REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE




ECOLE NATIONALE POLYTECHNIQUE

Département de Génie Electrique

# Mémoire de fin d'études

En vue de l'obtention du diplôme d'ingénieur d'Etat en Automatique

# ETUDE ET MISE EN FONCTION DE L'INTERFACE D'AXES INDUSTRIELS TURBO UMAC

Proposé et dirigé par :
*M^r O.STIHI*
*M^r M.TADJINE*

Etudié par :
*M^r BADJI Lyes*
*M^r BOUZIANI Fethi*

*Promotion : Juin 2002*

***E.N.P.10, Avenue Hassen-Badi, EL-HARRACH, ALGER***

# *REMERCIEMENTS*

*Les travaux présentés dans ce mémoire ont été menés au laboratoire de commande des processus à l'école Polytechnique d'Alger.*

*Nous remercions nos promoteurs, Monsieurs : O. Stihi et M.Tadjine, pour leurs conseils et efforts durant toute l'année.*

*Nous remercions aussi Monsieur H.Chekireb pour tous ses précieux conseil et indications qui nous ont été d'une très grande utilité.*

*Nos remerciements vont aujourd'hui a ceux qui ont participé à notre formation du primaire à l'universitaire.*

*Lyes BADJI & Fethi BOUZIANI*

# *DEDICACES*



*Je dédie ce travail avant tout, à ma mère ; qui m'a tout donné et à qui je doit tout, et j'espère que j'aurais un jour l'occasion de la remercier d'avantage.*

*Je dédie aussi ce travail, à mon père ; pour sa disponibilité et son soutien morale depuis mon enfance. Père merci.*

*Je n'oublie pas mon frère sofiane pour les moyens qu'il m'a fournis, et que sans sa contribution, ce travail n'aurait pas cette finition.*

*A tout les membres de ma famille ; mes sœurs Malya et Iméne, mes frères Fahim, et Boualem qui mon apporté leur soutien durant mes études.*

*A ma tante Fifi et ses fils et filles surtout ma cousine Fouzia qui mon beaucoup aidé et soutenue pour réaliser ce travail.*

*Aussi à ma tante Ratiba et mon cousin Hakim.*

*A mon ami et binôme Fethi Bouziani. Merci.*

*Lyes BADJI*

# DEDICACE



*Je dédie ce projet*

*A La mémoire de mon cher père qui ma encouragé a choisir cette spécialité.*

*A Ma très chère mère qui ma aidé et soutenu pendant tout mon projet .*

*A Mes professeurs qui grâce a leur aide j'ai pu élaborer ce projet*

*A Mes chers frères et sœurs .*

*A Mes tantes et mes oncles, à toute ma famille .*

*A tous mes amis(e).*

*A mon ami et binôme Iyes Badji.*

*Fethi BOUZIANI*

# *Sommaire*

**CHAPITRE II. Description du langage de programmation de l'UMAC**

**CHAPITRE III. Fonctionnement de L'UMAC**

# *Liste des figures*

## *CHAPITRE I*



## *CHAPITRE II*

## *CHAPITRE III*



## *CHAPITRE IV*

# INTRODUCTION

## INTRODUCTION GENERALE :



La commande de mouvement est d'une très grande importance dans le domaine industriel. Elle a fait l'objet de nombreux travaux de recherche, basés essentiellement sur les stratégies de commande et de leur implémentation.

L'évolution des technologies informatiques en général et numériques en particulier a permis l'amélioration de la commande des moteurs grâce à l'exploitation de leur grande vitesse de traitement de l'information et la possibilité d'appliquer les différentes techniques qu'on pouvait pas utiliser avec les anciens circuits de commande. Cela à dynamiser l'industrie de l'électronique, de nouveaux matériels spécialisés dans la commande sont alors apparus, plusieurs firmes entre autre **DELTA TAU** [1] proposent une gamme variée de cartes de commande parmi elles : l'interface Turbo UMAC objet de notre étude.

Cette interface est autonome et programmable à partir d'un PC par une liaison série, elle contrôle jusqu'à huit moteurs simultanément et peut être étendue jusqu'à 32 moteurs, elle fournie une grande flexibilité et un large champ d'utilisation technologique.

En vue d'exploiter les facultés de l'interface UMAC, notre travail est porté sur «l'étude et la mise en fonction de l'interface d'axes industriels Turbo UMAC ».

Ce mémoire est structuré de la manière suivante :

- Le premier chapitre portera sur l'étude de l'architecture matérielle et les différentes fonctionnalités des cartes composant l'interface UMAC.

- Nous aborderons dans le second chapitre l'aspect logiciel, où on présentera la structure des programmes et les variables de configurations ainsi que les différants modes et trajectoires possibles.

- Le troisième chapitre traite la chaîne d'acquisition des données, la présentation des algorithmes de commande implémentés ainsi que les cycles des taches exécutées en permanence dans l'interface.

- Dans le dernier chapitre on donne la configuration et la programmation nécessaire pour la mise en fonction de l'interface en vue de commander les actionneurs d'un bras de robot à deux degrés de libertés.

- Et enfin en conclusion on trouvera un résumé du travail fait, les résultats obtenus ainsi qu'une perspective à notre travail.

# CHAPITRE I : Description matériel de l'UMAC

## I.1 PRESENTATION GENERALE :

L'UMAC « *Universal Motion and Automation Controller* » est un contrôleur universel de mouvement et d'automatisation, constitué d'un système modulaire en format 3U, combinant la puissance et la flexibilité de la famille des contrôleurs de mouvement multi-axes programmables PMAC « *Programmable Multi-Axis Controller* » produit par la firme **DELTA TAU**. l'unité centrale de traitement du système UMAC est la carte Turbo PMAC2-3U CPU, relié aux cartes d'entrée/sortie DSPGATEs et IOGATE, d'amplification AMP-2 et d'alimentation électrique par la surface arrière UBUS (UMAC Bus) implantée dans une armoire munie de cartes : ces cartes sont dans un ordre optimal du bas en haut afin que le danger et le bruit électromagnétique soient éloignés aux maximum, la Figure I.1.a donne l'image l'UMAC et la Figure I.1.b présente la configuration de piles en format 3U de ce dernier.

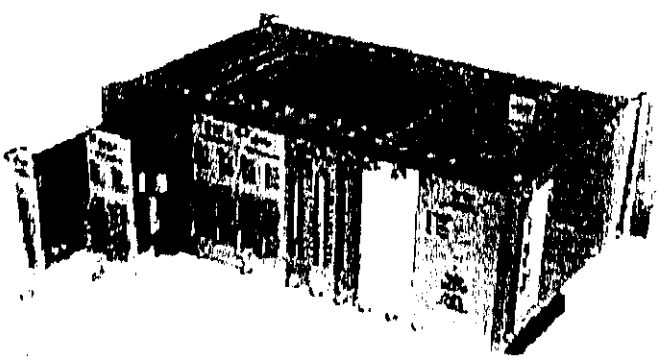


**Figure I.1.a** : Image du Turbo UMAC

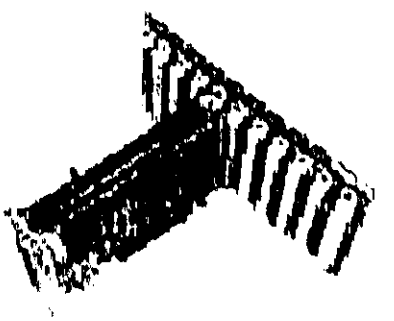


**Figure I.1.b :** Configuration de piles en format 3U Sur un UBUS

Les différent cartes composant l'UMAC sont :

**I.1.1 Turbo PMAC2-3U CPU [9] :** C'est l'unité centrale du traitement du système, elle joue le rôle d'un gestionnaire du système et d'un contrôleur pour les moteurs, Elle contient un DSP Motorola 56303, des mémoires où sont implémentés les micro-programmes et un connecteur série RS 232/422.

**I.1.2 DSPGATE [10] :** Nous disposons de deux DSPGATE qui jouent le rôle d'une première interface entre le CPU (Turbo PMAC2-3U CPU) et les signaux d'entrée/sortie pour les moteurs, chacune d'elles peut contrôler quatre axes et chaque axe envoie les signaux de sortie de commande pour les amplificateur et les signaux reçoit d'entrée encodeur et indicateurs.

**I.1.4 IOGATE [11] :** C'est un composant d'adaptation des signaux d'entrée-sortie digital avec isolation optique, elle est faite pour compléter les fonctions d'automatisation de l'UMAC.

**I.1.5 AMPLIFICATEUR DE PUISSANCE [13] :** Il y'a huit amplificateurs répartis sur deux cartes, chaque amplificateur a pour rôle de rendre les signaux de commande délivrée par les DSPGATE amplifié et exploitable.

**I.1.6 CIRCUIT D'ALIMENTATION ELECTRIQUE [14] :** Assurant une alimentation de 5V, 15V et -15V continue nécessaire pour l'alimentation des circuits analogiques et numériques du système.

## I.2 FONCTIONNEMENT GENERAL :

Le programme de mouvement une fois élaboré sur un PC sera chargé dans une zone mémoire située dans le Turbo PMAC2-3U CPU grâce à un logiciel associé à l'interface et par l'intermédiaire d'une liaison série RS232/422 (Figure I.2).

A l'exécution des instructions de commande de mouvement qui sont dans le programme, le CPU communique avec les registres de la DSPGATE par l'intermédiaire de l'UBUS en envoyant et en lissent les signaux de commande et de mesure de l'encodeur et d'indicateurs.

A ce moment la DSPGATE envoi les signaux de commande analogique en courant (après la conversion digitale/analogique) à l'amplificateur et compte les impulsions venant de l'encodeur (conditionnement des signaux de l'encodeur) tout en mettent l'état des d'indicateurs (par exemple l'état des fins de course) dans des registres bien spécifiques.

L'amplificateur pour sa part génère un courant proportionnel à la tension de commande livrée par la DSPGATE en appliquent une tension avec modulation de largeur des impulsions (PWM) sur les bornes du moteur.

Si des signaux entrées ou sortie digitaux sont utilises pour une application donner, alors ces signaux vont être achemines par l'IOGATE qui va les conditionner et le sauvegarder dans de registre pour que le CPU puisse lire ou écrire les données.

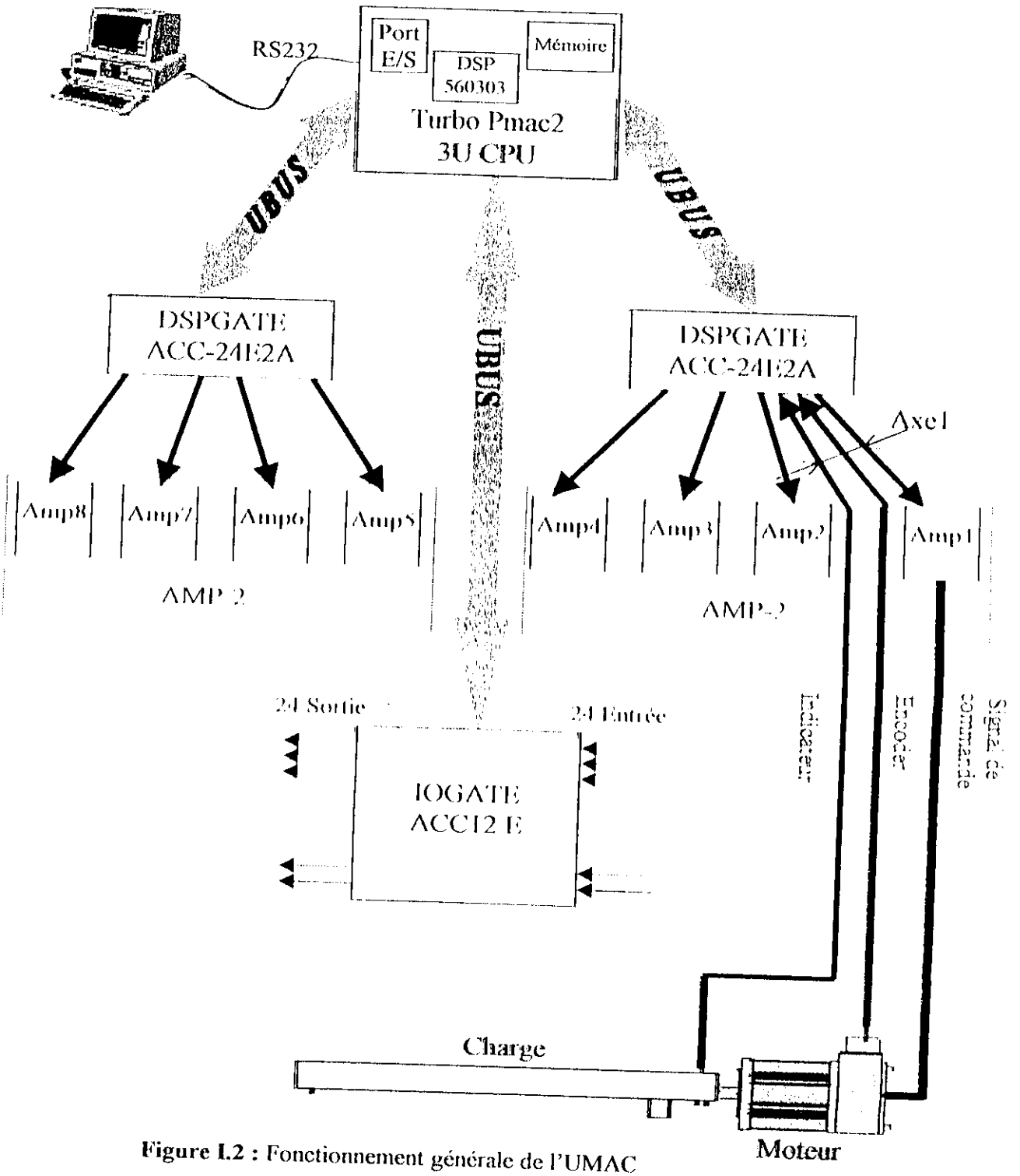



**Figure I.2 :** Fonctionnement générale de l'UMAC

## I.3 DESCRIPTION DU TURBO PMAC2 3U CPU [2], [4], [5], [9] :

PMAC, prononcé " *Pe'-MAC* ", représente un contrôleur multi-axes programmable, c'est une famille des contrôleurs a rendement élevé, capables de commander jusqu'à huit moteurs (huit axes) simultanément et extensible jusqu a 32 axes, pour cette famille de contrôleur, le Turbo Pmac2 3U CPU est considéré parmi les plus récents avec un niveau très sophistiqué, l'image du Turbo Pmac2 3U CPU est présenté dans la Figure I.3.

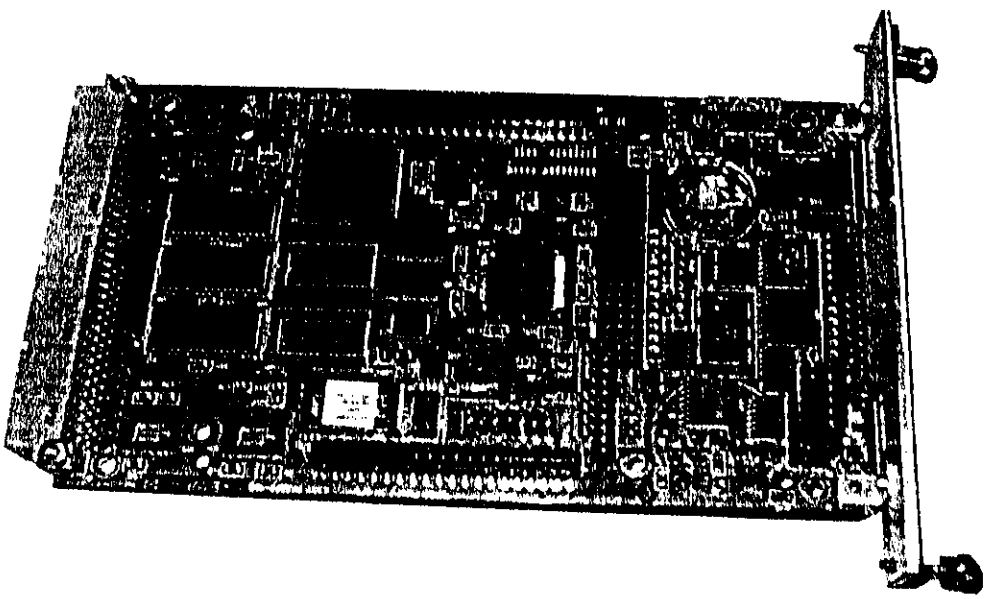


**Figure I.3** : Turbo Pmac2 3U CPU

Le processeur de traitement signal numérique **DSP56303** de **Motorola** est l'élément principal du Turbo Pmac2 3U CPU, car c'est l'unité centrale de cette carte ou tous les algorithmes de commande seront calculés pour les huit axes.

Cette carte a sa propre mémoire et microprocesseur, par conséquent elle fonctionne comme un contrôleur autonome ou un ordinateur qu'ont peut commander par l'intermédiaire d'un port série. Le port série est connecté directement à cette carte, le diagramme en block du Turbo Pmac2 3U CPU est présenté par la Figure I.4 suivant :

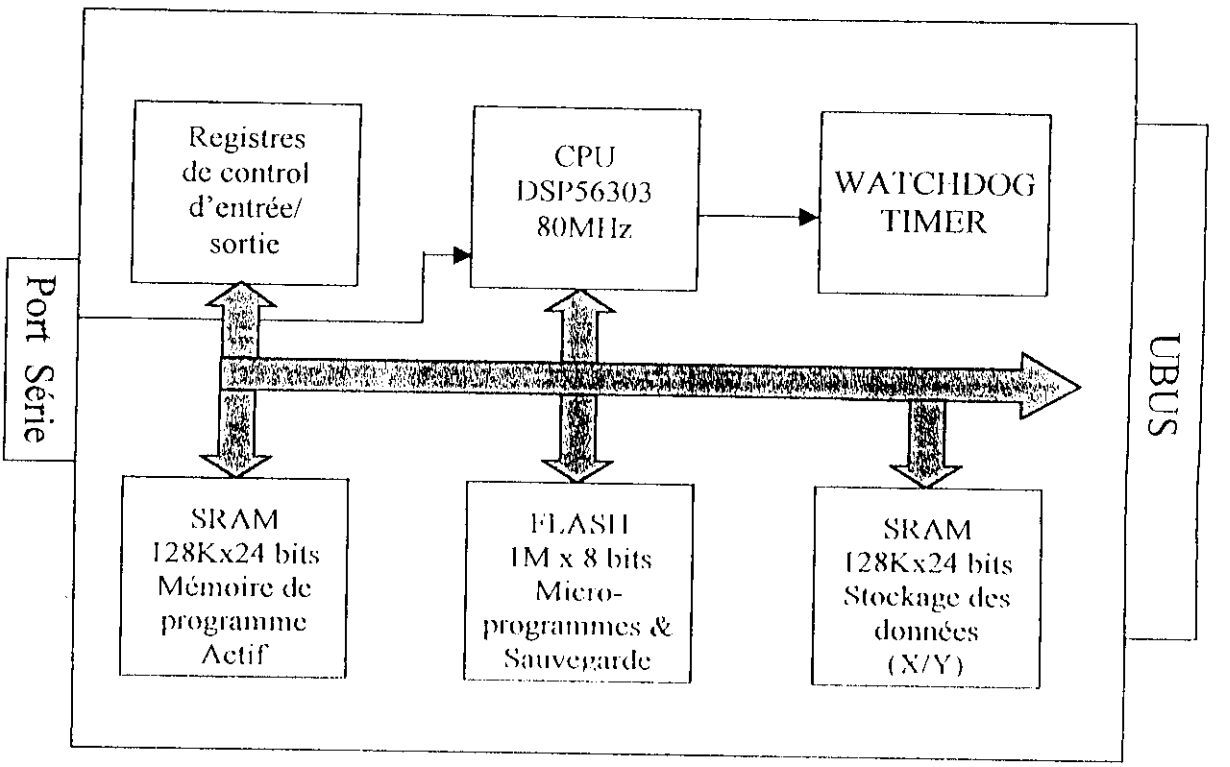



**Figure I.4** : Diagramme en block du Turbo Pmac2 3U CPU

D'après ce schéma on distingue les principaux élément suivant :

Un unité centrale de traitement Motorola DSP56303 80MHz 24 bits.

- Mémoire flash (1M x 8 bits) pour la sauvegarde d'utilisateur et les micro-programmes avec version 1.936.
- Mémoires SRAM (128k X 24 bits) actif pour les programmes, la compilation et l'assemblage.
- Mémoires SRAM (128k X 24 bits) de données d'utilisateur.
- Connecteur avec interface série RS-232/422.
- Whatchdoge timer.
- Registres de control.

### I.3.1 DESCRIPTION DU DSP56303 [15], [16], [17], [18], [19],[20] :

Le DSP56303 est un processeur de la famille DSP56300 spécialisé en traitement numérique du signal, commercialisé assez récemment (à partir de 1996). Cette famille utilise un rendement élevé, avec un cycle d'exécution d'instruction augmenté doublement par rapport à la famille des DSP56000 tout en ayant le même code instruction. Les principales caractéristiques du DSP56002 sont les suivantes :

- 80 millions instructions par seconde (MIPS) à une fréquence d'horloge de 80 MHz.
- Architecture Harvard parallèle permettant l'exécution d'une instruction en parallèle avec l'accès mémoire.
- Technologie CMOS très faible consommation.
- Code instruction compatible avec le noyau DSP56000.
- Unité arithmétique et logique de données (ALU) avec un multiplieur-accumulateur parallèle de 24 x 24 bits fonctionnant en un cycle machine, et support arithmétique de 24-bits ou 16 bits sous la commande de logiciel.
- Unité de commande de programme (PCU), mode d'adressage optimalisé pour les applications de DSP, contrôleur d'instruction cache intégré.

#### I.3.1.1 Description de l'architecture interne du DSP56303 :

Les processeurs de la famille 56K sont constitués d'un module de base identique et de ressources propres à chaque DSP. Le DSP56303 se distingue des autres processeurs par des périphériques et des zones de mémoire organisées selon l'architecture présentée sur la Figure I.5. Sur cette Figure on distingue deux parties, l'une constituant le module de base de la famille DSP56K et l'autre, en gris, présentant les fonctions complémentaires propres au DSP56303.

