{L_1L_2\sin(\theta_2)}$$

ANNEXE B :

Les variables I les plus utilisées :

Variable I	signification
I0-I99	variables d'installation globale de l'interface
Ixx00-Ixx99	configuration du moteur « xx » pour les applications désirées
Ixx00 = 1	pour activer le moteur « xx »
I7mn0-I7mn9	configuration du canal « n » de la DSPGATE « m »
I13	temps de segmentation des mouvements
I5n90	doit être initialisée à 1000 pour une échelle de temps en « ms » pour le système en coordination « n »
Ixx16	valeur limite de la vitesse pour le moteur « xx »
I5x11, I5x12	pour la temporisation
I5x90	définit l'unité de vitesse (mm/min ou deg/s)
I5n50 = 1	activer les calculs géométriques pour le système en coordination « n »
Ixx02	commande de sortie en courant pour le moteur « xx »
Ixx03	position du moteur « xx »
Ixx04	vitesse du moteur « xx »
Ixx05	adresse de la position principale pour le moteur « xx »
Ixx93	adresse de la source de la base de temps
Ixx30	gain proportionnel K_p du régulateur PID+Feed Forward
Ixx31	gain dérivé K_d du régulateur PID+Feed forward
Ixx33	gain intégral K_i du régulateur PID+Feed forward
Ixx34	I_M (mode d'intégration) $\begin{cases} = 0 : l'action\ intégrale\ activée\ tout\ le\ temps \\ = 1 : l'action\ intégrale\ activée\ quand\ la\ vitesse\ est\ nulle \end{cases}$
Ixx32	gain de vitesse feed forward K_{vff} du régulateur PID+Feed Forward
Ixx35	gain d'accélération feed forward K_{aff} du régulateur PID+Feed Forward
Ixx68	Gain Feed Forward de frottement. Activée quand l'erreur sur la position n'est pas dans les limites
Ixx69	DAC limite (valeur Maximum du courant avec lequel on peut attaquer l'amplificateur
Ixx11	limite fatale de l'erreur sur la position
Ixx12	valeur limite sur la position pour laquelle un drapeau (flag) d'avertissement est activé
Ixx13	valeur maximum positive de la position autorisée pour le moteur « xx »
Ixx14	valeur maximum négative de la position autorisée pour le moteur « xx »
Ixx15	taux de décélération du moteur « xx »

Tableau B.1 : *Les variables I usuelles*

ANNEXE C :

Table des codes d'erreurs :

Le code de l'erreur	Problème	Solution
ERRO 01	Commande non permise pendant l'exécution du programme	Arrêter le programme avant d'exécuter la commande
ERRO 02	Erreur de mot de passe	Entrer le mot de passe approprié
ERRO 03	Erreur de donnée ou commande non reconnue	Corriger la syntaxe de la commande
ERRO 04	Caractère illégal ou erreur sérielle de parité	Corriger le caractère ou vérifier le bruit sur le câble série
ERRO 05	Commande non permise à moins que le buffer soit ouvert	Ouvrir le buffer
ERRO 06	Aucun espace dans le buffer pour la commande	Laisser plus d'espace dans le buffer par la commande DELETE ou CLEAR
ERRO 07	Buffer déjà en service	Fermer le buffer ouvert par la commande CLOSE
ERRO 08	Erreur de lien	Le registre X :$0789 contient la valeur de l'erreur
ERRO 09	Erreur structurelle de programmation	Corriger la structure du programme
ERRO 10	Une des deux limites de dépassement pour un moteur est déclenchée	Corriger ou neutraliser les limites
ERRO 11	Le mouvement précédent est non accompli	Arrêter le mouvement (Abort)
ERRO 12	Un moteur dans le SC est en boucle ouverte	Fermer la boucle sur le moteur (#xxJ/)
ERRO 13	Un moteur dans le SC n'est pas activé	Mettre Ixx00 à 1
ERRO 14	Aucun moteur dans le SC	Définir au moins un moteur dans le SC
ERRO 15	Ne pas se pointer au bon programme	Employer la commande B d'abord
ERRO 16	Essayer de reprendre après ou avec des moteur hors la position d'arrêt	Employer J= pour remettre les moteurs en position d'arrêt

Tableau C.1 : *Codes d'erreurs*

ANNEXE D :

Détermination des paramètres de la boucle de régulation :

Pour la détermination des paramètres du régulateur PID, on suit les étapes suivantes :

1ére étape :

La première étape est d'utiliser *l'Auto Tuninig* de l'application Pmac Tuninig Pro pour avoir une première estimation des paramètres.