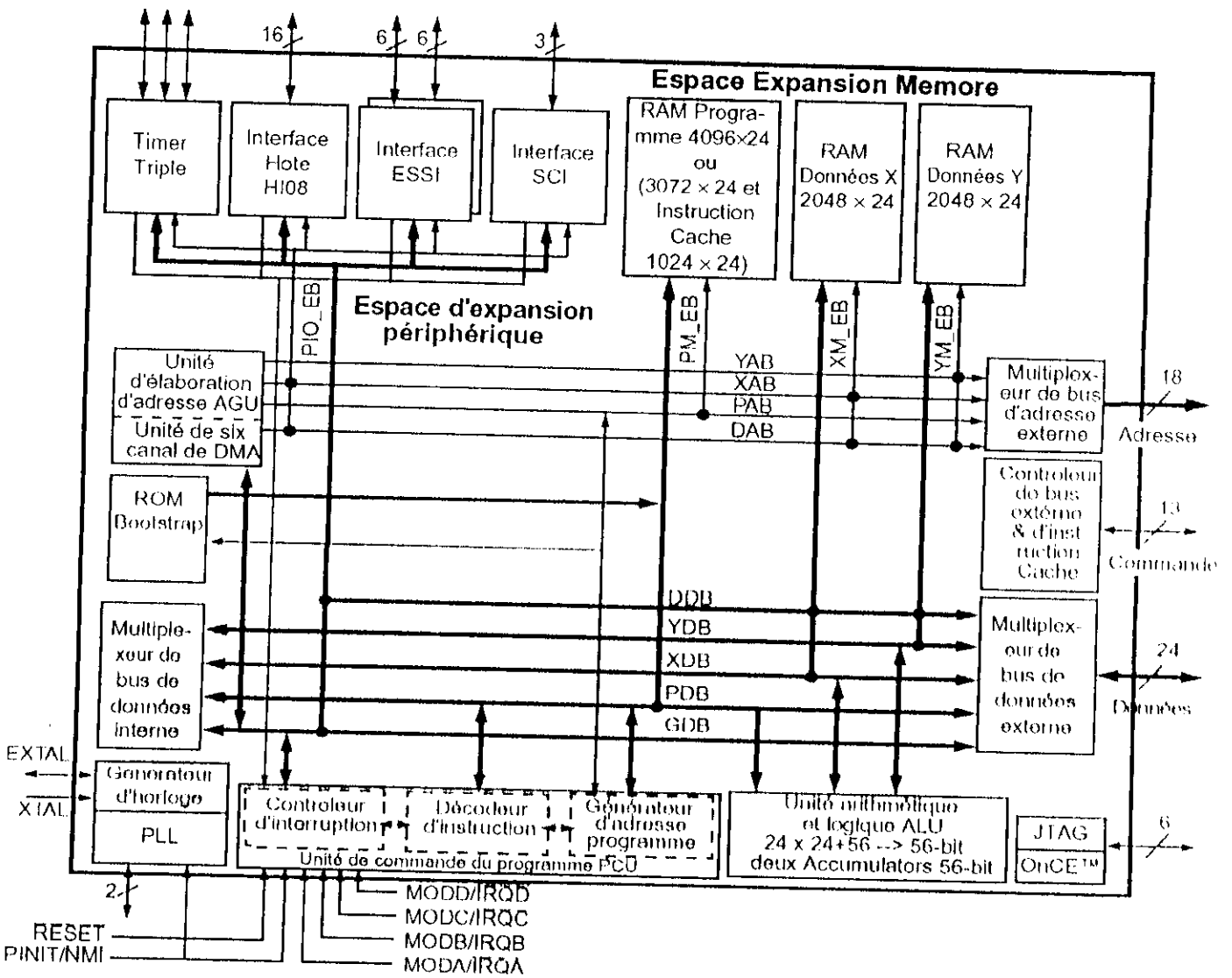



**Figure I.5** : Architecture interne du DSP56303

Le processeur DSP56303 est constitué par :

- Une unité arithmétique et logique (ALU).
- Une unité d'élaboration d'adresse (AGU).
- Une unité de commande du programme (PCU).
- Contrôleur DMA Concourant de six-canaux.
- Un émulateur intégré (OnCE).
- Un multiplieur de fréquence à asservissement de phase (PLL) associé à l'horloge.
- Deux interfaces séries Synchrones (ESSI).
- Interface séries asynchrones (SCI) avec le générateur d'horloge de transmission.

• Une Interface parallèle hôte de 8 bits (HI08), supporte une variété de bus (par exemple ISA).

• Module de temporisateur (Timer) triple.

• Cinq bus de données interne 24 bits (XDB données X, YDB données Y, PDB programmes, GDB expansion périphérique et DDB accès direct à la mémoire).

• Quatre bus d'adresse interne 24 bits (XAB données X, YAB données Y, PAB programmes et DAB pour les données de DMA).

• Un multiplexeur de bus de données associé à une unité de manipulation de bits.

• Trois zones mémoire externe (XAB données X, YAB données Y et PAB programmes) avec un bus de données de 24 bits et un bus d'adresse de 18 bits.

• RAM interne, avec taille programmable et un maximum de :

  • 4096 x 24 bits RAM Programme (mémoire cache d'instruction).

  • 2048 x 24 bits RAM de données X.

  • 2048 x 24 bits RAM de données Y

• ROM bootstap (programme de démarrage) de 192 x 24 bits.

• Extension mémoire :

  • Port extension mémoire externe.

  • Extension mémoire de données de deux 256 x 24 K bit espaces mémoire.

  • Extension mémoire Programme de deux 256 x 24 K bit espaces mémoire.

  • Logique de sélection pour se connecter à des SRAMs et à SSRAMs n'exigent aucun circuit supplémentaire.

• Le contrôleur de DRAM n'exige aucun circuit supplémentaire pour connecter DRAMs.

• Jusqu'à 34 broches E/S programmables de l'usage universel de (GPIO), qui dépend de l'activation des périphériques.

L'ensemble des éléments cités précédemment constitue le processeur spécialisé DSP56303. Ce dernier dispose donc de toutes les fonctions nécessaires pour développer des applications numériques complexes comme le contrôle des moteurs. Il faudra néanmoins le compléter par des convertisseurs numérique/analogique et des compteurs, de façon a mesurer et à restituer un signal analogique après les traitements appliqués par le DSP56303.

Les différentes unités, interface et mémoires qui constituent le DSP56303 communique entre elles par les cinq bus de données et les quatre bus d'adresses tous indépendamment. Cette

architecture à bus multiples est de type Harvard parallèle où les instructions constituant le programme sont séparées des données, elles même sont séparées dans deux champs de mémoire indépendant. Ainsi dans cette architecture interne, chaque zone de mémoire P, X et Y possède son propre bus d'adresse (..AB) et de données (..DB), ce qui rend possible l'accès simultané ou en parallèle à trois informations (une instruction et deux données) contenue dans les trois champs de mémoire. Ce parallélisme est aussi rendu possible grâce aux trois unités de calcul indépendantes que sont l'ALU (unité arithmétique et logique), l'AGU (unité d'élaboration d'adresse) et le PCU (unité de commande du programme) et l'utilisation de registre tampons. Ces derniers jouent un rôle important dans le fonctionnement du processeur car ils permettent de libérer les bus internes qui pourront alors être utilises pour des accès en mémoire.

Par contre, les différentes bus internes ne sont pas disponibles dans la configuration parallèle à l'extérieur du DSP. Ils sont multiplexés afin de réduire le nombre de liaisons externes qui constituent le port A, et pour distinguer parmi les bus internes celui connecté en sortie par le multiplexeur, un bus de commande fournit les signaux électriques nécessaires à son identification. On pourra donc, à l'extérieur du DSP56303, câbler d'avantage de mémoire P, X et Y pour étendre la mémoire disponible.

Sur la Figure I.5, on remarque que le DSP56303 est équipé de trois systèmes de communication de nature différente ce qui en fait un processeur largement ouvert vers l'extérieur. En effet, il possède trois interfaces de communication série SCI et deux ESSI et une interface Hôte HI08. L'interface SCI est utilise pour la communication série avec l'ordinateur principal, car elle permet une transmission rapide avec ce dernier.

### I.3.1.2 Description des broches du DSP56303 :

La description des broches qui est présentée sur la Figure I.6 [15] comprend le type de liaison (entrée, sortie ou bidirectionnelle) et ses états électriques possibles (trois états, bidirectionnelle) notamment à l'initialisation du DSP.

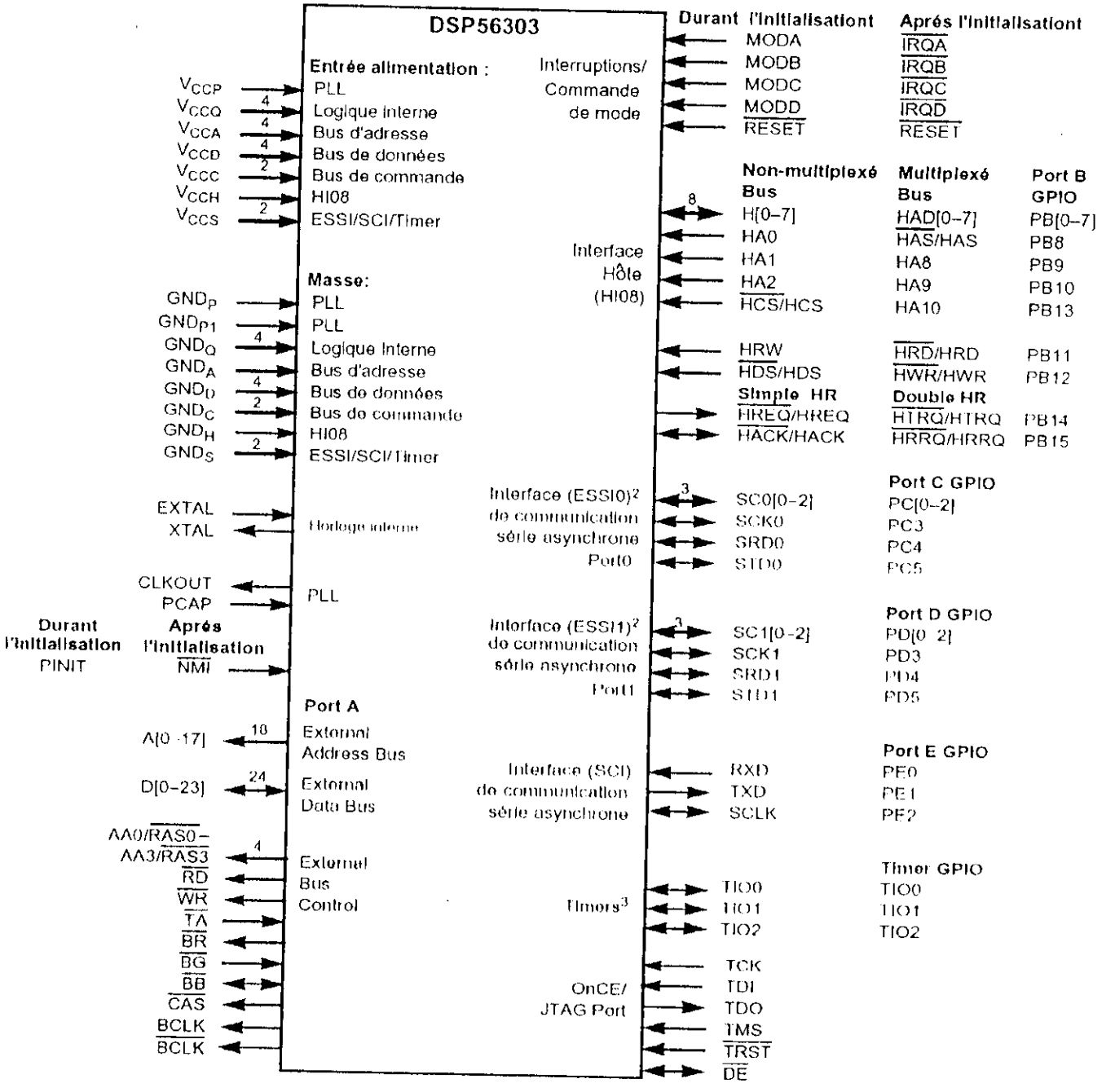



**Figure I.6 :** Désignation des broches du DSP56303

### I.3.1.2.1 les broches d'alimentation du DSP56303 :

Le DSP56002 possède de nombreuses broches d'alimentation associées à ses différentes unités intégrées. La séparation de ces broches d'alimentation permet de réaliser un

câblage adéquat afin de minimiser les risques d'interférence électrique entre les unités de calcul.

**a)** $V_{CCA}$ **et** $GND_A$ : +5 et 0 V d'alimentation des circuits PPL.
**b)** $V_{CCQ}$ **et** $GND_Q$ : +5 et 0 V d'alimentation des circuits logiques internes.
**c)** $V_{CCA}$ **et** $GND_A$ : +5 et 0 V d'alimentation des circuits de bus d'adresses.
**d)** $V_{CCD}$ **et** $GND_D$ : +5 et 0 V d'alimentation des circuits de bus de données.
**e)** $V_{CCC}$ **et** $GND_C$ : +5 et 0 V d'alimentation des circuits de bus de commande
**f)** $V_{CCH}$ **et** $GND_H$ : +5 et 0 V d'alimentation des circuits de l'interface hôte.
**g)** $V_{CCS}$ **et** $GND_S$ : +5 et 0 V d'alimentation des circuits des port série et Timers.

#### I.3.1.2.2 le circuit d'asservissement de phase et l'horloge interne :

Les cinq broches présentées ci-après sont associées au circuit d'horloge du DSP qui comprend également un multiplieur de fréquence par asservissement de phase (PLL).

**a) EXTAL** : *External Clock Crystal Input*. Cette broche permet la connexion d'un quartz externe à un oscillateur intégré ou d'injecte un signal d'horloge provenant d'un oscillateur externe.

**b) XTAL :** *Crystal Input*. La sortie de l'oscillateur intégré est disponible sur cette broche afin d'y connecter un quartz externe.

**c) CLKOUT** : *Clock Output*. Lorsque le circuit PLL est actif et asservi, sa sortie délivre sur CLKOUT un signal d'horloge de rapport cyclique ½, parfaitement synchronisé sur l'horloge interne du DSP. Si le circuit PLL n'est utilisé, cette broche fournit un signal de même fréquence et de même rapport cyclique que le signal XTAL.

**e) PCAP** : *PLL FilterCapacitor*. Le filtre passe- bas du circuit PLL nécessite un condensateur externe dont la valeur fixe ses caractéristiques fréquentielles et par là même celles du circuit à asservissement de phase PLL.

**f) PINIT** : *PLL Initial*. L'état logique de l'entrée PINIT est échantillonné au front montant du signal $\overline{\text{RESET}}$, est copié dans le bit PEN du registre de commande du PLL. Il permet d'activer ou non le circuit PLL.

**d)** $\overline{\textbf{NMI}}$ **:** *Nonmaskable Interrupt*. Après la phase d'initialisation, l'état de l'entrée PINIT est ignoré, autrement dit cette entrée sera considérée comme une entrée d'interruption $\overline{\text{NMI}}$ .

**I.3.1.2.3 le port A :**

Il regroupe les bus d'adresses, de données et de commande dont toutes les broches sont dans l'état impédance pendant la phase d'initialisation du processeur.

**a) bus d'adresse externe (A[0-17 ]) :** ces 18 connections dont $A_0$ est le bit de poids faible (LSB) et $A_{17}$ celui de poids fort (MSB), permettent d'accéder à 256 K cellules de mémoire des champs X, Y et P.

**b) bus de données ( D[0-23] ):** Il est composé de 24 connections dont $D_0$ est le bit de poids faible (LSB) et $D_{23}$ le bit de poids fort (MSB) ou bit de signe. Il permet le transfert de données entre les champs de mémoire externe et les unités de calcul internes du DSP.

**c) bus de commande :** Les signaux générés ou reçus par le bus de commande assurent le transfert de données entre le DSP et les calculateurs externes, des périphériques et des champs de mémoire externe.

• **AA [0–3] :** *Address Attribute.* Une fois définies comme AA, ces signaux peuvent être utilisés comme chip selects (CS ) ou les lignes supplémentaires d'adresse, comme c'est le cas pour cette application où AA[1-3] sont utilises pour la sélection des champs de mémoire X, Y et P externe, et $AA_0$ est ajouté au bus d'adresse externe pour une totalité de 19 bits d'adresse.

• **RAS[0–3] :** *Row Address Strobe.* Une fois définis comme RAS, ces signaux peuvent être utilisés comme une entrée RAS pour connecter des mémoires vives dynamiques DRAM avec le DSP. Ces signaux sont des sortis en trois états avec polarité programmable.

• $\overline{\text{RD}}$ : *Read Enable.* lecture active. Cette sortie a trois états, elle permet de lire une donnée de la mémoire externe par l'intermédiaire du bus de données de $D_0$ à $D_{23}$.

• $\overline{\text{WR}}$: *Write Enable.* Cette sortie a trois états, elle permet d'écrire dans la mémoire externe une donnée disponible sur le bus de données de $D_0$ à $D_{23}$.

• $\overline{\text{TA}}$ : *Transfer Acknowledge.* Cette entrée est active pour mettre les lignes $\overline{\text{RD}}$ et $\overline{\text{WR}}$ en attente (la reconnaissance de transfert de données).

• $\overline{\text{BR}}$ : *Bus Request.* Cette entrée permet à un système externe de devenir maître des bus d'adresses et de données externes et du bus de commande. Dans ce cas les lignes décrites précédemment sont placées en haute impédance.

• $\overline{\text{BG}}$: *Bus Grant.* Cette sortie permet de signaler au système externe que sa demande de libération du bus externe par le DSP est accordée.

• **$\overline{BB}$** : *Bus Busy*. Cette entrée/sortie indique que le bus est en activité. Seulement après que $\overline{BB}$ est désactivé que le système en attente de bus devient le maître de bus.

• **$\overline{CAS}$** : *Column Address Strobe*. Cette sortie a trois états employé pour interface avec une mémoire vive dynamique DRAM.

#### I.3.1.2.4 interruption et commande du mode de fonctionnement :

Les connexions suivantes sont associées aux interruptions matérielles du DSP56303 et permettent également de sélectionner le mode de fonctionnement du processeur.

**a) $\overline{RESET}$** : (Initialisation), cette entrée équipée d'un trigger de Schmitt permet de déclencher la phase d'initialisation du DSP56303

**b) MOD[A-D] :** Quand le DSP56303 sort de l'état initial, il échantillonne les lignes du mode de fonctionnements MOD | A-D | et charge leurs valeurs dans les bits de MA, MB, MC et MD dans l'OMR (*Operating Mode Register*) pour placer le mode initial d'opération dans un état qui corresponde à l'état de ce dernier.

**c) $\overline{IRQ[A-B]}$** : Apres la sélection du mode de fonctionnement, ces broches deviennent des entrées d'interruptions matérielles. Chaque interruption correspond à un branchement vers une adresse bien spécifique.

#### I.3.1.2.5 interface Hôte HI08 (port B) :

Cette interface permet de relier facilement un autre processeur au DSP56303 à travers son port B. Ce dernier est composé de bus de données hôte (maître) de huit bits, des sept adresses hôte sur trois bits et de quatre bits de commande assurant les transferts de données. Le port HI08 (*Host Interface*) peut fonctionner en mode multiplexé ou non-multiplexé selon l'état des registres de control, dans le cas où les registres de contrôles sont effacés, les lignes de cette interface sont utilisées comme des entrées/sorties digitales.

**a) non multiplexé :** En mode non-multiplexé, le HI08 exige un signal chip select et trois lignes d'adresse pour choisir un des huit registres accessibles à l'interface Hôte. Huit lignes sont utilisées pour des données.

• **H[0-7]** : *Host Data*. Ces lignes bidirectionnelles où les transferts de données entre un processeur hôte (maître) et le DSP56303 sont effectués sur ce bus de huit bits.

• **HA[0-2] :** *Host Address.* Ces trois entrées permettent d'accéder aux différents registres de l'interface Hôte, ils peuvent être programmés soit en sortie soit en entrée.

• **HRW** : *Host Read/Write.* Cette entrée permet de sélectionner la direction du transfert à chaque accès du processeur hôte.

• **HDS:** *Host Data Strobe.* Cette entrée qui s'associe avec HRW pour sélectionner la direction (entrée ou sortie) et le sens (lecture ou écriture) du transfert des données.

• **HCS :** *Host Chip Select.* Cette entrée est utilisée pour la sélection de l'interface Hôte.

• **HREQ :** *Host Request.* Ce signal est engendré par l'interface Hôte lors d'une requête adressée à un processeur hôte, à un contrôleur DMA ou simplement à un contrôleur externe quelconque.

• **HACK :** *Host Acknowledge.* Deux fonctions sont réalisées par cette entrée. Elle permet, d'une part, de gérer, par échange d'accusé de réception ou de poignée de main, le transfert des données en DMA et, d'autre part, de recevoir une interruption d'acquittement du processeur hôte.

**b) multiplexé:** En mode multiplexé, les signaux des huit lignes d'adresse inférieures sont multiplexés avec les huit lignes de données.

• **HAD[0-7] :** *Host Address.* Ces signaux sont des lignes bidirectionnelles du bus d'Adresse / Données.

• **HAS :** *Host Address Strobe.* Cette entrée est faite pour sélectionner la direction de l'adresse de l'hôte.

• **HA[8-10] :** ***Host Address.*** Ces signaux sont les lignes 8-10 du bus d'adresse de l'interface Hôte.

• **HRD :** *Host Read Data.* Cette entrée permet à l'interface Hôte de lire les données.

• **HWR :** *Host Write Data.* Cette entrée permet à l'interface Hôte d'écrire des données

• **HTRQ :** *Transmit Host Request.* Cette sortie est la demande de l'interface Hôte de transmission.

• **HRRQ :** *Receive Host Request.* Cette sortie est la demande de l'interface Hôte de réception.

### I.3.1.2.6 interface de communication série synchrone ESSI ( Port C et D ) :

Les six broches réservées à chacune des interfaces série synchrone ESSI (*Enhanced Synchronous Serial Interface*) permettent le transfert rapide de données sous une forme série synchrone.

**a) SC[0-1]0** : *Serial Control Signal 0.* Cette broche est utilisée comme indicateur d'entrée et de sortie de donnée.

**b) SC[0-1]1** : *Serial Control Signal 1.* L'interface SSI utilise cette connexion bidirectionnelle uniquement comme indicateur de transfert.

**c) SC[0-1]2** : *Serial Control Signal 2.* L'interface SSI utilise cette connexion bidirectionnelle t comme indicateur de synchronisation de trame.

**d) SCK [0-1]** : *Serial Clock.* Cette ligne bidirectionnelle délivre ou reçoit le signal d'horloge du débit des données lorsqu'une seule horloge de référence est utilisée par les systèmes de communication.

**e) SRD[0-1]** : *Serial Receive Data.* Cette entrée réceptionne les données en série et les transfère dans le registre de réception (registre à décalage).

**f) STD [0-1]** : *Serial Transmit Data.* Les données en provenance du registre d'émission, sont envoyées en série vers l'extérieur par l'intermédiaire de cette connexion.

### I.3.1.2.7 interface de communication série asynchrone SCI ( port D ) :

Trois broches sont réservées à l'interface de communication série SCI (*Serial Communication Interface*). Elles assurent la transmission sous la forme série des trois octets.

**a) RXD** : *Serial Receive Data.* Cette entrée réceptionne les bits sous une forme série. Ces bits sont transférés dans un registre à décalage qui constitue le registre de réception de l'interface SCI.

**b) TXD** : *Serial Transmit Data.* Cette sortie permet de transmettre les bits sous une forme série, les bits de la donnée contenue dans le registre de transmission de l'interface SCI.

**c) SCLK** : *Serial Clock.* c'est la broche d'horloge de transmission qui peut être une entrée ou une sortie au DSP ( selon la configuration du Registre de control ) et elle fournit une horloge séquentielle de débit binaire pour le SCI

**I.3.1.2.8 timer triple :**

Le DSP56303 a trois Timers identiques et indépendants. Chaque temporisateur peut utiliser une horloge interne ou externe, et il peut interrompre le DSP56303 après un certain nombre d'événements (horloge) ou signaler un dispositif externe après le compte d'un certain nombre d'événements internes.

**TIO[0-2] :** *Timer Schmitt-Trigger Input/Output.* Les broches TIO assurent la liaison entre l'extérieur et le module temporisateur et compteur. Dans le mode de fonctionnement en compteur d'évènements, les lignes TIO sont configuré en entrée et permet de mesurer la période d'un signal ou la largueur d'une impulsion. Quand la fonction temporisateur est sélectionner, TIO est configuré en sortie, et délivre un signal impulsionnelle.

**I.3.1.2.9 émulateur intégré OnCE /JTAG :**

Le module d'OnCE fournit un moyen de se connecter par interface série au noyau DSP56303 et à ses périphériques de sorte qu'il est possible d'examiner des registres, des mémoires ou des périphériques interne

**a) TCK :** *Test Data Input.* C'est une entrée d'horloge qui à pour but de synchroniser la logique d'essai.

**b) TDI :** *Test Data Input.* C'est une entrée où les tests séquentiels d'instruction et des données sont reçus.

**c) TDO :** *Test Data Output.* C'est une sortie où les tests séquentiels d'instructions et des données sont transmis.

**d) TMS :** *Test Mode Select.* Cette entrée est utilisée pour sélectionner l'état du contrôleur de test.

**e) TRST :** *Test Reset.* Cette entrée permet l'initialisation du contrôleur d'essai.

**f) $\overline{\text{DE}}$ :** *Debug Event.* L'activation DE fait que le DSP56303 termine l'exécution de l'instruction actuelle, et entre dans le mode Débuggeur. Une impulsion est produite sur la ligne DE chaque fois que le DSP reconnaît l'exécution d'une instruction dans le mode Débuggeur.

**I.3.1.2.10 entrée-sortie d'usage universel (GPIO) :**

Dans le cas où des périphériques ne sont pas utilisé (Timer, Interfaces série, Interface Hôte), les lignes de ces derniers peuvent être configurés comme signaux GPIO (*General-Purpose Input/Output*), et le DSP56303 les utilise comme 34 signaux digitaux bidirectionnels.

## I.3.2 TYPE DE MEMOIRE UTILISER :

Pour profité du parallélisme d'adressage du DSP56303, trois type de mémoire sont utilisés

Mémoire Flash
Mémoire actif du programme
Mémoire de données ("X / Y")

### I.3.2.1 Mémoire Flash :

Un circuit de mémoire Flash (instantané) est installé sur la carte qui forme la mémoire non-volatile pour les micro-programmes, les variables d'installation (variables I), les programmes utilisateurs, les tables de conversion, et les algorithmes de commande (par exemple algorithme de PID). Vue par le DSP56303, la mémoire flash est situé dans le champ de mémoire " P ", ce circuit de taille 1M x 8 bits est le **28F008S3** de **Intel,** il est situé à U10 dans la carte.

### I.3.2.2 Mémoire actif du programme :

Un arrangement de trois circuits de mémoire SRAM (*Static RAM*, qui n'a pas besoin de rafraîchissement) sont installés sur la carte, ces circuits forment la mémoire de compilation/assemblage active (résultats et données intermédiaires) pour les micro-programmes, les PLCs compilé et les algorithmes de commande écrits par l'utilisateur. Vue par le DSP56303 les mémoires SRAM de compilation/assemblage sont situé dans le champ de mémoire " P ", ces trois circuits de taille 128K x 8 bits chacune, sont des **GVT73128A8J-10** de **GALVANECH,** ils sont situées dans U14, U15, et U16 dans la carte.

### I.3.2.3 Mémoire de données ("X / Y") :

Un arrangement de trois circuits de mémoire SRAM est installé sur la carte, ces circuits forment la mémoire active pour la sauvegarde des programmes de mouvement d'utilisateur, les programmes PLC non compilés et les données. Vue par le DSP56303 les mémoires SRAM de données sont situés dans les champs de mémoire " X " et " Y ", ces trois circuits de taille 128k x 8 bits chacune, sont des **TC55V8128B** de **TOSHIBA**, ils sont situés dans U11, U12, et U13 dans la carte.

## I.3.3 CONNECTEUR SERIE RS-232/422 [9]:

Un connecteur série RS-232/422 DB25 est mis en place pour la communication séquentielle entre l'ordinateur principale et le Turbo PMAC2-3U CPU, cette communication se fait avec un protocole asynchrone en mode half-duplex (transmission en deux sens mais pas simultanément) et un format de donnée en code ASCII, la disposition des broches de ce connecteur est détaillé dans l'Annexe 1.

Cette fois si, les lignes de l'interface SCI (*Serial Communications Interface*) du DSP56303 sont utilisées, car les lignes RXD, TXD et SCLK du port série sont connecté à cette interface, après une adaptation des signaux (CMOS/TTL) faite par le circuit **MAX 8215** situé à U47.

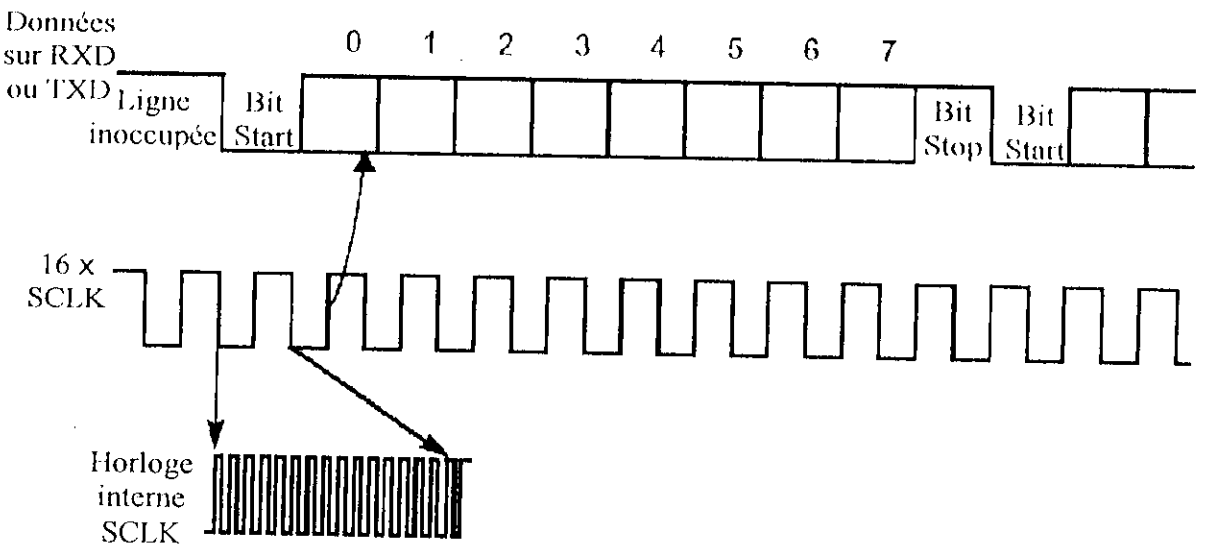



**Figure I.7** : Transmission série des données

Comme le montre la Figure I.7, l'émission et la réception utilisent une horloge interne qui est 16 fois la fréquence du débit des données au SCI. Le format de données exige que chaque octet de données à un bit de départ ( Start ) et un bit d'arrêt (Stop ) sans bits de parité et une vitesse de transmission 38200 bits/sec (baud rate).

Par défaut, la connexion RS232 est utilisée, elle se caractérise par des signaux de transmission bipolaire ±12 V, ceci dit par configuration la connexion RS422 peut être utilisée pour des signaux de transmission différentielle 5 V.

### I.3.4 *WHATCHDOG TIMER* (CHIEN DE GARDE) [4], [9]:

C'est un système dont le travail est de détecter un certain nombre de conditions, comme une sous tension ou une faible fréquence d'exécution des programmes, qui pourraient avoir des conséquences dangereuses sur le fonctionnement de la carte ; Si c'est le cas il se déclenchera, et arrêtera la carte, le fonctionnement détailler de ce circuit sera décrit après dans le chapitre III.

Le périphérique Timer triple de le DSP56303 est utilisé pour réaliser la fonction de Whatchdog timer, de ce faite une ligne TIO est utilisé en entrée pour le comptage d'évènements et une autre ligne TIO est utilisé en sortie pour délivrer un signal impulsionnel qui arrête complètement le système une fois nécessaire.

### I.3.5 REGISTRES DE CONTROLE :

La représentation des registres de contrôle dans un block dans la Figure I.4 n'est que symbolique, car ces derniers sont répartis dans des zones mémoire bien séparé du CPU et dans toutes les autres cartes (DSPGATEs et IOGATE), ces registres fixe l'état des entrées/sorties ainsi que le fonctionnement du système global. Pour ce fait des jumpers (Annexe 1) sont installés sur la carte ainsi des variables I [7] accessibles par logiciel pour configurer l'état des registres de contrôle, en conséquence personnalise le fonctionnement.

## I.4 DESCRIPTION DE LA DSPGATE [10]:

L'unité centrale de traitement (Turbo Pmac2 3U CPU) communique avec les capteurs et les amplificateurs par un assemblement de circuit fait sur commande particulière de **DELTA TAU** [1], désigné sous le nom de DSPGATES (portes de la DSP). Nous disposons de deux DSPGATES classes comme accessoire ACC-24E 4A, qui sont équivalent à deux ACC24E2A avec l'option 1, la Figure I.8 montre l'image d'une DSPGATE.

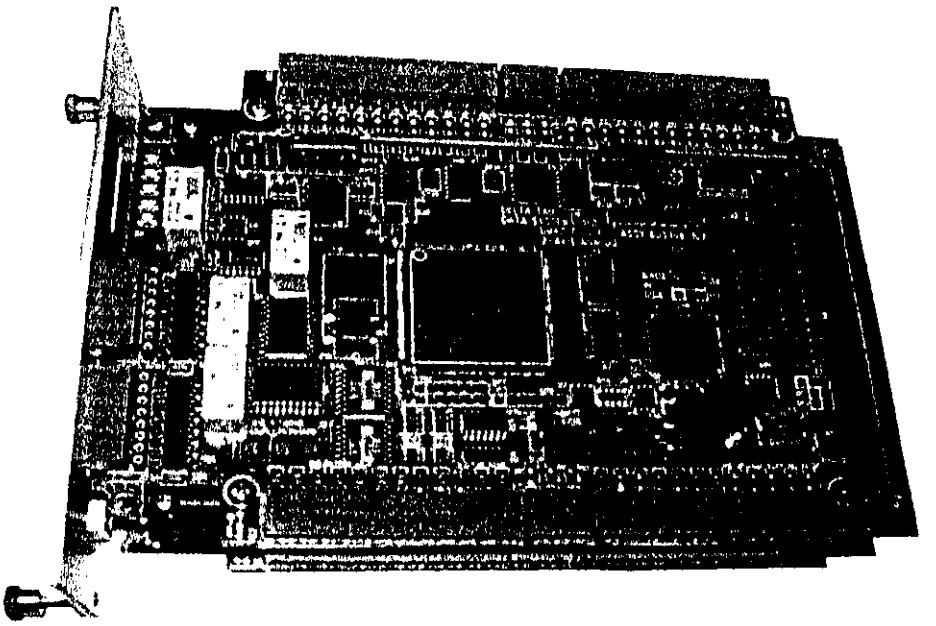


**Figure I. 8** : DSPGATE ACC-24E 4A

Chaque DSPGATE contient quatre canaux (pour commande jusqu a 4 moteurs), de ce fait, il y'a huit canaux qui peuvent être utilises pour commander jusqu'à huit moteurs. Connectes à la surface arrière UBUS, l'ACC-24E 4A prend un total de deux slots (deux connections).

Une DSPGATE ne contient aucun processeur, il y'a quatre canaux contenant les circuits de conversion numérique/analogique DAC, compteurs, temporisateurs et filtres digitaux pour la commande en associant autour d'eux des buffers et des connecteurs pour les accès entrée/sortie.

Cette DSPGATE permet les type de commande suivant :

- Commandes en modulation de largeur des impulsions de ±10V.
- Commandes analogiques de couple de ±10V.
- Commandes d'Impulsion -et- direction pour les moteurs pas à pas.

Chaque canal de ces DSPGATE est constitué par :

- 3 sorties pour les signaux de commande des amplificateurs configurables pour les types de commande suivant :
  - 3 sorties de commande en tension avec modulation de largeur des impulsions PWM.
  - 2 sorties de Convertisseur Digital Analogique DAC, commande en courant.
  - 1 sortie de signaux impulsion -et- direction pour les moteurs pas à pas.
- 3 entrées pour lignes de l'encodeur incrémentale.
- 8 indicateurs entrés, 2 indicateurs sorties.

Les lignes de ces entrées/sorties sont apportées sur des connecteurs nommés TB (Annexe1).

**Remarque** : Vue par le CPU et exactement par la DSP les registres des DSPGATEs et IOGATE sont considérés comme des zones mémoire qu'on peut lire ou écrire.

### I.4.1 GENERATEUR DE SIGNAUX DE SYNCHRONISATION [6]:

Dans la DSPGATE, chaque processus est synchronisé par un signal d'horloge qui est bien spécifique, ces signaux d'horloge se distinguent les uns des autres et ils sont génères à partir de quelques transformations sur signal d'horloge de base (20 MHz), comme le montre la Figure I.9, ces transformations se font avec des multiplicateurs et diviseur de la fréquence de base.

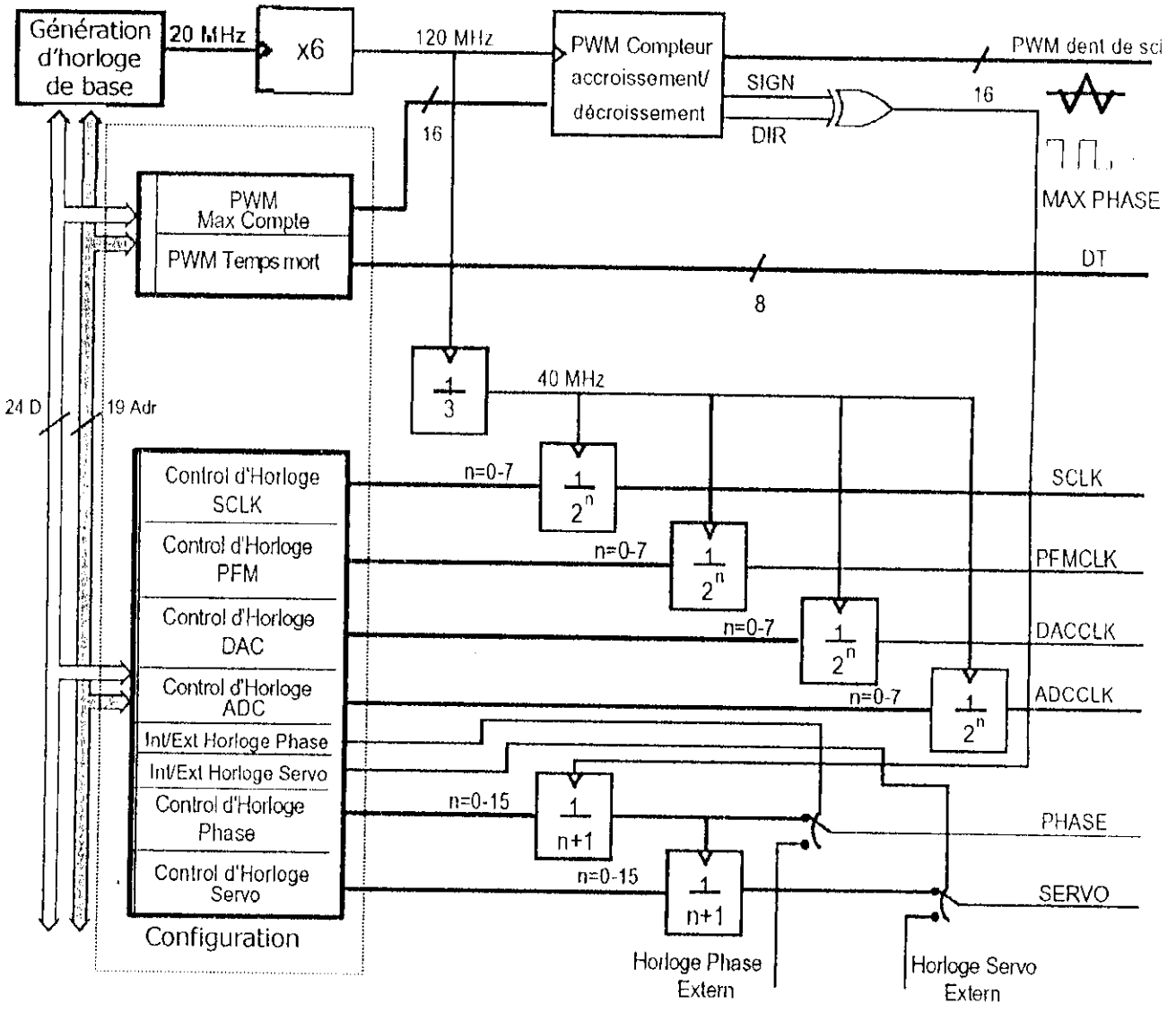



**Figure I.9** : Block générateur de signaux de synchronisation dans la DSPGATE et control de fréquence

Comme le montre la Figure I.9 pour chaque horloge les facteurs de division ne sont pas fixe mais variable selon la configuration des variables I [7]. Les différents signaux d'horloge utilisés sont :

• **SCLK :** *Encoder Sample ClocK* (par défaut 39.3216 MHz). Horloge simple d'encodeur, elle est utilisée dans plusieurs processus, comme le filtrage digital.

• **DACCLK :** *Digital-to-Analog Converter Clock* (par défaut 4.9156 MHz). Horloge de convertisseur digital/analogique, il est directement utilisé pour les signaux de sortie de commande en courant, pour contrôler la fréquence du flux des données de la commande au convertisseur.

• **ADCCLK :** *Analog-to-Digital Converter Clock* (par défaut 2.4576 MHz). Horloge de convertisseur analogique/ digital, il est directement utilisé dans la boucle digitale de courant (si elle est activer), pour contrôler la fréquence du flux de données de mesure du courant au convertisseur.

• **DT :** *Dead time* (par défaut 2.03 µs). Temps mort de la commande avec modulation de largeur des impulsions pour la boucle de courant digital.

• **Dent de scie PWM :** (la période par défaut 1.526 µs). Ils sont utilises pour la commande avec modulation de largeur des impulsions.

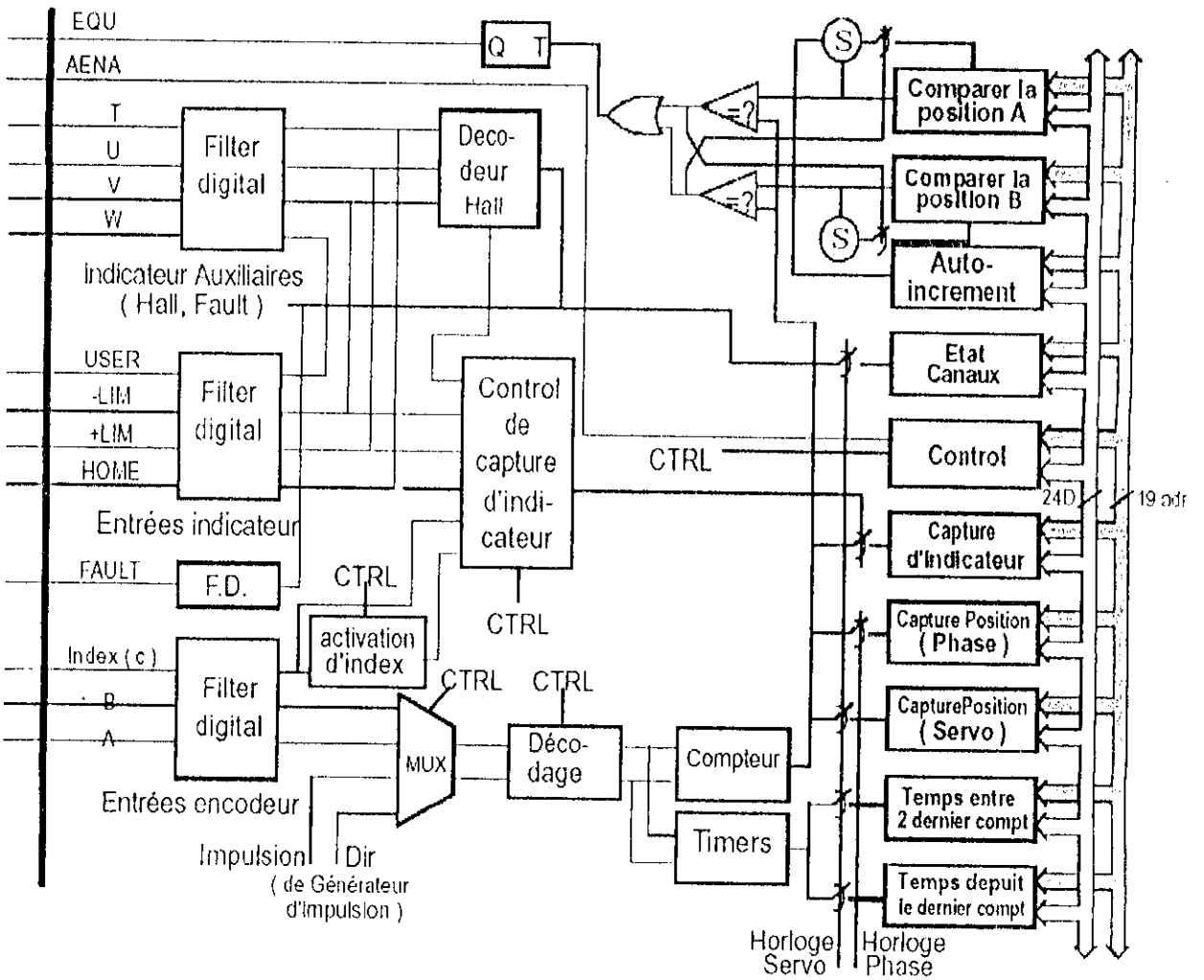
• **PHASE** : *PHASE Clock* (par défaut 9.0346 KHz). Elle est utilisée pour la fermeture de la boucle digitale de courant (si elle est activée), et pour la commutation des phases des moteurs (pour les moteurs qui ne sont pas à simple phase).