2ème étape :

Elle consiste à réajuster ces paramètres pour améliorer la réponse à un échelon en utilisant l'estimation interactive des paramètres. Cela est fait en visualisant la réponse à un échelon et en apportant les modifications nécessaires comme présenté dans le tableau suivant :

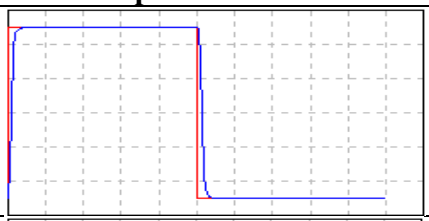
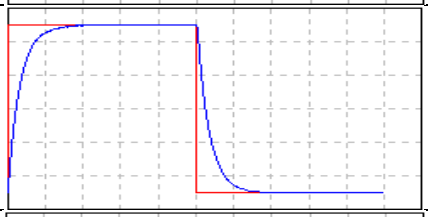
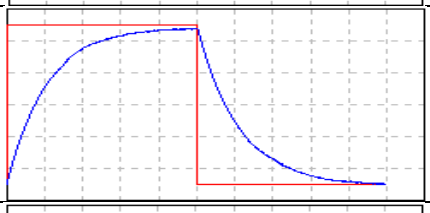
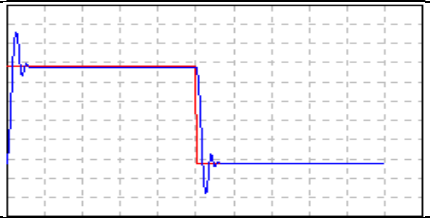
Réponse obtenue	Observation	Modification à apporter
	Cas idéal	/
	Réponse lente due au frottement	Augmentation du Ki et du Kp
	Réponse lente due à un gain proportionnel très faible	Augmentation du Kp ou diminution du Kd
	Dépassement et oscillations dus à un gain proportionnel très élevé	Diminution du Kp ou augmentation du Kd

Tableau D.1 : *Réglage des paramètres du régulateur PID à partir de la réponse indicielle*

3ème étape :

Une fois que les paramètres Kp, Kd et Ki sont ajustés, la troisième étape consiste à réajuster les paramètres Kvff et Kaff pour améliorer le mouvement parabolique (éliminer l'erreur en vitesse). Ceci est présenté dans le tableau suivant :

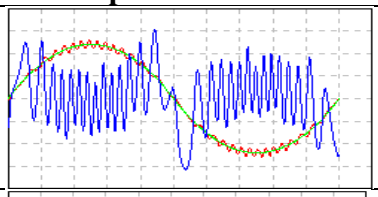
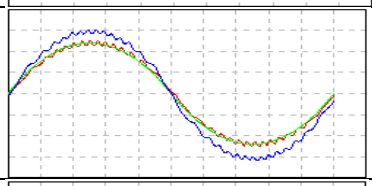
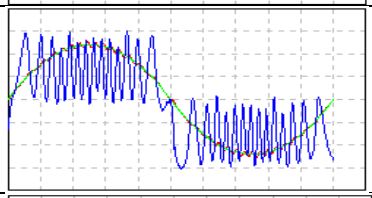
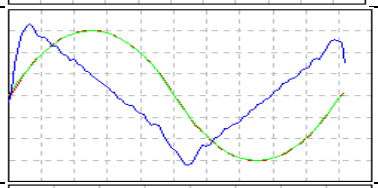
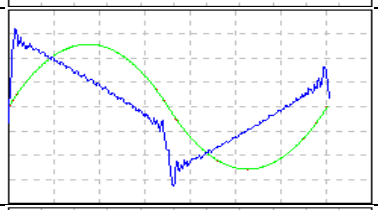
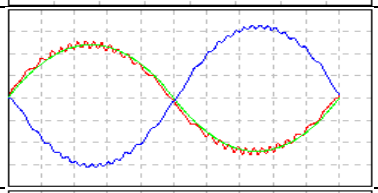
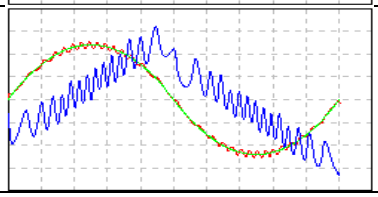
Réponse obtenue	Observation	Modification à apporter
	Cas idéal. L'erreur de poursuite est décorrélée.	/
	Corrélation vitesse/erreur élevée due à l'atténuation	Augmentation du gain Feed Forward en vitesse Kvff
	Corrélation vitesse/erreur élevée due au frottement	Augmentation du gain Feed Forward de limitation (Ix68)
	Corrélation accélération/erreur élevée	Augmentation du gain Feed Forward en accélération Kaff
	Corrélation accélération/erreur élevée due aux limitations physiques du système	Eviter les changements brusques de l'accélération
	Corrélation vitesse/erreur négative due à un gain de vitesse Feed Forward trop élevé	Diminution du gain Feed Forward en vitesse Kvff
	Corrélation accélération/erreur trop élevée due à gain Feed Forward en accélération trop élevé	Diminution du gain Feed Forward en accélération Kaff

Tableau D.2 : *Réglage de paramètres Feed Forward à partir de la corrélation (vitesse ou accélération/erreur)*

Avec :

— : Commande en vitesse

— : Réponse

— : Erreur

REFERENCES BIBLIOGRAPHIQUES :

[1] PIERRE TOURMASSOUD, « *Planification et contrôle en robotique* », Hermes, 1992.