• **SERVO** : *SERVO Clock* (par défaut 2.2587 KHz). Elle est utilisée pour la commande de position et la fermeture de la boucle de commande vitesse/position.

**Remarque :** Les horloges PHASE et SERVO fonctionnent comme des interruptions matérielles au processeur, elles peuvent être internes ou externes selon la configuration (externe : par L'UBUS). Pour un bon fonctionnement du système, la source de ces horloges doit être unique pour tout les cartes pressent dans le système.

### I.4.2 ENCODEUR INCREMENTALE ET INDICATEURS [6]:

Chaque canal de la DSPGATE à 3 entrées de l'encodeur incrémental pour les signaux de retour de la mesure de position et plusieurs entrées/sorties indicateurs, la Figure I.10 montre comment ces entrées/sorties sont traitées en schéma block.




**FigureI.10 :** Block entrées/sorties encodeur et indicateurs

Comme le montre la Figure I.10 ces signaux sont en premiers lieux traites par un filtre digital pour éliminer l'effet du bruit, puit ils sont suivis d'une logique de contrôle et de multiplexage qui est selon les configurations de ces entrées/sorties.

### I.4.2.1 Filtre digitale [7] :

Ce filtre est composé d'un ensemble de quatre bascules D montés en cascade sur chaque ligne d'entrée digitale, par exemple la Figure I.11 illustre le filtrage des lignes CHA et CHB des entrées d'encodeur.

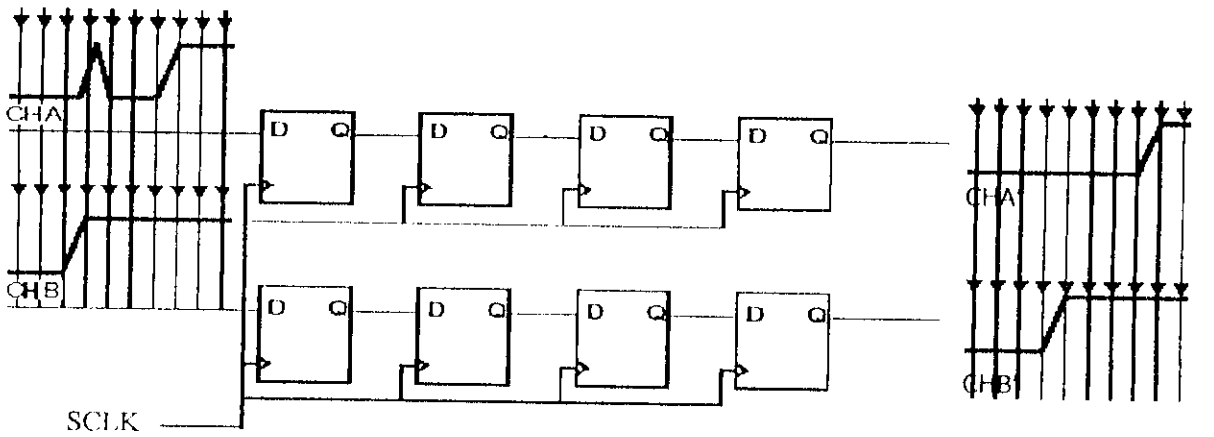



**Figure I.11** : Filtre digital implémenté dans la DSPGATE

Les bascules D du filtre digital sont synchronisées par le signal d'horloge de SCLK, de ce fait le rendement de ce filtre peut être amélioré par augmentation ou par diminution la fréquence de signal d'horloge SCLK.

### I.4.2.2 Entré encodeur [6]:

Il y'à trois entrées d'encodeur incrémental CHA, CHB et CHC, où comme le montre la Figure I.12 l'encodeur envoie des imputions quant il y'a un mouvement du moteur, pour la mesure de position, ces impulsions ont un rôle important ceci est décrit par la suite :

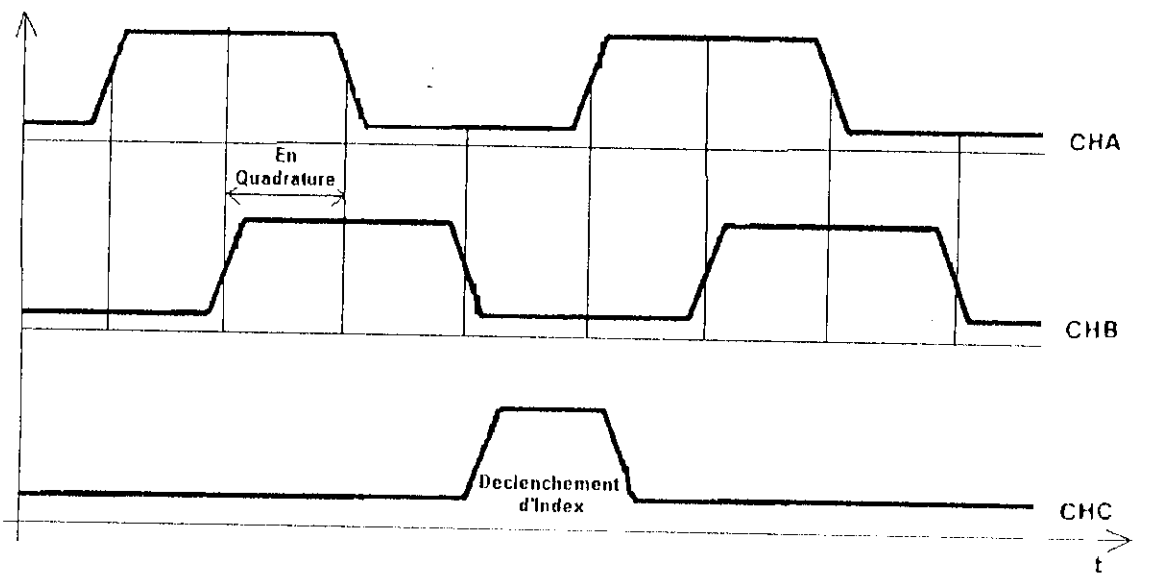



**Figure I.12** : Signaux délivrés par l'encodeur lors d'un mouvement

• **CHA :** Quant le moteur tourne, le disque de l'encodeur tourne et la DPSGATE reçoit sur CHA des impulsions a chaque incrément, ces impulsions vont être acheminées vers un compteur et deux timers pour déterminer la position du disque et le temps correspondent respectivement.

• **CHB :** De même la DPSGATE reçoit sur CHB des impulsions à chaque incrément, sauf que cette fois ci ces impulsion sont en quadrature par rapport aux impulsions qui sont reçues sur CHA, ces impulsions sont importantes pour déterminer le sens de la rotation du disque de l'encodeur, car dans le cas ou ces impulsions seront en retard par rapport aux impulsions qui sont reçues sur CHA, le disque tourne dans le sens positif et le compteur qui est utilisé pour la mesure de la position s'incrémente par quatre (pour plus de précision), et dans le cas contraire alors le disque tourne dans le sens négatif et le compteur se décrémente par quatre. la détermination du sens de la rotation est fait matériellement par le décodeur installé dans la DSPGATE qui est formé par deux bascules D montés en parallèle comme il est montré par la Figure I.13.

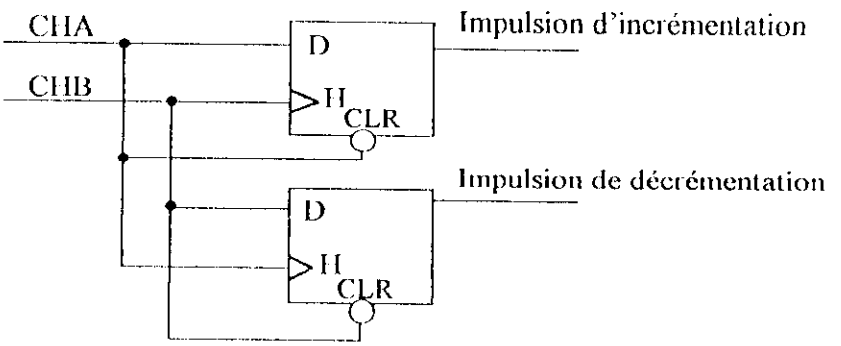



**Figure I.13** : Décodeur du sens de mouvement implémenté dans la DSPGATE

• **CHC** : cette entrée représente l'index de l'encodeur qui reçoit une impulsion à chaque tour du disque de l'encodeur, cette dernière est importante pour la détermination du nombre exact d'incrément par tour (ou compte par révolution), comme c'est le cas pour les moteurs dont nous disposons qui ont un encodeur avec 500 CPR.

Matériellement ces entrées, peuvent recevoir des signaux asymétriques (c-a-d la tension sera entre 0 et 2.5V) sur les ligne CHA+/AGND, CHB+/AGND et CHC+/AGND respectivement ou des signaux différentiels (c-a-d la tension sera entre Umin et Umax avec Umax–Umin=5V) sur les ligne CHA+/ CHA-, CHB+/ CHB- et CHC+/CHC- respectivement.

### I.4.2.3 Indicateurs [6] :

Pour chaque canal il y'à huit entrées et deux sorties digitales d'indicateur qui peuvent être activées ou désactivées selon la configuration, ces indicateurs fonctionnent comme des interruptions au processeur pour qu'il exécute des taches bien particulières quant il le faut : comme l'arrêt (+LIM ) de mouvement des moteurs en cas de nécessité, ces indicateurs sont :

• **+LIM, -LIM** : ces entrées fournissent un fonctionnement important de sûreté et d'exactitude du mouvement du moteur, ils sont les limites de mouvement dans les deux directions sensibles aux dépassements, de ce fait +LIM devrait être placé à la fin négative du déplacement, et - LIM devrait être placé à la fin positive du déplacement, les noms des connecteurs réels de ces entrées sont PLIMn et MLIMn.

• **HOME** : cette entrée est utilisée pour la fonction d'enregistrement de position, pour qu'au repos ou à l'application de la commande HOME (sur logiciel), le moteur revient automatiquement à la position enregistrée, le connecteur réel de cette entrée est nommé HMFLn.

• **T, U, V, W et USER** : Il y a cinq entrées supplémentaires d'indicateur pour chaque axe, qui peuvent être utilisées pour plusieurs applications si elles sont activées, l'une de ces applications (Figure I.14) est la mesure avec capteurs a effet Hall qui ont trois sorties digitales connecter à ces indicateurs où le mot compose de ces trois sorties donne un code qui correspond à une certaine position calculée par le Décodeur Hall (Figure I.10), les connecteurs réels de ces entrées sont nommés : CHTn, CHUn, CHVn, CHWn et USERn.

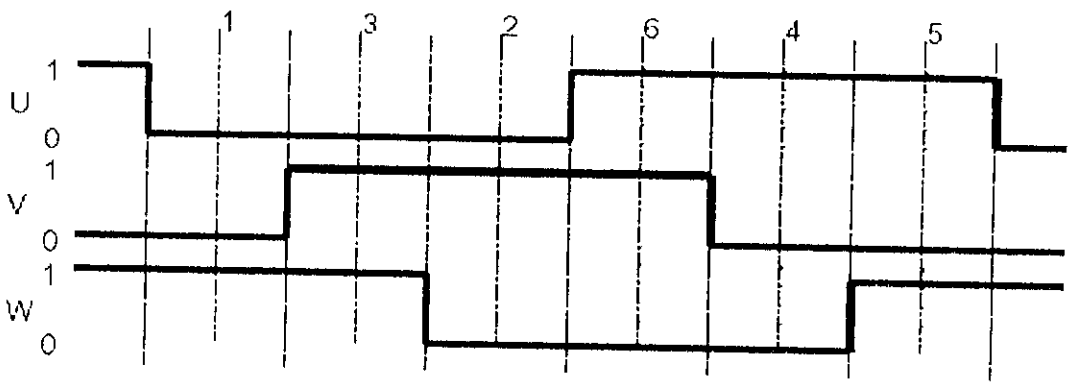



**Figure I.14** : Utilisation des indicateurs supplémentaires pou les capteurs à effet Hall

• **FAULT :** *Amplifier Fault.* Lorsque qu'il y'à un défaut d'amplification ou quelque chose fonctionne mal dans un amplificateur (comme une sous tension d'alimentation), un signal FAULT est émis à cette entrée par cet amplificateur pour indiquer au CPU qu'il y'a un mauvais fonctionnement, les connecteurs liés à cette entrée sont nommés AFAULT_n+ et AFAULT_n-.

• **AENA :** *Amplifier Enable.* Commandés par le CPU cette sortie est utilisée pour active ou désactive instantanément les axes d'amplificateur, cette opération est très importante pour des raisons de sûreté, s'assurent que l'amplificateur peut être complètement arrêté une fois nécessaire, car une valeur nulle de la sortie de commande peut entraîner des retards du courant qui peuvent facilement s'accumuler de sorte qu'une commande nulle ne cause pas un arrêt immédiat, les connecteurs liés à cette sortie sont nommés AE_NO_n, AE_NC_n, AE_COM_n.

• **EQU :** *Compare-Equals.* Cette sortie est le résultat de la comparaison de la position réelle par rapport au contenue de deux registres (A et B), cette sortie est utilisée pour des applications bien particulières qu'on ne peut cité brièvement, le connecteur liés à cette sortie est nommé BEQUn.

**Remarque** : le connecteur de la ligne du signal de retour des indicateurs USERn, +LIMn, -LIM, HOME est nommés FLG_n_RET (*Flag Return*).

#### I.4.2.3.1 isolation optique des indicateurs [10] :

afin de protéger les circuits digitaux des courants forts, les indicateurs sont optiquement isoles à l'aide du circuit **PS2705-4NEC-ND** qui est composé par des phototransistors, ceci est illustré par la Figure I.15.

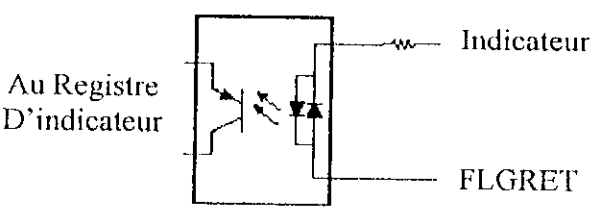



**Figure I.15** : Isolation optique des indicateurs

### I.4.2.3.2 branchement sourcing et branchement sinking [10] :

A l'exception des indicateurs T, U, V et W, qui s'activent à une tension de 5V, à l'état de fonctionnement normal les indicateurs doivent être branchés à une tension de 12V à 24V avec FLGRET, et si cette tension disparait alors un état de haute impédance apparait sur les bornes d'un indicateur (la lumière ne passe pas aux phototransistor) ce dernier s'active pour que le CPU exécute la tache qui le correspond.

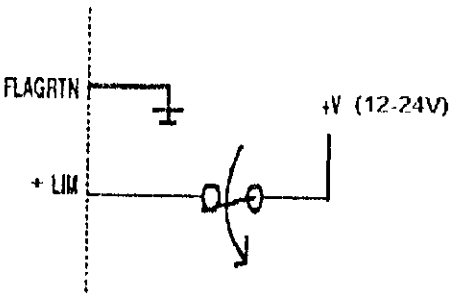



**Figure I.16.a** : Branchement en mode Sinking des indicateurs

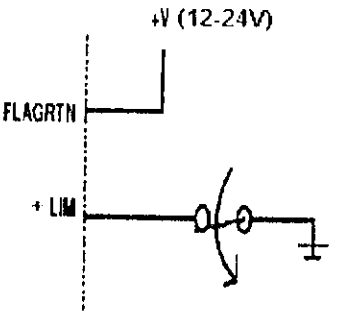



**Figure I.16.b** : Branchement en mode Sourcing des indicateurs

Ces indicateurs peuvent être branches soit en mode Sourcing (FLGRET à la masse et indicateur à la tension 12 à 24V) ou soit en mode Sinking (indicateur à la masse et FLGRET à la tension 12 à 24V), car dans les deux la lumière passe aux phototransistors, ceci est illustre par les Figures I.16.a et I.16.b.

## I.4.3 SORTIE DE COMMANDE [6]:

Chaque canal de la DSPGATE à trois sorties de commande A, B et C, chacune de ces sorties peut être configurés selon plusieurs type de commande, ces sorties sont optiquement isolées des circuits d'amplification (pour protéger les circuits digitaux des courants de forte intensité). La Figure I.17 schématise la façon dont les différents types de commande sont génère par la DSPGATE.

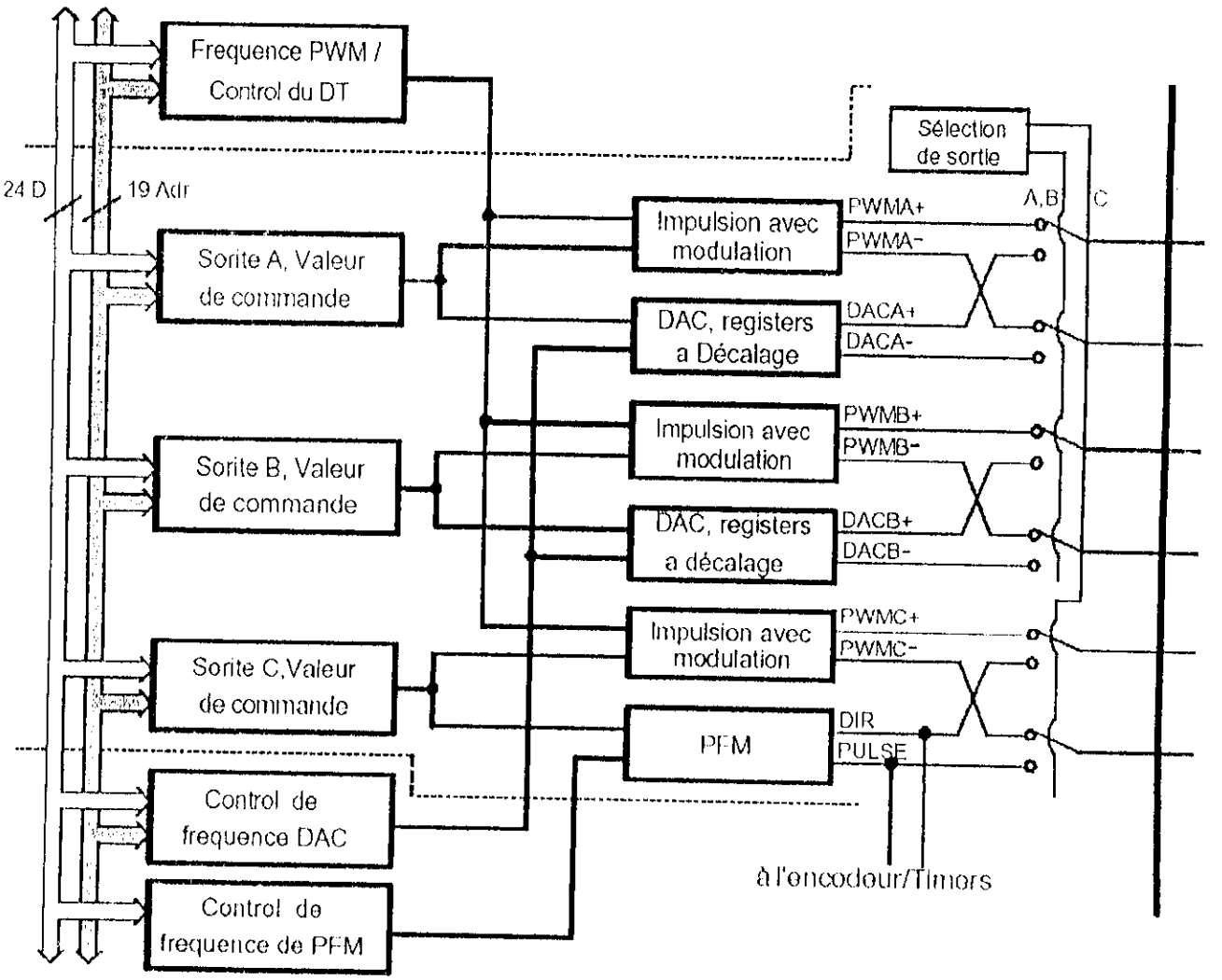



**Figure I.17**: Block de sortie de commande délivré par la DSPGATE

• **Sortie A :** Cette sortie peut être configurée soit en sortie DAC soit en sortie PWM. La sortie DAC (*Digital-to-Analog Converte*) est une commande en courant que reçoit l'amplificateur (en cas d'absence de la boucle courant digital), qui est une tension analogique entre 10 et -10V, résultant d'une simple conversion numérique/analogique avec une résolution de 0.0003V de la valeur du signal de commande, synchronisée par le signal d'horloge DACCLK. La valeur du signal de commande est écrite dans un registre de 16 bits situé dans la DSPGATE par le CPU après l'application de l'algorithme de commande de position, ce type de commande nécessite un amplificateur de courant (comme c'est le cas). La sortie PWM (*pulse width modulation*) qui commande en tension avec modulation de largeur des impulsions, est une tension à deux états +10 et –10 V, synchronisé par le signal d'horloge PHASE, et avec un rapport cyclique calculé par le CPU après l'application de

l'algorithme de commande de position et l'algorithme de la commande de courant digital, ce type de commande nécessite un amplificateur de tension (ce n'est pas le cas). Les connecteurs lies à la sortie A sont nommés DACnA+ et DACnA+.

• **Sortie B :** de même pour la sortie B, elle peut être configuré soit en sortie DAC soit en sortie PWM, les connecteurs lies à cette sortie sont nommés DACnB+ et DACnB+.

• **Sortie C :** cette sortie peut être configurée soit en sortie commande en tension PWM, et soit en sortie PULSE et DIR, qui est une commande sur deux lignes, synchronisé par le signal d'horloge PFMCLK, où sur la premier ligne l'amplificateur reçoit la commande d'impulsion, et sur l'autre il reçoit la commande de direction (signe), ce type de commande est nécessaire pour la commande des moteurs pas à pas, les connecteurs liers à cette sortie sont nommés PUL_n-, PUL_n+, DIR_n-, DIR_n+.

L'utilité de la présence de ces trois sorties par canal n'est pas pour pouvoir commander plusieurs moteurs à la fois, mais de pouvoir commander plusieurs types de moteurs comme les moteurs à courant continue, les moteurs triphasés et les moteurs pas à pas.

Par exemple la commande des moteurs triphasés avec boucle de courant digital où les trois sorties sont utilises comme suit :

- PWMA+, PWMA- sont branche à la phase A du moteur.
- PWMB+, PWMB- sont branche à la phase B du moteur.
- PWMC+, PWMC- sont branche à la phase A du moteur.

## I.5 DESCRIPTION DE L' IOGATE [11]:

Nous disposons d'une carte IOGATE (*Input/Output GATE*) classée comme accessoire ACC12E de DELTA TAU, c'est une carte d'entrée/sortie d'usage universel au système UMAC, en format 3U conçus pour la transmission et la réception des donner de l'extérieur du système au CPU, par l'intermédiaire de l'UBUS, elle fournit 24 lignes d'entrées optiquement isolées et 24 lignes de sorties à tension élevée qui sont aussi optiquement isolés, la lecture et l'écriture des données d'entrées/sorties est effectuée en utilisant les variables M, qui correspondent à des pointeurs de zone mémoire (voir le chapitre II), la Figure I.18 montre l'image de cette carte.

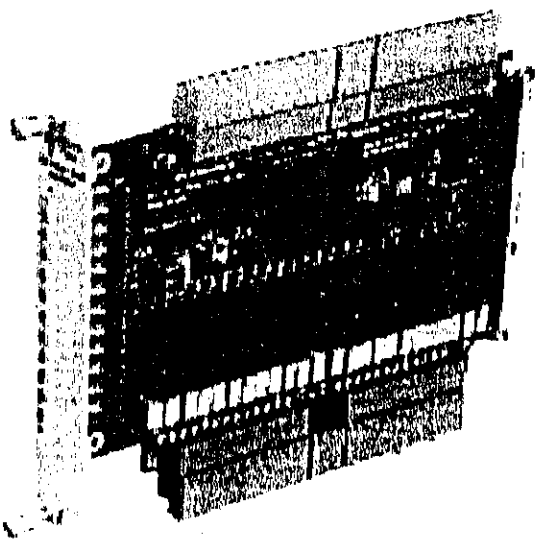


**Figure I.18** : IOGATE

### I.5.1 CIRCUIT D'ENTREE :

Les entrées au de la carte IOGATE sont activées dans un intervalle de 12V à 24V, et elles peuvent être soit au mode sinking ou soit en mode sourcing (décrit auparavant), selon la tension de référence connectée à la carte, Le circuit d'entrée opto-isolation utilisé est le **PS2705-4NEC-ND,** il est composé de quadripôle comprenant un phototransistor à la sortie (Figure 19), de ce fait, le circuit permet au courant de circuler librement dans les deux sens.

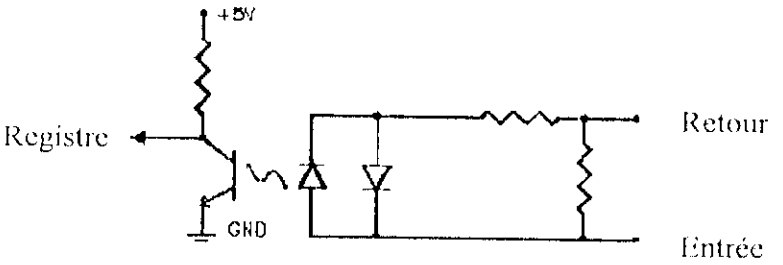



**Figure I.19** : Circuit d'entrée de IOGATE

### I.5.2 CIRCUIT DE SORTIES :

Le relais semi-conducteur utilisé pour les sorties à courant continue est le **CRYDOM DMO063**, et pour les sorties à courant alternatif c'est le **CRYDOM ASO242**, pour chaque circuit le courant de sortie maximale est évalué à 1A.

### I.5.3 LOCALISATION DES REGISTRES D'ENTEE/SORTIE :

Un ensemble de jumpers (Annexe 1) disponible sur la carte est utilisé pour fixer les six adresses des registres où le transfert des données entrées/sorties aura lieu, cela est fait ont choisissant la plus basse adresse B qui correspond au premier registre, et les sept autres registres utilisés se trouvant juste après aux adresses {B+ n} (telle que $n \in [1, 7]$). Chaque registre se compose de huit bits, et l'état de chaque bit signifie une ligne d'entrée ou de sortie (selon la direction) dont l'état correspond à l'état de cette ligne, le Tableau I.1 présente la signification de chaque registre. Ainsi ils sont facilement accessibles par logiciel (programme de mouvement ou PLC) en utilisent les pointeurs d'adresse (variable M).

| Adresse | Signification |
|---|---|
| B | Lignes d'entrées 0 à 7 |
| B + 1 | Lignes d'entrées 8 à 15 |
| B + 2 | Lignes d'entrées 16 à 23 |
| B + 3 | Lignes de sorties 24 à 31 |
| B + 4 | Lignes de sorties 32 à 39 |
| B + 5 | Lignes de sorties 40 à 47 |
| B + 7 | Registre de control |

**Tableau I.1** : Localisation des registres d'entées/sorties de l'IOGATE

### I.5.4 CONTROL DES ENTEES/SORTIES:

Pour le CPU les sens de transfert des données (entée ou sortie) des registres utilisés par IOGATE ne sont pas prédéfini d'avance, de ce fait, le registre de contrôle qui est situé à l'adresse {B+7} permet de spécifier l'état de transfert des données pour chaque application, il se compose de deux sections : commande de direction de transfert, et état des registres.

La commande de direction permet à l'utilisateur de placer les octets d'entrées à une lecture seule, ce mécanisme permet de s'assurer que les entrées seront toujours lues correctement. Le bit *n* (de 0 à 5) du registre de control, commande la direction (entrée ou sortie) du registre situé à l'adresse {B + *n*}, car une valeur 0 de ce bit (par défaut) permet la fonction d'écriture du registre qui correspond, et les lignes de ce dernier seront considérées en sortie, tout en permettent l'utilisation de ces lignes comme des entrées, une valeur de 1 dans le bit n de commande ne permet pas l'écriture dans le registre qui correspond, et désactive la sortie, réservant ainsi le registre que pour les entrées.

Dans une application typique, les valeurs différentes de zéro des bits 6 et 7 du registre de contrôle sont seulement utilisées pour l'initialisation.

Pour notre cas, l'utilisation de l'IOGATE implique la configuration du registre de contrôle présenté par la Figure I.20, car les trois premiers registres sont en entrée et les trois derniers sont en sortie.

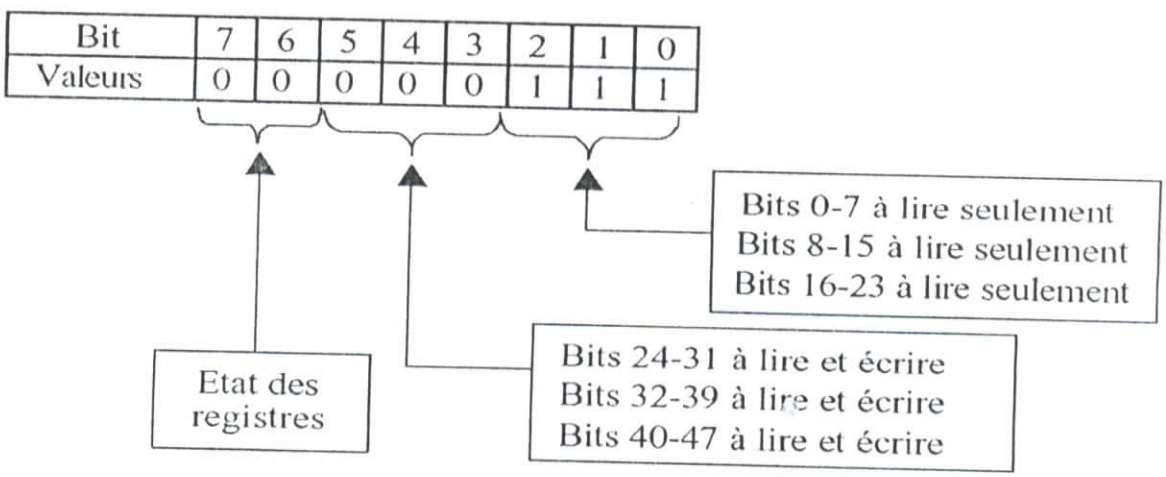



**Figure I.20** : Configuration nécessaire du registre de contrôle de l'IOGATE

## I.6 DESCRIPTION DE L'AMPLIFICATEUR (AMP-2) [13]:

Nous disposons de deux cartes d'amplificateur AMP-2 (quatre axes analogiques, amplification PWM, 48 VDC 3/5A) en format 3U prenons en globalité 4 slot (quatre connections), destines pour être connectes à la DSPGATE ACC24E2A et pour contrôler des moteurs à courant continue ou pas à pas, chaque carte possède quatre axes d'amplifications avec une boucle de régulation analogique de courant (pour la régulation du couple du moteur à courant continue), et avec des signaux de sortie du type modulation de larguer des impulsions PWM. La tension maximum d'alimentation pour ces amplificateurs est de 40VDC, et l'estimation du courant maximum pour chaque amplificateur est de 3A continue pour une puissance maximale de 120W, l'image de cette carte est présenter par la Figure I.21.

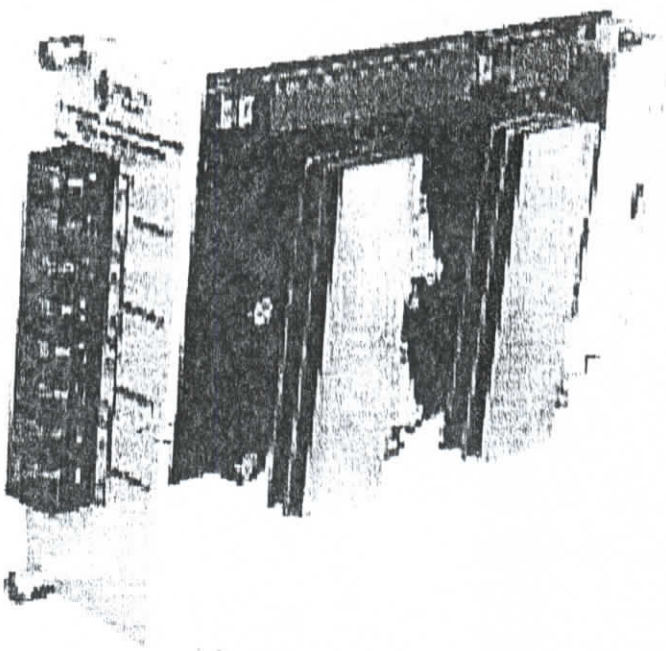


**Figure I.21**: Amplificateur AMP-2

### I.6.1 LIGNES AENA/FAULT :

Comme pour les DSPGATEs, les ligne AENA/FAULT (*Amplifier Enable/Amplifier Fault*) sont utilisées pour la sûreté du fonctionnement. Alors la ligne FAULT est utilisée comme sortie à la DSPGATE pour indiquer s'il y'à un défaut d'amplification, et la ligne AENA est utilisée comme une entrée pour arrêter instantanément l'amplificateur.

### I.6.2 BOUCLE DE COURANT :

D'après l'équation de base des moteurs, le couple est proportionnel au courant qui circule dans le moteur, c'est pour cette raison que ces amplificateurs sont destinés pour être commandé en courant (en couple) par le CPU, ils produisent un courant (respectivement couple) proportionnel à la tension d'entée de commande qui est produite par la DSPGATE par l'intermédiaire de la ligne DAC, tout en appliquant sur les bornes du moteur une tension avec modulation de largeur des impulsions PWM, de ce fait une boucle de courant avec un régulateur PI est implémenté.

L'élément principal de cette boucle est le circuit **LMD18200 3A, 55V H-bridge** de **Nationnal Semiconductor**.

#### I.6.2.1 Description du circuit LMD18200 [21]:

Le LMD18200 est un pont en format H, conçu pour les applications de commande de mouvement, le dispositif est établi en utilisant un processus multi-technologique qui combine les circuits bipolaires et les circuits de commande CMOS avec des dispositifs de puissance DMOS sur la même structure monolithique, idéal pour piloter les moteurs a courant continue et les moteurs pas à pas, la Figure I.22 pressent le schéma interne du LMD 18200.

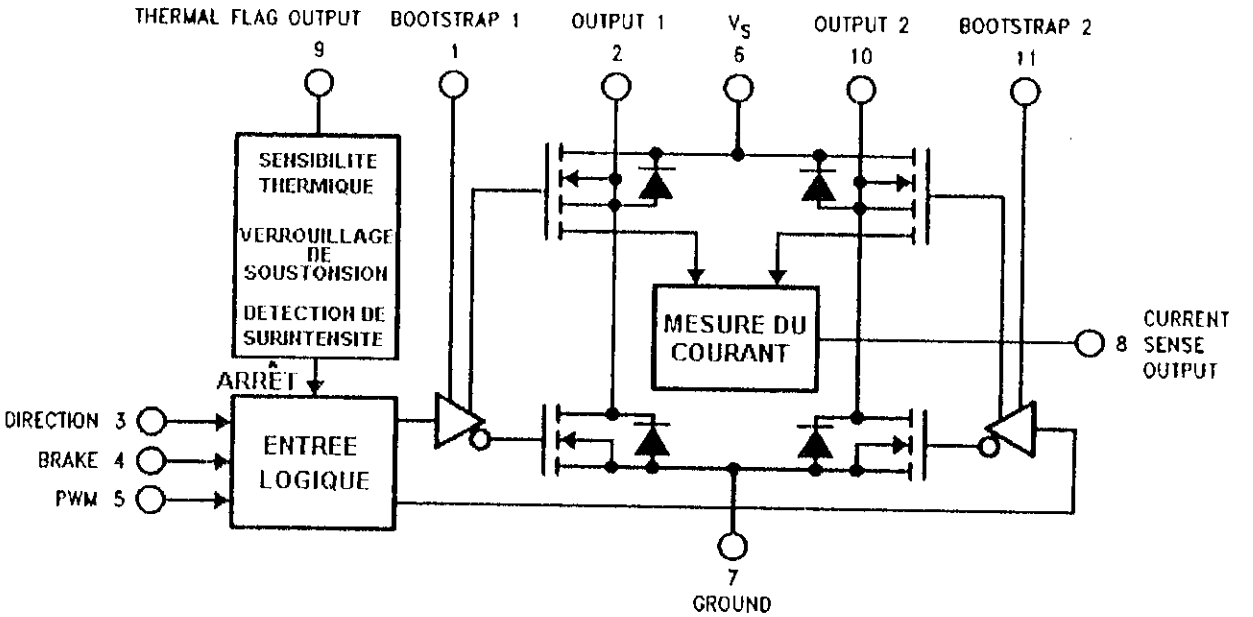



**Figure I.22** : Schéma interne du LMD 18200

Comme le montre la Figure I.22, le LMD18200 possède 11 broches, les fonctions de ces broches sont :

• **Pin 1 et 11, BOOTSTRAP :** Ce sont des entrées où des condensateurs devront être branches, ce qui est nécessaire pour le fonctionnement du circuit

• **Pin 2 et 10, OUTPUT1 et OUTPUT2 :** ces sorties forment toutes les deux le signal de commande que doit recevoir le moteur en modulation de largeur des impulsions PWM.

• **Pin 3 et 5, DIRECTION et PWM :** Ces entrées forment toutes deux le signal de commande à amplifier.

• **Pin 4, BRAKE :** Cette entrée digitale est utilisée pour l'arrêt instantané de l'amplification en cas de nécessité, réellement cette entrée est branchée directement avec la sortie AENA de la DSPGATE pour que l'ordre d'arrêt soit donné par le CPU.

• **Pin 6 et 7, Alimentation électrique :** ces entrées sont utilisées pour l'alimentation électrique de puissance, car si la tension d'alimentation de ce circuit serait Vs, alors le circuit délivre un signal en PWM entre Vs et -Vs.

• **Pin 8, CURRENT SENSE :** cette sortie est nécessaire pour la fermeture de boucle du courant, car elle fournit la mesure de courant dans une tension proportionnelle à ce courant.

• **Pin 9, THERMAL FLAG :** cette sortie qui s'active à la température 145°C, est un indicateur de dépassement de la température acceptable pour l'environnement du circuit.

### I.6.2.2 Fonctionnement de la boucle :

Etant présent le courant de référence sur la sortie DAC de la DSPGATE, elle va se comparer avec le courant qui circule dans le moteur (Figure I.23), la différence passe par un filtre PI (Proportionnelle Intégrale).

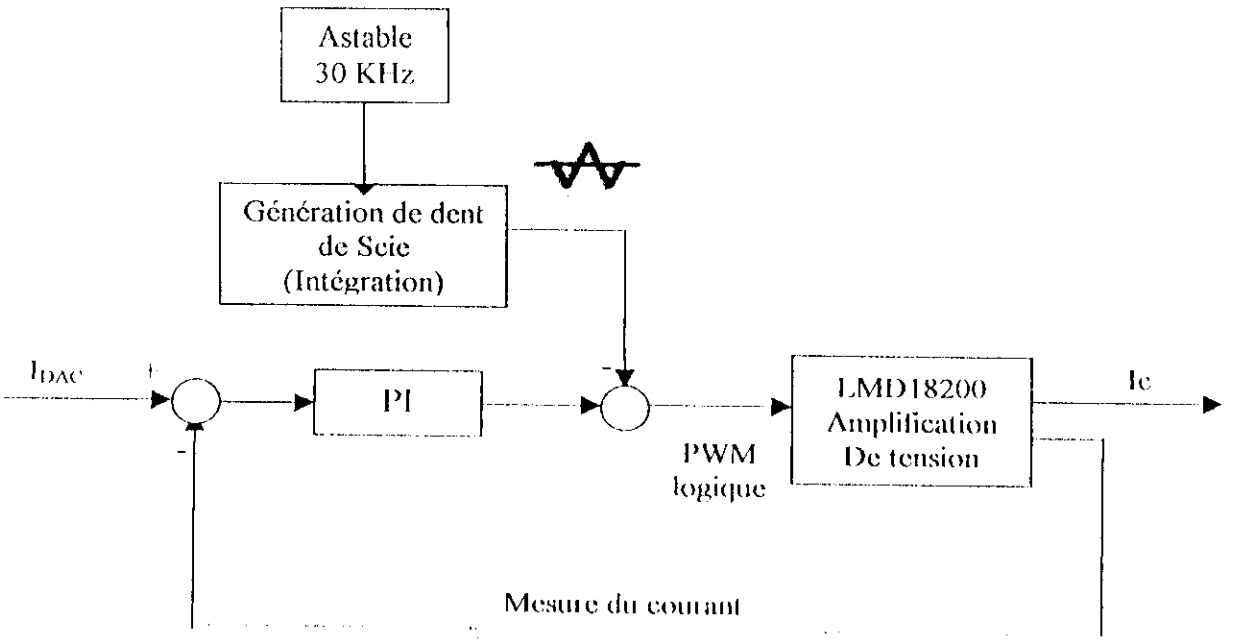



**Figure I.23** : Schéma fonctionnel de la boucle de courant

La sortie du filtre va se comparer avec un signal en dent de scie de 30 KHz qui est générée séparément de la boucle et dépendamment de la tension d'alimentation, pour donner un signal de comparaisons logique, la Figure I.24 illustre cette opération.

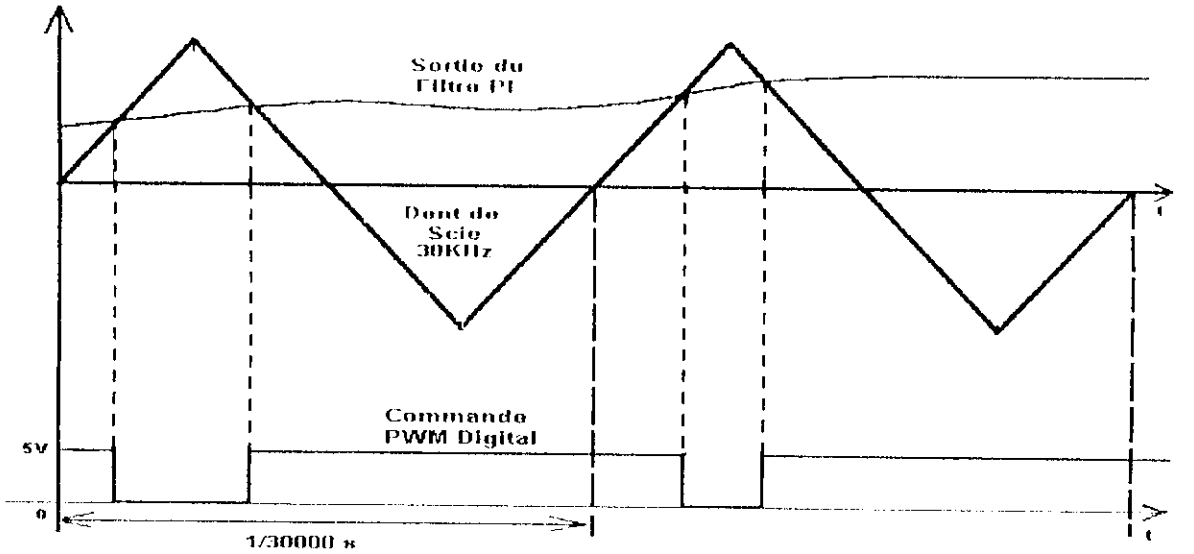



**Figure I.24** : Présentation du principe de génération de la commande PWM

Le signal résultant de comparaison vas être utilisé comme une entrée DIRECTION du circuit LMD18200 et l'entrée PWM sera mise à état haut, ce qui va donner sur la sortie OUTPUT1 le même signal amplifie et sur output2 le signal inversement amplifie, il en résulte la tension Vo1-Vo2 sur les bornes du moteur à commander, ceci est illustre par la Figure I.25.