[2] WISSAMA KHALIL et ETIENNE DOMBRE, « Modélisation et commande des robots », 1988.

[3] BERNARD BAYLE, « *Introduction à la robotique* », 2004-2005.

Disponible en format PDF sur internet.

[4] WISSAMA KHALIL et ETIENNE DOMBRE, « *Modélisation et commande des robots* », 2e édition, Hermes Science Publications, Paris, 1999.

[5] CHIKH LOTFI et ZIOUI NADJET, « *Conception, réalisation et commande d'un robot manipulateur* », Projet de fin d'études, ENP, 2006.

[6] HAMMAMI, « Cours DSP et famille C6000 », ENP, 2011.

[7] DELTA TAU DATA SYSTEMS, INC, « *Hardware Reference Manual* ».

Disponible en format PDF de ftp://ftp.deltatau.com/umac.pdf, Septembre 2009.

[8] DELTA TAU DATA SYSTEMS, INC, « *Turbo PMAC User Manual* ».

Disponible en format PDF de ftp://ftp.deltatau.com/pmac.pdf, Septembre 2008.

[9] CHRISTOPHE LE LANN, « *Le PID utilisé en régulation de position et/ou de vitesse des moteurs électriques* », 2006-2007.

Disponible en format PDF sur internet.

[10] DELTA TAU DATA SYSTEMS, INC, « *Software Reference Manual: Pmac Tuning Pro*».

Disponible en format PDF de ftp://ftp.deltatau.com/pmactuningpro.pdf, Janvier 2003.

[11] DELTA TAU DATA SYSTEMS, INC, « *UMAC Quick Reference* ».

Disponible en format PDF de ftp://ftp.deltatau.com/umacqref.pdf, Octobre 2003.

[12] DELTA TAU DATA SYSTEMS, INC, « *Software Reference Manual: Pewin32 Pro* ».

Disponible en format PDF de ftp://ftp.deltatau.com/pewin32pro.pdf, Janvier 2003.

ملخص:
العمل المقدم في هذه الأطروحة هو تخطيط المسار في الفضاء الديكارتي للآلي SCARA و هذا باستعمال واجهة التحكم UMAC ل DELTATAU . و لهذا الغرض تم وصف مختلف قوانين تخطيط المسار و النماذج الهندسية و الحركية للآلي و المحركات المستخدمة. ثم تم وصف الواجهة المستعملة و المصحح المرافق لها. قدم أيضا البرنامج المستعمل PEWIN32 Pro و كذلك التطبيقات المرافقة له. و في الأخير تم استعمال الآلي كمخطط.
كلمات مفتاحية:
الآلي SCARA ، تمثيل الذراعات الآلية ، محرك التيار المستمر ، تخطيط المسارات ، DELTATAU ، واجهة التحكم ، UMAC ، المصحح PID ، PEWIN32 PRO.

Résumé :

Le travail présenté dans ce mémoire est la génération de trajectoires dans l'espace cartésien pour le robot SCARA à l'aide de l'interface UMAC de DELTA TAU. Dans cette fin, différentes lois de génération de trajectoires ainsi que les modèles géométriques et cinématique du robot seront présentés ainsi que les actionneurs utilisés. Puis, une description de l'interface utilisée et de l'algorithme de régulation implémenté sera faite. Ensuite, le logiciel de travail (PEWIN32 PRO) ainsi que les différentes applications qui l'accompagnent seront présentés. Pour valider le travail, le robot sera utilisé comme traceur.

Mots clés : Robot SCARA, modélisation des bras manipulateurs, moteur à courant continu, génération de trajectoires, DELTA TAU, interface de commande UMAC, régulateurs PID, PEWIN32 PRO.

Abstract :

The following work presents the generation of trajectories in Cartesian space for the SCARA robot using the interface UMAC of Delta Tau. To achieve this goal, a description of different laws of trajectory generation, geometric and kinematic models of the robot and the used actuators will be done. Then, a description of the interface and the implemented control algorithm will be exposed. After that, the software PEWIN32 PRO and its various applications will be presented. Finally, in order to validate the previous work, the robot will be used as a tracer.

Keywords: SCARA robot, modeling of robot arms, DC motor, trajectory generation, DELTA TAU, UMAC control interface, PID, PEWIN32 PRO.