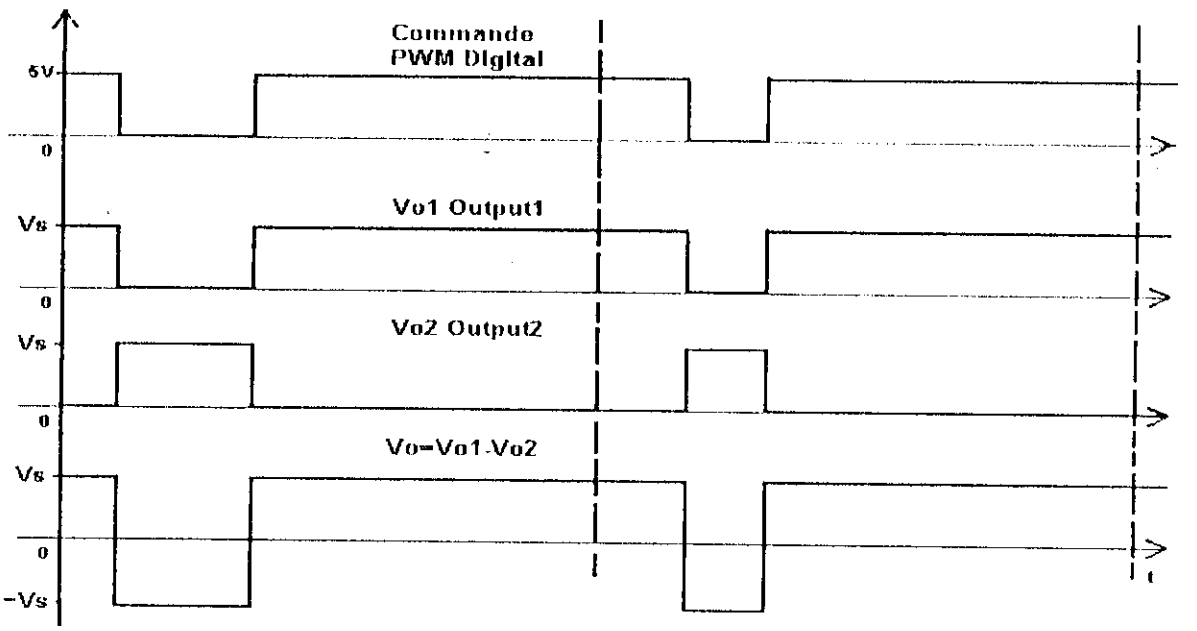



**Figure I.25** : Amplification de la PWM digitale

### I.6.2.3 Gain courant/tension :

La transconductance (ou le gain courant / tension) de ces amplificateurs est par défaut 0.5A/V pour une tension qui ne dépasse pas ± 10V ce qui veut dire que le courant peut atteindre une valeur de 5A, ceci dit ont peut réduire la transconductance à 0.25A/V (Annexe 1).

## I.6.3 ALIMENTATION ELECTRIQUE ET CONDITION DE FONCTIONNEMENT :

L'alimentation électrique aux circuits numérique est de 15V fournie par l'intermédiaire de l'UBUS, et l'alimentation électrique nécessaire pour l'amplification est comprise entre 15V et 40V fournie par l'intermédiaire des lignes d'entrée BUS+ et BUS, le courant ne peut pas excéder 12A (3 par axe) en moyenne et 20A (5 par axe) en pique, une installation des fusibles de 5A a action lente est mise en place pour protéger les shunts (faible résistance), et des fusibles de 15A a action rapide pour une protection des lignes d'alimentation.

La température opérationnelle de ces amplificateurs est de 0°C à 55°C, et la température de stockage et de -12°C à 82°C pour un pourcentage d'humidité de 0% à 95%.

## I.7 BUS DE CONNECTIONS (UBUS) [12] :

Le désir d'avoir une meilleure flexibilité lors de la conception d'une application de contrôle de mouvement complète a causé le développement de l'UBUS ( UMAC BUS ), c'est un protocole de bus parallèle d'architecture ouvert, et qui est conçu comme interface simple et robuste pour câbler les différents dispositifs (module) en l'utilisant comme une surface arrière commune et familière (standard pour les produits DELTA TAU), cette interface crée un mécanisme uniforme par lequel un processeur peut contrôler toute les cartes qui sont dans le même système.

Cette norme de bus est décrite à **ANSI/IEEE STD1014-1987** elle à des caractéristiques semblables a la norme de VME, à l'acception de quelques descriptions de broches qui sont différentes, ce qui fait que l'UBUS est non compatible avec la norme de bus VME.

Les connecteurs des cartes sont de modèle **DIN41612** à 96 lignes, le nombre de slot désigne le nombre de connecteurs de ce type (indépendamment de l'alimentation électrique) qui sont présents sur l'UBUS, pour notre cas ont dispose de deux UBUS un pour connecter les amplificateurs avec quatre slots et un pour connecter le CPU, les DSPGATEs et IOGATE avec six slots comme le montre la Figure I.26.

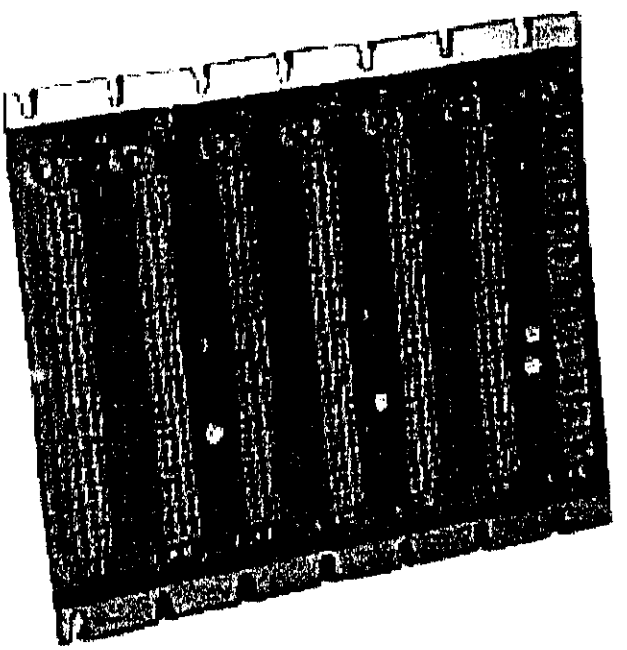


**Figure I.26** : Image de l'UBUS

### I.7.1 ORGANISATION DE L'UBUS :

L'UBUS est capable de transporter les tensions ± 5V et ± 15V continue aux modules attachés, et il est conçu pour un bus de donner de 24-bit, 15-bits de bus d'adresses, 9 lignes de sélection, 4 lignes de configuration, l'adresse, les données, et les lignes de commande sont groupées dans le Tableau I.2 comme suit:

| Lignes d'Adresse | Lignes de Sélection | Lignes de Donner | Lignes de Control | Lignes d'Horloge | Lignes de Sélection du slot |
|---|---|---|---|---|---|
| BA00(A0)<br>BA01(A1)<br>BA02(A2)<br>BX/Y(A3)<br>BA03(A4)<br>BA04(A5)<br>BA05(A6)<br>BA06(A7)<br>BA07(A8)<br>BA08(A9)<br>BA09(A10)<br>BA10(A11)<br>BA11(A12)<br>BA12(A13)<br>BA13(A14) | CS2-<br>CS3-<br>CS4-<br>CS10-<br>CS12-<br>CS14-<br>CS16-<br>MEMCS0-<br>MEMCS1- | BD00 - BD23 | BRD-<br>BWR-<br>WAIT-<br>RESET<br>IREQ1-<br>IREQ2-<br>IREQ3-<br>PWRGUD | SERVO+<br>SERVO-<br>PHASE+<br>PHASE- | BS0 - BS3 |

**Tableau I.2** : Organisation de l'UBUS

**Remarque :** La description d'UBUS est appliquée seulement pour un seul connecteur, ce connecteur est commun pour toute les cartes.

**a) lignes d'adresse :** Il existe 15 lignes d'adresse (BA00-BA13 et BX/Y) qui correspondent aux lignes A00–A14, la mise en place de BX/Y à A3 fait la sélection du champ mémoire X ou Y, cet arrangement de segmentation est typique pour les processeurs 24-bit qui sont capables de charger de mots doubles contenant des données de largeur 48 bit.

**b) lignes de sélection :** Les lignes CS2, CS3, CS4, CS10, CS12, CS14, CS16, MEMCS0 -, MEMCS1 -, sont utilisées pour sélectionner des zones mémoires bien définies, par rapport à la DSP, ces lignes correspondent à des lignes d'adresse ou d'attribution d'adresse.

**c) lignes de données :** Les 24 lignes de données bi-directionnelles sont BD00-BD23, où le BD00 est le LSB et le BD23 est le MSB.

**d) lignes de control :** Ces lignes sont employées pour déclencher les processus fournis sur l'UBUS.

• **lignes de commande BWR-, BRD - :** Quand ces signaux sont activés par le processeur, l'UBUS signal que le processeur applique une lecture ou une écriture, à l'arrêt de cet état bas de ce signal, les données sont verrouillées dans le processeur en cas de lecture (BRD -) ou dans les registres des autres cartes en cas d'écriture (BWR -).

• **ligne de commande WAIT- :** Quand les signaux BRD- et BWR- sont activés, ils peuvent ne pas être affirmés par une autre carte du système qui active le signal WAIT- qui laisse les signaux BRD- et BWR- en attente, et quand les transactions de lecture ou d'écriture sur les cartes sont possibles, le signal WAIT - est désactivé.

• **lignes de commande IREQ1 -, IREQ2 -, IREQ3- :** Ces lignes sont activées par une autre carte pour produire une demande d'interruption au processeur.

• **ligne de commande RESET :** Cette ligne est active par le processeur quand il exécute une initialisation interne, où quand le WATCHDOG TIMER est appliqué.

• **ligne de commande de PWRGUD :** Ce signal est activé par le processeur pour indiquer que l'alimentation d'énergie est présente, si une alimentation d'énergie chute, cette ligne est désactivée.

**e) lignes d'horloge Phase+, Phase -, Servo+ et Servo- :** Ces lignes sont utilisées en tant qu'interruption en temps réel, qui tiennent compte de la coordination des processus entre le processeur et les accessoires connecte à l'UBUS. Ces signaux sont fournis comme deux paires différentielles. Les fréquences des signaux d'horloge sont approximativement 10 kHz pour le signal PHASE est de 2,5 kHz pour le signal SERVO.

**f) lignes de Sélection du slot BS0 - BS3 :** les lignes BS0 - BS3 sont utilisées pour adresser les slots de l'UBUS, ces slots sont numérotés séquentiellement du slot 1 au slot 16 (maximum) du bas ver le haut.

Ces lignes, une fois choisies, composent une valeur de 4 bits à partir de 0 à 15 qui est un numéro d'identification de chaque slot.

## I.8 ALIMENTATION ELECTRIQUE [14]:

Cette une carte qui fourni une alimentation à découpage électrique stabiliser, de ±15V pour les circuits analogiques avec un courant maximal de 1.5A et 5V, et pour les circuits numériques un courant maximal de 14A, connecter à l'UBUS elle fournit l'alimentation nécessaire pour tout cartes, de plus elle fournit deux masses l'une pour les circuits analogiques AGND (*Analogique Ground*) et l'autre pour les circuit numérique GND, ces deux lignes peuvent être rassembler pour une masse commune par configurée. La Figure I.27 montre l'image de la carte d'alimentation électrique de l'UMAC.

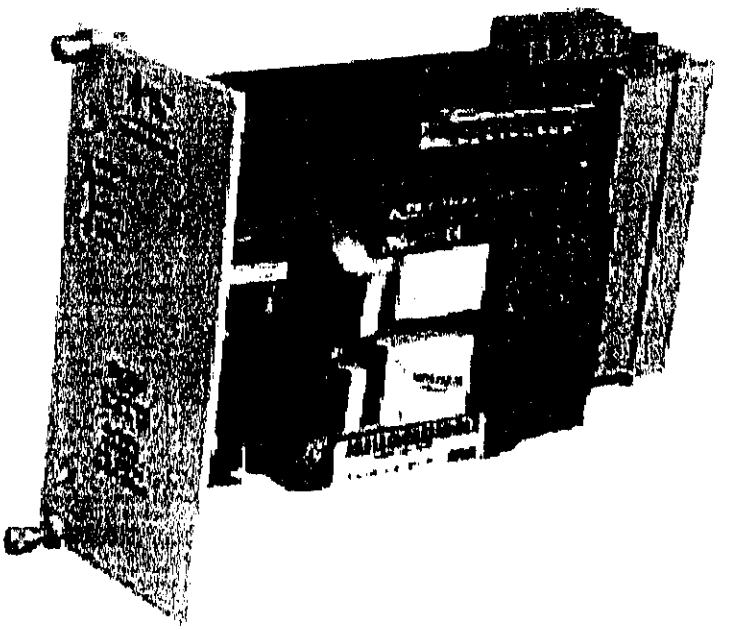


**Figure I.27** : Carte d'alimentation électrique de l'UMAC

## I.9 CONCLUSION :

De ce qui précède, on voit bien que la disposition matérielle de l'interface UMAC est idéale pour concevoir des applications de commande de mouvement, sa conception autonome montre la puissance qu'elle peut atteindre au niveau de l'exécution simultané des nombreuses taches et la flexibilité de se placer dans différentes situations. Reste à l'exploiter convenablement, et pour cela une bonne connaissance du logiciel est requise.

# *CHAPITRE II :*
# *Description du langage de programmation de l'UMAC*

## II.1 INTRODUCTION :

Pour la gestion de n'importe quelle interface on à besoin d'un pilote approprié fournissant un environnement permettant la configuration ainsi que la programmation de cette dernière ; L'interface que nous disposons possède un logiciel de gestion combiné de plusieurs applications dont l'utilisation est classée par ordre. On peut accéder à toutes ces applications à partir de l'exécutable principale PEWIN32Pro développer par **DELTA TAU**.

PEWIN32 nous permet de configurer, et de commander l'UMAC d'une manière très flexible en accédant à toutes ses variables internes, il est conçu comme un outil de développement interactif pour créer des applications de mouvement avec l'UMAC, il contient un éditeur de programme de mouvement et de programme de PLC. En plus, PEWIN32 contient une suite d'outils pour configurer et travailler avec UMAC, ces outils contiennent des interfaces pour les différent types de moteurs et des fenêtres pour visionner : les diverses variables internes de l'interface, les registres d'états des moteurs en temps réel (telle que la position, la vitesse et l'erreur associée).

Il est recommandé que l'utilisateur fait appel aux applications accompagnants PEWIN32Pro par ordre lors de l'installation initiale, afin de bien s'organiser lors de la configuration de l'application désirée :

- En premier lieu on cite l'application Turbo Setup Pro, c'est l'application où on peut définir l'état du fonctionnement de l'interface associé aux accessoires disponibles, ainsi que la configuration des paramètres d'une manière interactive toutes en visualisant les différentes valeurs affectées à ces derniers.

- En second lieu PMAC Tunig Pro, elle est utilisé pour la configuration et l'amélioration des différents paramètres des boucles d'asservissement, avec laquelle on peut optimiser les gains numériques du régulateur PID de la boucle de position/vitesse implémentée dans l'interface, tout en visualisant les réponses des moteurs, et possibilité d'introduire les paramètres des filtres utilisés ainsi que le calibrage de l'offset.

- Pour avoir les profils de la position, la vitesse, l'accélération et l'erreur associée a la position commandée, de n'importe quelle trajectoire programmée, l'application PmacPlot

Pro soit utilisé, elle permet l'acquisition des données et la présentation des profils sous forme de graphe.

## II.2 VARIABLES DE TRAVAIL [4], [5] :

Pour permettre à l'utilisateur de travailler dans un environnement adéquat et sûr, DELTA TAU a crée ses logiciels en mettant en point des variables qui permet de programmer et de configurer des applications sans avoir a s'encombrer avec les spécifications des adresses, ces variables ont chacune leur fonction et se devisent en quatre classes :

- **Variables I :** Permet d'accéder au registre de contrôle indirectement et de configurer l'interface pour les applications désirées.
- **Variables P :** se sont les variables utilisées pour le traitement des données (variables globales).
- **Variables Q :** se sont des variables utilisées comme des arguments pour les fonctions (variables locales).
- **Variables M :** Permet de pointer des adresses mémoires.

La disponibilité de ces variables permet aussi la séparation entre données et adresses, et entre registre de contrôle et registre de données, ce qui éloigne toute confusion.

### II.2.1 VARIABLES I :

Les Variables I (variables d'installation ou initialisation) déterminent la personnalité de la carte mère de l'UMAC (c.a.d la carte Turbo PMAC2) et ses accessoires. De ce faite, pour une application donnée, il est nécessaire de les configurer correctement, Ils sont dans des endroits fixes dans la mémoire Flash, et ils sont des significations prédéfini. Notons que l'interface est configuré déjà par défaut pour être satisfaisante dans les types d'application les plus communs. La plupart sont des valeurs de nombre entier (leur gamme change selon la particularité de la variable), comme ils peuvent être le résultat d'une expression.

Les variables I sont classé par catégorie pour un totale de 8191, dont on peut citer les gammes les plus importantes du point de vue de l'utilisateur qui sont :

- I0 - I99 : variable d'installation globale de l'interface.
- Ixx00 – Ixx99 : configuration du moteur xx pour les applications désirées.
- I7mn0 – I7mn9 : configuration du canal n de la DSPGATE numéros m.

(Pour plus de détail sur la spécification de chaque variable consulté le document [7]).
On peut accéder à ces variables directement par le biais de la fenêtre Terminale disponible dans le PEWIN32, par exemple si on veut avoir la valeur actuelle d'une variable quelconque on écrit simplement la ligne de commande I{constant}<RC> dans cette fenêtre, et l'interface retournera la valeur qui correspond à cette variable. Par contre si nous voulons la modifié on écrit I{constant}={constant}<RC>, ou si on veut affecter sa valeur par défaut on écrit I{constant}=* (pour tout un rang de variables on fait I{constant}..{constant}=*), après avoir affecter une nouvelle valeur a une variable, il faut la sauvegarder par la commande SAVE dans le terminale.

Pour les applications particulières, il n'est pas nécessaire de chercher toute ces variables car le logiciel communique avec l'interface « Turbo Setup Pro » nous fournie les étapes de configuration préliminaire adéquate pour chaques applications, toute en visualisant les valeurs affectées aux variables a chaque étape, (préliminaire car une configuration supplémentaire faite par l'utilisateur est nécessaire).

### II.2.2 VARIABLES P :

Les variables P, comme les variables Q, sont des variables d'usage universel pour l'utilisateur, ils sont des variables réels stockées dans des cases mémoires dans l'interface sur 48-bit, sans une utilisation pré-define, pour un totale de 1024 variables.

Toutes les variables P sont accessible par les systèmes en coordination (contrairement au variables Q qui sont couplé avec), ceci tient compte de l'information utile passant entre les différents systèmes en coordination.

Si une constante est envoyée à l'interface comme une commande, elle sera affecter à la variable P0 ; par exemple, si nous envoyons la commande 342<CR>, l'interface l'interprétera en tant que P0=342<CR>. Par conséquent, il n'est recommandé d'employer P0

pour d'autres buts que ce dernier, parce qu'il est facile de la changer accidentellement. systèmes en coordination pour 128 variables chacun), la gamme pratique des variables Q à employer sans risque dans les programmes de mouvement est donc Q1 à Q99.

### II.2.3 VARIABLES Q :

Les variables Q, sont des variables d'usage universel de 48-bit sans fonction pre-définie. Cependant, la signification d'une variable Q donné dépend du quel système en coordination (décrit si dessous) elle est utilisée.

Ceci permet à plusieurs systèmes en coordination d'employer le même programme (par exemple, contenant la ligne X(Q1+25) Y(Q2)) sans pour autant s'inquiéter du mélange de leurs valeurs. Plusieurs variables Q ont une utilisation particulière, comme par exemple :

- La fonction ATAN2 utilise automatiquement Q0 comme deuxième argument.
- La commande READ place les valeurs qu'elle lit après les lettres A à Z dans Q101 à Q126, respectivement.
- La commande S dans le programme de mouvement place sa valeur dans Q127.

Basé sur ce qui précède, et puisqu'un totale de 1024 variables Q (qui sont partagés entre 8 potentiels

### II.2.4 VARIABLES M :

Les variables M sont fournies pour permettre un accès facile de l'utilisateur à la mémoire interne de l'interface et aux registres d'entrées/sorties. Ils sont affectés à des locations mémoires dont l'utilisateur peut définir la taille (en bits), le format ( virgule fixe ou flottante ), et la valeur attribuée à cette location. D'une façon générale, une définition d'une variable M doit être faite une fois seulement par une commande en ligne (la commande SAVE doit être utilisée pour la sauvegarde de la définition).

Comme les autres variables, elles peuvent être attribuées par une constante ou une expression : M576 ou M(P1+20). La définition d'une variable M est faite en utilisant le symbole '' -> '', elles peuvent prendre un des types de préfixe suivants :

- X : 1 a 24 bits virgule fixe dans la mémoire X
- Y : 1 a 24 bits virgule fixe dans la mémoire y
- D(double) : 48 bits virgule fixe à travers les deux mémoires X et Y.
- L(long) : 48 bits virgule flottante a travers les deux mémoires X et Y.
- *: Pas de définition d'adresse ; utilisation comme variable générale.

La définition d'une variable M est faite par une définition du bit de début, le nombre des bits et le format. On cite si dessous un exemple de définition typique des variables M :

M103->X:$C003,0,24,S
M161->D:$002B,U

Où $ : adresse spécifié en hexadécimale.
S : signifie que le nombre est signé.
U : spécifie que le nombre est non- signé.

Par la suite les définitions des variables M sont stockées dans des cases mémoires situées dans le champ Y à l'intervalle Y:$BC00 (pour M0), Y:$BFFF (pour M1023), les 16 bits de poids faible contiennent l'adresse du registre pointé par la variable M, et les 8 bits du poids fort indiquent quelle partie de l'adresse est employée et comment l'interpréter.

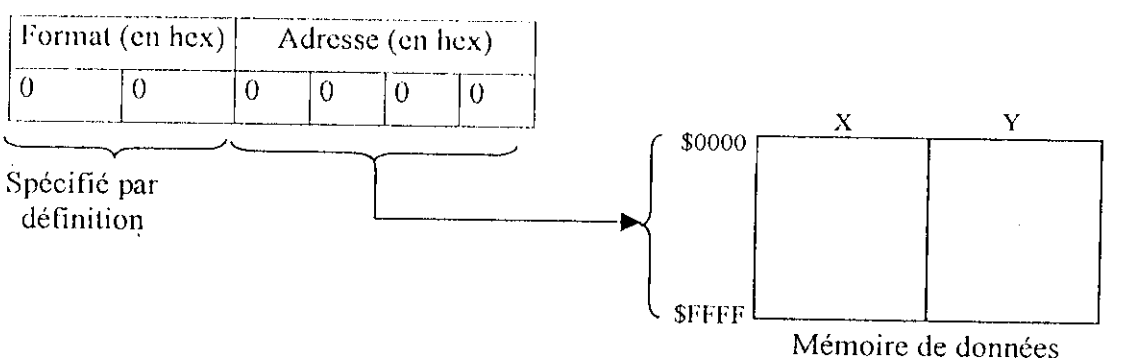



**Figure II.1 :** Format de la définition d'une variable M

Une fois définie, une variable M peut être employée dans les programmes justes comme n'importe quelle autre variable – par des expressions. Quand l'expression est évaluée, le

processeur lit l'endroit de mémoire défini, calcule la valeur basée sur la taille et le format définis, et l'utilise dans l'expression.

La prudence dans utilisation des variables M est exigée, car celle çi peut être changée sur l'évaluation d'une expression au cours de l'exécution d'un micro-programme.
Par exemple, si dans l'expression « (M16-M17)*(M16+M17) » les variables M sont lus instantanément de la mémoire, il n'est pas sûr que M16 ou M17 auront la même valeur pour les deux termes dans l'expression précédente, ce problème peut être surmonté en affectant P1=M16 et P2=M17.

## II.3 LES SYSTEMES EN COORDINATION [4], [5] :

Un système en coordination est un groupement de moteurs élaboré afin d'avoir la même synchronisation du mouvement. L'exécution d'un programme de mouvement nécessite la définition des systèmes en coordination (même avec un moteur seulement). Avec l'UMAC on peut avoir jusqu'à huit systèmes en coordination (extensible jusqu'à 32), adressés par &1 jusqu'à &8, avec un mode très flexible (exemples : 8 systèmes en coordination contenant un moteur chacun, un système en coordination contenant 8 moteurs, 4 systèmes en coordination contenant 2 moteurs chacun, ... etc).

En générale si on veut que certains moteurs bougent en mode coordonnée, on les met dans le même système en coordination, et si on veut les faire bouger indépendamment les un des autres, on les met dans des systèmes en coordination séparés.

Les systèmes en coordination peuvent exécuter diffèrents programmes de mouvement indépendamment, mais aussi d'exécuter le même programme à des moments différent ou simultanément. Un système en coordination doit être d'abord établit en affectant des axes aux moteurs dans un rapport de définition d'axe (décrit çi dessous), cependant, il doit y avoir au moins un moteur attaché à un axe, sinon il ne peut pas exécuter un programme de mouvement.

## II.4 PROGRAMME DE MOUVEMENT [4], [5] :

C'est un programme qui contient des informations et des lignes de commande où le processeur de l'UMAC les utilisent pour planifier et déclencher le mouvement, l'interface peut supporter jusqu'à 256 programmes de mouvement en même temps (avec extension de 7 autre cartes Turbo PMAC2- 3U). N'importe quel système en coordination peut exécuter n'importe lequel de ces programmes à tout moment, même si un autre système en coordination exécute déjà ce même programme.

Un programme de mouvement peut appeler n'importe quel autre programme de mouvement comme sous-programme, avec ou sans arguments. Le langage de programme de mouvement de UMAC peut-être mieux décrit comme langage un couplage entre un langage de programmation évolué (comme le BASIC ou le Pascal), et de langage machine-outil (G-Code ou D-Code).

### II.4.1 DEROULEMENT DE L'EXECUTION DES PROGRAMMES DE MOUVEMENT DANS UMAC :

Fondamentalement, un programme de mouvement existe pour que le processeur fait passer les données de ce dernier vers les routines de générateur de trajectoire qui calculent la série de positions de référence commandée pour les moteurs à chaque cycle Servo, donc un programme de mouvement doit fonctionner en tête du mouvement commandé actuel pour maintenir l'alimentation des générateurs de trajectoires avec les données.

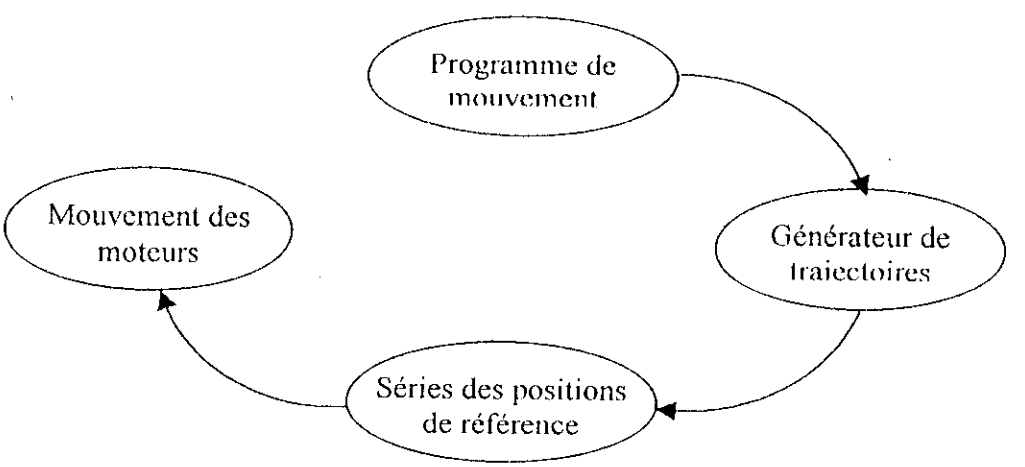



**Figure II.2 :** Principe d'interprétation de programme mouvement

Le processeur traite les lignes de programme de façon à ce qu'un ou deux mouvements soient calculés en avance, ce qui est nécessaire afin de pouvoir composer des mouvements (pour de prévoir les limitations appropriées d'accélération et de vitesse qui doivent être faites), et puisque c'est le cas pour les mouvements linéaires composés (I13 doit être égale à 0 : temps de segmentation de mouvement nul) où le processeur calcule deux mouvements à l'avance, et dans tous les autres cas (I13 > 0) il calculera un seul mouvement à l'avance.

**Remarque :** S'il y'a une singularité au cours du calcule de trajectoire, par conséquent le processeur ne peut pas finir le calcule, il abandonnera le programme avant que l'exécution de ce mouvement soit censée commencer, en envoyant une erreur d'exécution temporelle à l'ordinateur principale.

### II.4.2 DEFINITION D'AXE :

Un axe de point de vue logiciel est un élément important des systèmes en coordination, il se réfère à un moteur par une lettre alphabétique sélectionnée parmis X, Y, Z, A, B, C, U, V, et W de telle façon à ce qu'une affectation d'une valeur à cette lettre corresponde à un déplacement du moteur.

Un axe est défini en l'affectant à un moteur avec un facteur de graduation et un offset (notons que X, Y, et Z peuvent être définis en tant que combinaisons linéaires de trois moteurs, de même que pour U, V, et W).

La plus simple définition est : #1->X (attaché le moteur #1 à l'axe X pour le système en coordination courent), maintenant si l'axe X exécute un mouvement, le moteur #1 fera le mouvement. Pour affiner une application (de point de vue réglage et précision) il faut définir les caractéristiques d'axe suivantes :

- Graduation d'axe.
- Offset.
- Type d'axe.

### II.4.2.1 Graduation d'axe :

Dans cette partie, on définit une nouvelle unité d'utilisation (en se basant sur un certains nombres d'incrémentation de l'encodeur) dont elle sera la nouvelle résolution de déplacement du moteur.

**Exemple :** Nous disposons d'un encodeur de 500 CPR (compte par révolution) avec un moteur dont le rapport de réduction est $\eta = 65.5$. Etant donné que la résolution est augmentée de 4 fois par l'interface (suivant la configuration de I7mn0), ce qui signifie qu'on a 2000 compte par tour d'encodeur, alors une rotation complète de l'arbre du moteur corresponde à : $\eta * 2000 = 131000$ comptes.
Ce qui fait que l'affectation « #1->131000X » permet de travailler avec une unité de déplacement du moteur #1 égale à un tour.

### II.4.2.2 Offset :

Par exemple le rapport #1->10000X+20000 place un offset (de 2 unités d'utilisateur) de la position zéro du moteur #1, notons que cet offset est rarement utilisé.

### II.4.2.3 Type d'axe :

Pour la définition de ce type d'axe, il peut y avoir deux attributs géométriques où chacune correspond à un mode de mouvement dans l'espace, le troisième type est une propriété :

- Les axes cartésiens.
- Les axes de rotation.
- Les axes fantôme.

Notons que pour la plupart des applications, le choix du type d'axe n'est pas nécessaire. Cependant, pour certaines applications comme l'implémentation du modèle géométrique d'un robot, on ne peut utiliser que les noms particuliers X, Y, Z, U, V et W qui correspondent aux axes cartésiens.

**a) Axes cartésiennes :**

Ce type d'axe peut être groupé suivant une combinaison linéaire de deux ou trois axes pour causer le mouvement de deux ou de trois moteurs : X, Y, Z forment la première base, U, V, W forment la seconde.

**Exemple :** #1->8660.25X-5000Y ; avec : cos (30°) = 0.866 et sin (30°) = 0.5.
#2->5000X+8660.25Y

Dans ce cas le mouvement d'un seul axe X ou Y cause le mouvement des deux moteurs avec un écart final de 30° entre les deux axes des moteurs.

**Remarque :** On ne peut avoir d'autres combinaisons entre les axes. C'est à dire non-linéaire, cependant, les relations non- linéaires qui peuvent exister entre les axes se feront dans des programmes spéciaux. (Voir à la dernière section de ce chapitre).

**b) Axes de rotation (A, B, et C) :**

Quand on affecte un moteur a ces axes, la fonction *rollover* du moteur est activée, et quand le programme fonctionne en mode absolue (ABS), cette fonction permet au moteur de prendre la plus courte distance du point de départ au point destiné au tour d'un rang spécifié par Ixx27.

**Exemple :** I127 = 131000 compte par tour
#1->363.889A ; A est dans les unités des degrés

Dans ce cas, si on applique la commande de mouvement 'A270' dans un programme en mode ABS, le moteur fait le mouvement de 0 à -9000 (-90°), au lieu de faire le mouvement de 0 à 27000 (ce qui l'aurai fait avec I127 = 0).

**Remarque :** Contrairement aux axes linéaires, ces axes ne peuvent pas être dans une combinaison linéaire.

**c) Axes fantôme :**

Un axe peut n'avoir aucun moteur attaché ''un axe fantôme'', bien que les mouvements programmés pour cet axe ne causent aucun mouvement, le fait qu'un mouvement a été programmé pour cet axe, il puisse affecter les mouvements d'autres moteurs. Par exemple, si on désire d'avoir des profils sinusoïdaux sur un axe simple, la manière la plus facile de le faire, est d'avoir en second lieu un axe fantômes et a programmer des mouvements d'interpolation circulaire (voire si dessous).

## II.4.3 ELABORATION D'UN PROGRAMME DE MOUVEMENT :

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

**1) OPEN PROG {constante} :** Ouvre un programme dans un buffer sur l'UMAC, où la constante représente le numéro du programme, et qui varie entre 0 à 32767 (de 0 à $2^{15}$-1).
Les programmes de mouvement 1000,1001,1002,et 1003 contiennent les G-code, T-code, D- code pour les machines outils.
Le programme 0 assure le chargement des lignes de programmes, pour les longs programmes le chargement se fait au fur et a mesure de l'exécution de dernier.

**2) CLEAR :** Efface le contenu du programme ouvert.

**3) LINEAR, RAPID, CIRCLE, PVT, et SPLINE :** Ces commandes servent à définire les trajectoires (profil de vitesse).

**4) INC et ABS :** indiquent le mode de mouvement, la commande INC (mode incrémentale) définie un déplacement relatif à la position actuelle, la commande ABS (mode absolu) définie un déplacement absolu.

**5) Commande de mouvement :** Un mouvement est défie par la spécification d'un axe suivi d'un déplacement.
Tous les mouvements spécifiés dans une même ligne de commande seront exécutés simultanément, tandis que les lignes consécutives s'exécuteront séquentiellement.

**6) Paramètres temporels :** Si les temps de mouvement TA (temps d'accélération), TS (temps de courbure pour le mode S-Curve), TM (temps de mouvement), et/ou la vitesse F (Feedrate) ne sont pas déclaré, les valeurs par défaut sont fournies par les paramètres Ixx87, Ixx88, et Ixx89.

Cependant, il est recommandé de ne pas compter sur ces paramètres et à déclarer les temps de mouvement dans le programme. Ceci permettra une meilleure maintenance du programme.

**7) Instructions usuelles :** Dans un programme de mouvement, on peut faire appel aux instructions d'itérations condition tel que la boucle WHILE, IF..ELSE, ainsi que les commande de branchement tel que GOTO, et la commande GOSUB : qui signifie branchement au sous- broutine, on peut aussi faire appel a un autre programme avec la commande CALL à partir de n'importe quelle ligne sur ce programme, exemple : CALL 20.150 : entrer dans le programme 20 à la ligne 150.

**8) CLOSE :** ferme le buffer ouvert. Doit être utilisée immédiatement après l'entrée d'un programme de mouvement, sinon toutes commandes en ligne (écrites dans le Terminale Windows) risquent d'entrer dans le programme. Pour éviter ceci il est recommander d'inclure CLOSE au début et à la fin de chaque programme à télécharger à UMAC.

**Exemple :**

```
CLOSE                    ; ferme tous les buffers ouverts.
DELETE GATHER            ; Effacer toutes les données intermédiaires.
UNDEFINE ALL             ; Effacer les définitions lié a tous les système en coordination.
#1->  363.889X           ; lier le moteur #1 a l'axe X avec une graduation d'axe en degré
                         ; 65.5*2000 / 360 = 363,889.
OPEN PROG 1 CLEAR        ; Ouvrir le buffer pour écrire le programme.
LINEAR                   ; interpolation Linéaire (le profile de vitesse est linéaire).
INC                      ; mode Incrementale.
TA100                    ; temps d'accélération est 100 msec.
TS0                      ; pas de courbure dans le profile de vitesse.
F50                      ; le Feedrate est 50 unités par 15x90 msec.
X180                     ; 180 unités de distance : 180°.
CLOSE                    ; ferme le buffer, programme 1.
```

I5x90 : doit être initialisé à 1000 pour avoir une échelle de temps en msec pour le ssystéme en coordination x.

Le temps de mouvement TM est donné directement dans le programme, ou indirectement on se basons sur la distance de l'axe, et la vitesse désirée (feedrate).

**Exemple :**

```
TM500      ou bien      X5F10          ; Vx = 0.5µs
X5
```

**Exemple pour deux axes :**

```
TM500      ou bien      INC            ; Distance = SQRT(3² + 4² ) = 5
X3Y4                      FRAX(X,Y)   ; TM = 5/10 = 0.5 µs
                        X3  Y4  F10   ; Vx = 3/0.5 = 6
                                      ; Vy = 4/0.5 = 8
```

### II.4.4 L'EXECUTION DU PROGRAMME DE MOUVEMENT :

**1)** Choisir le système en coordination pour l'exécution du programme désiré, rappelons que pour choisir le 1$^{er}$ système en coordination par exemple, il faut taper la commande &1 dans le *Terminal windows*.

**2)** Choisir le programme a exécuter, tapez la commande B suivie du numéro du programme dans le *Terminal Windows* (se brancher au début du programme n), puis tapez la commande R pour lancer l'exécution.

Le programme s'exécutera complètement à moins qu'il soit arrêté par une commande ou une condition d'erreur (les codes des erreurs sont rapportés dans l'annexe B).

**3)** Avant de commencer l'exécution du programme, il faudrait que *tous les moteurs* dans le système en coordination soit en boucle fermée, pour ce faire taper « CTRL+A ».

En cas de doute, le fonctionnement de chaque moteur pourrait être contrôlé individuellement avant l'exécution du programme au moyen de la commande *Jog* (essai).

**4)** Un programme de mouvement pourrait être arrêté en envoyant la commande A<CR> (*Abort*) cela causerais la décélération immédiate des moteurs, ou bien la commande Q<CR> (*Quit*) cela stopra le programme a la fin de l'exécution du mouvement en court.

Si le mouvement d'un axe devient incontrôlable et qu'on souhaite arrêter, la commande K<CR> (*Kill*) neutralise les amplificateurs permettant l'alimentation des moteurs, cependant, le moteur subira un arrêt incontrôlable pouvant induire un mouvement résultant de l'inertie du moteur.

### II.4.5 LES SOUS-PROGRAMMES :

Il est possible de créer des sous-programmes, dans le but de créer des programmes modulaires bien structurés et réutilisables.

La commande GOSUBx fait un saut à l'étiquette Nx (donc au sous- programme) dans le même programme, il revient à exécuter la ligne suivante après avoir rencontrer la commande RETURN.

La commande CALLx dans un programme de mouvement cause un saut au PROG X, avec un saut de nouveau à la commande suivante immédiatement après la rencontre de la commande RETURN.

La commande READ fait placer les valeurs après les lettres dans les variables Q ( lettre A dans la variable Q101, B dans Q102 ... Z dans Q126 – excepté N et O –) d'où la possibilité d'appel les procédures avec ces arguments,

**Exemple :**

```
close delete gathor undefine all
#1->2000X
open prog1 clear
LINEAR INC TA100 TS0 F50  ; paramètres de temporisation et le mode.
gosub 100 X10             ;appel du subroutine en passant la valeur 10 a X.
return                    ; fin du programme principale.
n100                      ; la section du subroutines, étiquette 100
IF (Q100 & $800000 > 0)   ;si le paramètre de X est lu.
      X(Q124)             ;utiliser le parametre contenue dans la valeur Q124
                          ;pour l'axe X
endif
return                    ; fin du subroutine d'étiquette 100
close                     ; fin du programme de mouvement.
```

## II.5 LES TRAJECTOIRES ET LES PROFILES DE VITESSE PRINCIPAUX DU MOUVEMENT [4], [5] :

### II.5.1 LE MOUVEMENT LINEAIRE :

C'est un mouvement a un profil de vitesse linéaire, dont on peut déterminer les paramètres dans le programme :

Les temps : d'accélération TA, de mouvement TM, de courbure au bout de deux extrémités de la vitesse TS (on parlera alors du mode S-curve), et /ou la vitesse F dont la limite maximale est déterminée par le paramètre Ix16 a condition que I13 = 0 pour les mouvements linéaires mélangés.

Les figures suivantes montrent les différents cas du mouvement linéaire :

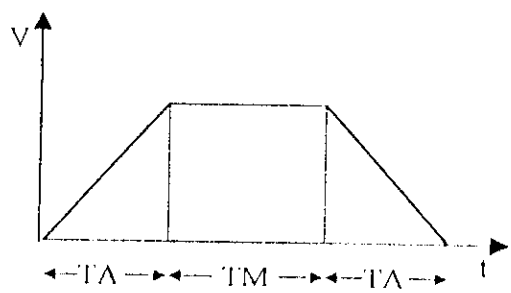



**Figure II.3.a :** Mouvement linéaire sans S- courbe

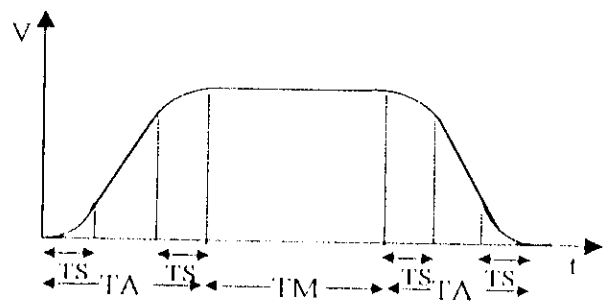



**Figure II.3.b :** Mouvement linéaire avec S- courbe

S'il y'a une erreur d'estimation de ces paramètres, un auto ajustement est fait par l'interface lors de la compilation :

Si TM < TA alors TM = TA.

Si TA < 2*TS alors TA = 2*TS.

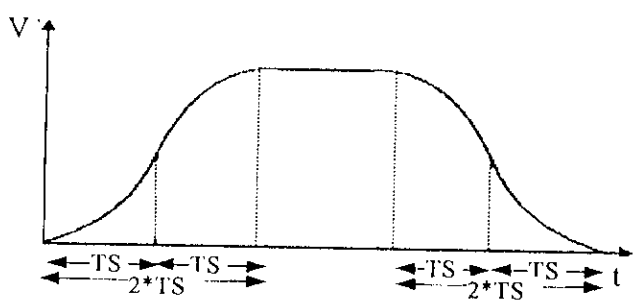



**Figure II.4 :** Auto ajustement cas où TA < 2*TS

## II.5.2 LE MOUVEMENT SPLINE :

Le principe du mouvement SPLINE est le raccordement d'une série de mouvement d'une manière cubique (l'équation de position est un polynôme du 3ème degré), pour qu'il y pas de changement brusque de vitesse ou d'accélération. Il y'a deux types :

- Dans SPLINE1 un long mouvement est coupé en segments de temps égaux TA. Chaque axe ait donné une destination, par exemple X1000Y2000 : Cette fonction fait appel a la position précédente et future et créer une courbe de position cubique pour chaque axe, de sorte qu'il n'y ait aucun changement soudain de vitesse ou d'accélération aux frontières de segment. La position commandée à la frontière du segment peut être détendue légèrement pour rencontrer les contraintes de vitesse et d'accélération.

  Au commencement et à la fin d'une série de mouvements, la fonction ajoute automatiquement un segment d'une distance zéro et de temps TA pour chaque axe, et exécute le raccordement entre ce segment et l'adjacent, ceci a comme conséquence une accélération de S- courbe au deux extrémités : début et fin de la trajectoire.

- Le mode SPLINE2 est très semblable au mode SPLINE1, sauf que le temps de segment TA est variable.

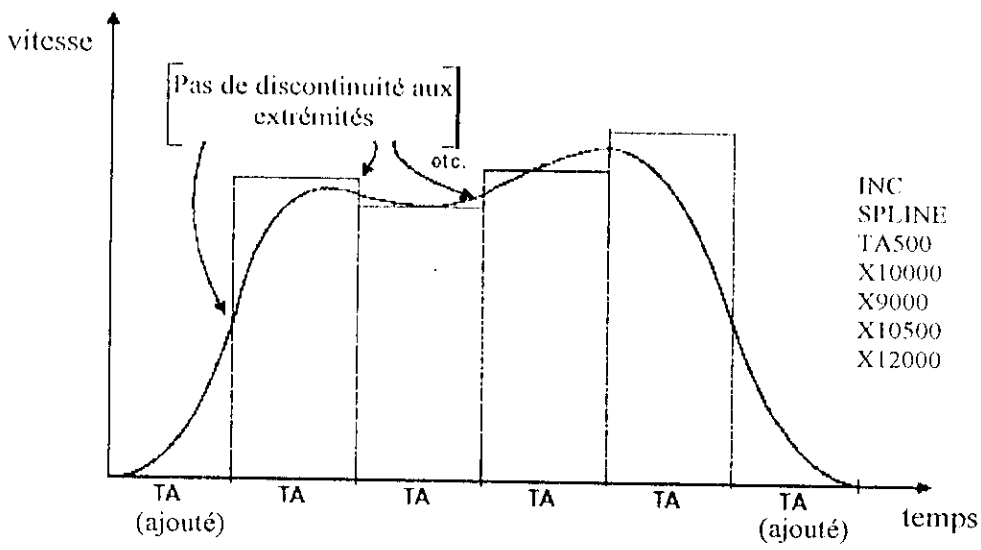



**Figure II.5 :** Le mouvement SPLINE

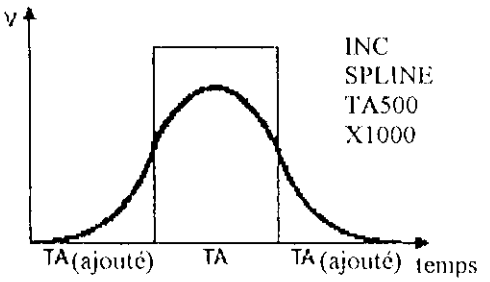



**Figure II.6.a :** Mouvement Spline programmé pour un segment

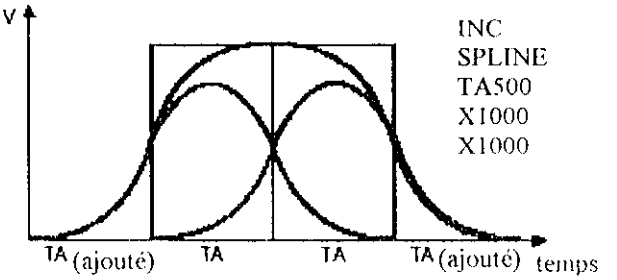



**Figure II.6.b :** Mouvement Spline programmé pour deux segments

### II.5.3 LE MOUVEMENT PVT ( POSITION- VELOCITY- TIME ) :

On utilise le mode PVT pour un contrôle plus direct du profil de la trajectoire. Dans ces mouvements, l'utilisateur indique les états d'axe directement aux transitions entre les mouvements, ceci exige plus de calcul par le processeur, mais permet une commande plus précise de la forme de la trajectoire. Pour chaque partie d'un mouvement, l'utilisateur indique la position ou la distance de fin, la vitesse de fin, et le temps de mouvement.

La syntaxe pour ce mode est de la forme suivante :

- PVT{data}, où {data} peut être une constante, une variable, ou une expression représentant le temps de mouvement en msec (cette valeur devrait être un nombre entier; Si elle n'est pas, elle est arrondie automatiquement au nombre entier le plus proche).
- {axe}{data}:{data} où {axe} une lettre indiquant l'axe, la première {data} la valeur indiquant la position de fin, et la seconde {data} représente la valeur de vitesse de fin.

**Exemple :**

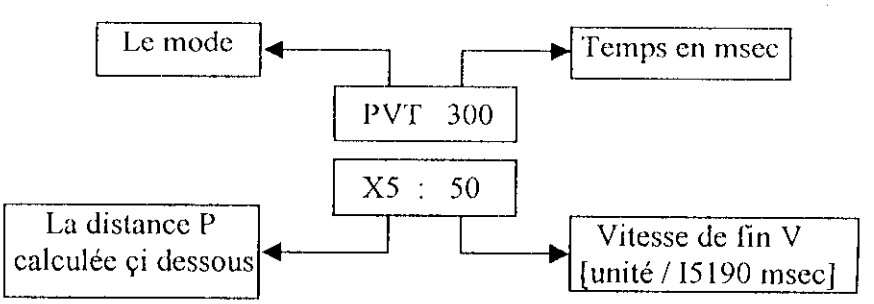



**Figure II.7 :** Exemple de la syntaxe de PVT

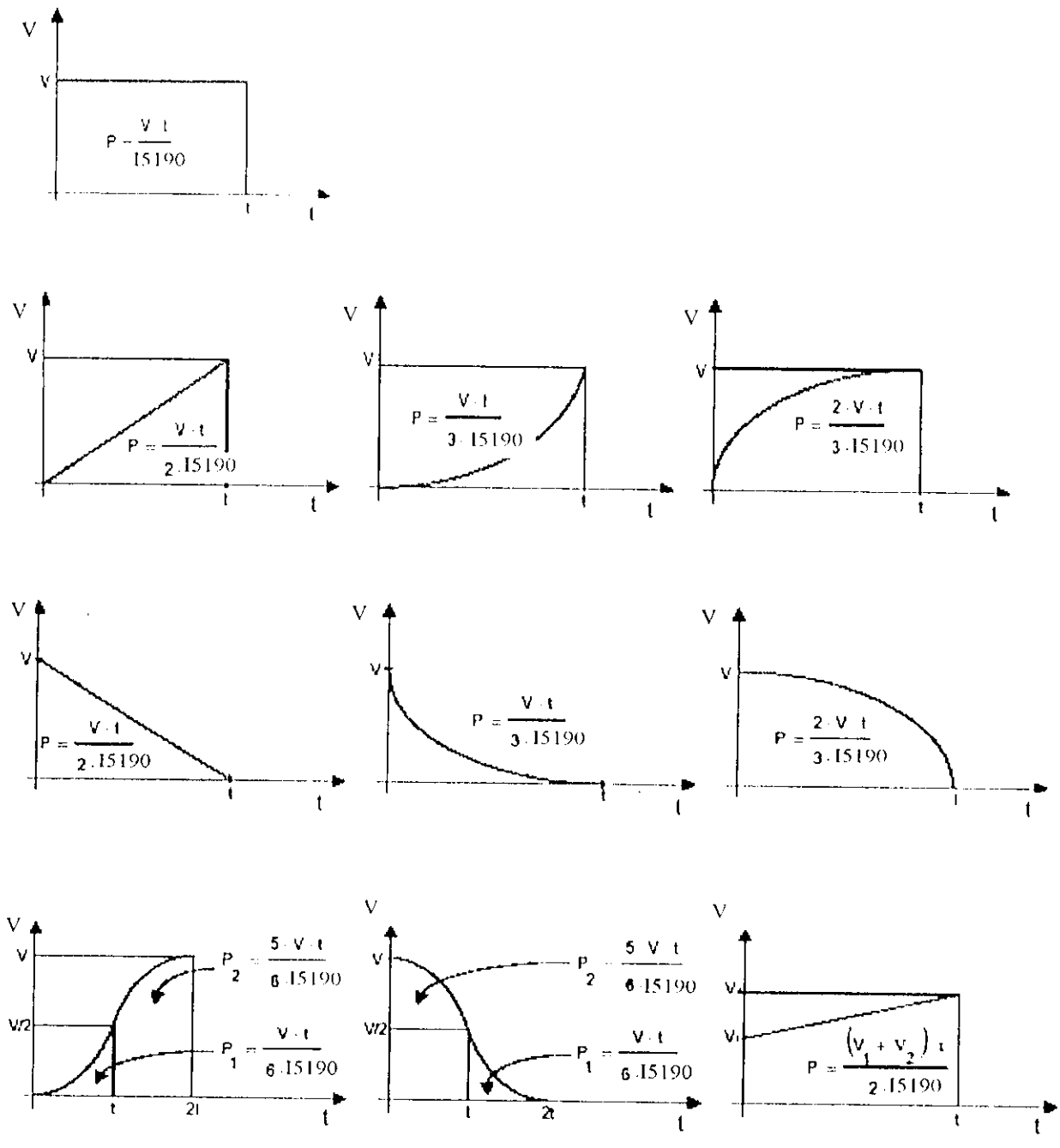



**Figure II.8 :** Méthode de calcule de la position P pour le mode PVT

**Remarque :** Remplacez 15190 pour la variable 15x90 appropriée pour déterminer la base du temps du système en coordination x.

**Exemple:**

```
CLOSE
DELETE GATHER
UNDFINE ALL
&1 #1->2000X
OPEN PROG 1 CLEAR
INC
PVT 300 ; Temps = 300 msec par section
X5 :50 ; 50 unité * 300 msec / 1190*3 = 5 unités
X5 :0 ; 50 unités *300msec / 1190 * 3 = 5 unités
CLOSE
```

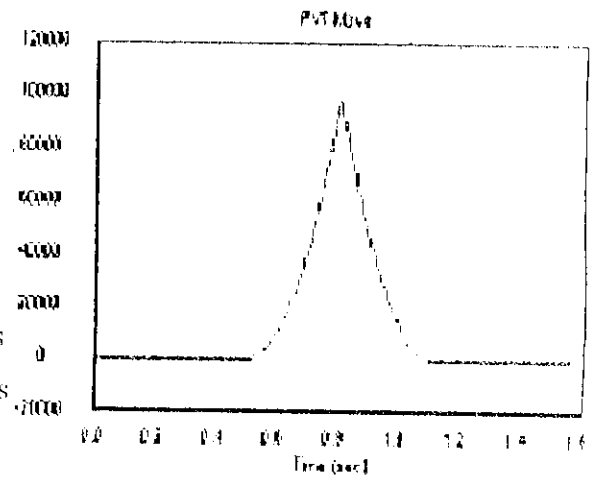
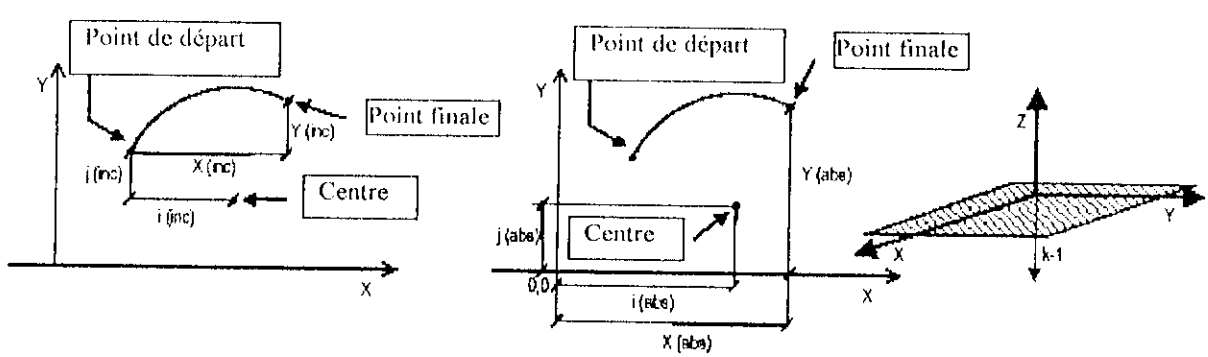


**Figure II.9 :** Simulation du programme de mode PVT

### II.5.4 LE MOUVEMENT CIRCULAIRE :

Le processeur peut effectuer des mouvements circulaires en segmentant l'arc et en exécutant le meilleur ajustement cubique sur chaque segment (I13 détermine le temps de segmentation). La gamme pratique de I13 pour le mode circulaire est 5-10 msec ; la valeur 10 msec est recommandée pour la pluparts des applications.

Pour les mouvements composés, il est possible de changer le mode de mouvement du linéaire au circulaire et vis versa dans un même programme au moyen de la commande LINEAR quand l'interpolation linéaire est nécessaire et CIRCLE1 ou CIRCLE2 pour l'interpolation circulaire.




Mode incrémental Mode absolus

**Figure II.10 :** Interpolation circulaire

Le plan de l'arc circulaire doit être défini par la commande NORMAL. Cette commande peut définir des plans seulement dans l'espace XYZ, ce qui signifie que seulement les axes X, Y, et Z peuvent être utilisé pour l'interpolation circulaire. Si d'autres axes sont indiqués dans le même programme de mouvement, ils seront interpolés linéairement dans le même temps :

NORMAL K-1 ; XY plane

NORMAL J-1 ; ZX plane

NORMAL I-1 ; YZ plane

Le centre du cercle est spécifié par le vecteur (I, J, K) = (la composante parallèle à X, Y, Z), et la position finale est indiquée par les valeurs des axes dans le programme sous un mode absolu ou incrémental (ABS ou INC).

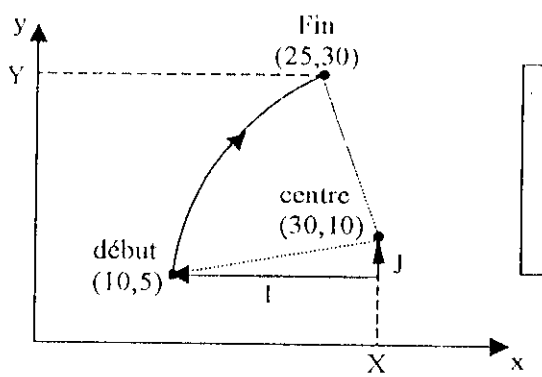
La syntaxe du mode circulaire est donnée par :

CIRCLE {1 ou 2}

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

X{Data} Y{Data} R{Data} ; pour ce cas le rayon du cercle est donné.

X{Data} Y{Data} I{Data} J{Data} ; pour ce cas le centre du cercle est donné.




```
NORMAL K-1
ABS (XY)
INC ( R )
CIRCLE1
F10
X25Y30I20J5
```

Ou bien

```
NORMAL K-1
INC
ABS ( R )
CIRCLE1
F10
X15Y25I30J10
```

**Figure II.11 :** Définition d'une trajectoire circulaire

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

CIRCLE2 : sens trigonométrique, CIRCLE1 : sens horaire.

**Remarque :** Si le rayon du cercle est positif, le mouvement prend le chemin le plus court d'arc (≤180°). Si le rayon du cercle est négatif, le mouvement prend le chemin le plus long d'arc (≥180°). Les deux centres des cercles sont symétriques par apport à la droite traversant le point de départ et le point final.

```
1)CIRCLE2
  TM2000
  X0Y10R10

2)CIRCLE2
  TM2000
  X0Y10R-10
```

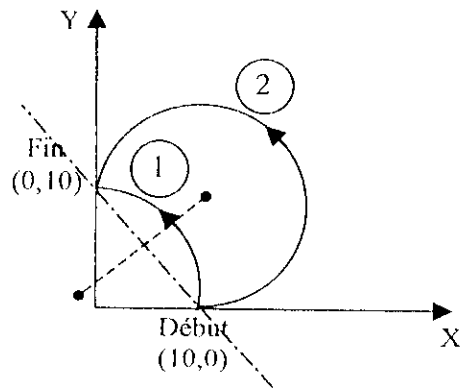
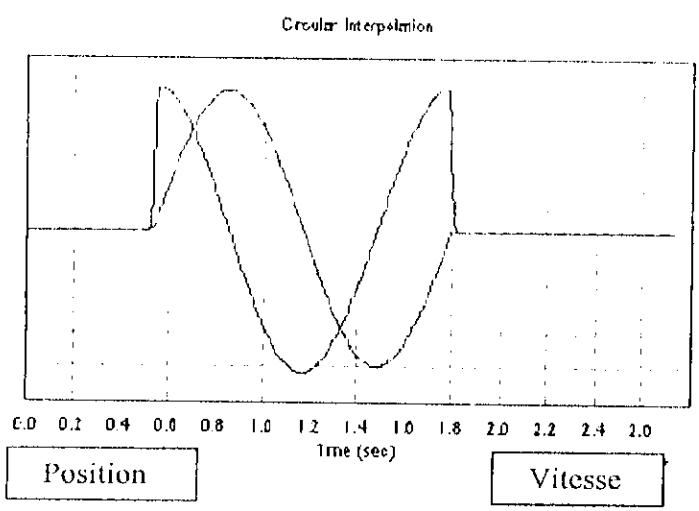



**Figure II.12 :** Sens et longueur du mouvement circulaire

On peut exploiter l'interpolation circulaire et la propriété des axes (axe fantôme) pour générer une trajectoire sinusoïdale propre à un axe dont on détermine la période (et ainsi la fréquence) par TA / TM.




```
CLOSE
DELETE GATHER UNDEFINE ALL
&1
#2->2000Y  ; X est un axe fantôme
OPEN PROG 1 CLEAR
INC
INC (R)
TA300
TS0
TM1000          ; TA/TM est la période
I13=10
NORMAL k-1      ; X-Y plane
CIRCLE1         ; mode circulaire
X0 Y0 I10       ; cercle complet (J = y initiale
CLOSE           ;                 par défaut )
```

**Figure II.13 :** Trajectoire sinusoïdale d'un axe simple avec un axe fantôme par interpolation circulaire

## II.6 LES PROGRAMMES PLC ET PLCC [5] :

En plus des programmes de mouvement, qui fonctionnent séquentiellement, le processeur a 64 programmes de PLC qui fonctionnent indépendamment des mouvements et avec une exécution répétitive et très rapide (32 programmes PLC compilés et 32 programmes PLC non compilés).

Les programmes PLC reposent sur le même principe que les contrôleurs programmables logique matériel (le balayage infini du programme jusqu'à ce qu'on le dise de s'arrêter ou une condition d'arrêt est satisfaite). Les programmes de PLC ont la plupart les mêmes constructions logiques que les programmes de mouvement, mais aucun PLC n'exécute aucun déplacement.

La plupart des programmes compilés de PLC sont très semblables, si non identique au PLC non compilé. En fait, avant de compiler un programme PLC il devrait être examiné et corrigé. La différence entre les deux types de programmes PLC est la gestion des buffers, des variables L, la syntaxe de commandes diffère aussi.

### II.6.1 LES PLC (*PROGRAMMABLE LOGIC CONTROLLERS*) :

Les programmes PLC fonctionnent de la même que les programme résidents sur PC. Les situations les plus fréquentes d'utilisation des PLC sont :

- Surveillance des entrées ;
- Forcer des sorties ;
- Envoi de messages ;
- Surveillance des paramètres de mouvement ;
- Changer les gains ;
- Démarrer et arrêter les programmes de mouvements.

Par leur accès complet aux variables de l'interface et aux E/S, et leur nature indépendante aux mouvements, les PLC constituent des outils très puissants de gestion des programmes de mouvement.

### II.6.1.1 Ecriture d'un programme PLC :

Les PLCs sont programmés de la même façon que les programmes de mouvement, dans une fenêtre d'éditeur de texte pour être télécharger plus tard vers l'interface. Il est à noté qu'après qu'un PLC serait fermé (avec la commande CLOSE) reste inhibé, mais il peut être rétabli avec la commande ENABLE PLC n (où n = 0 - 31 : représente le numéro du programme PLC).

Exemple de la structure d'un PLC :

```
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 de PLC dans la mémoire de l'interface sont activés (ENABLE) lors d'un *RESET*, il est recommandé d'avoir I5 = 0 tant qu'on développe des programmes de PLC, ceci permettra de n'avoir aucun fonctionnement de PLC après un reset (ainsi un PLC actif fonctionne seulement si I5 est placé correctement).

### II.6.1.2 Les commandes Conditionnelles :

La plupart des actions dans un programme PLC sont des instructions conditionnelles, dépendantes de l'état des variables de l'interface (de type : I, P, Q ou M), telles que les entrées, les sorties, les positions, et les compteurs, ... etc.
Leur teste peut se faire de deux manières soit par un déclenchement de niveau, soit par front d'impulsion ; les deux méthodes peuvent être faites, mais avec des techniques différentes :

- Teste de niveau :

On prend un exemple très simple : le teste d'une entrée affectée a la variable M11 :

```
IF (M11=1)
    P1=P1+1
ENDIF
```

Aussi longtemps que l'entrée est vérifiée, P1 s'incrémentera plusieurs fois par seconde, et quand l'entrée disparaît, P1 s'arrêtera de s'incrémenter.

- Teste du front d'impulsion :

Supposons maintenant qu'on veut incrémenter P1 à chaque fois que M11 est vérifiée, pour ce faire on ajoute une variable auxiliaire P11 :

```
IF (M11=1)
    IF (P11= 0)
        P1=P1+1
        P11=1
    ENDIF
ELSE
    P11=0
ENDIF
```

### II.6.1.3 La boucle WHILE :

Normalement un programme PLC s'exécute jusqu'à la fin dans un balayage simple. L'exception à cette règle se produit si le programme rencontre la boucle WHILE . Dans ce cas le programme s'exécute jusqu'à ENDWHILE et sortira de ce PLC. Après avoir parcouru tout les autres PLCs, il se branchera dans ce PLC a la ligne de la commande WHILE et non pas au début. Ce processus se répétera aussi longtemps que la condition est vraie. Quand la condition de WHILE devient fausse le programme PLC exécute le reste du programme.

**Exemple :**

```
WHILE (M11=1)
        P1=P1+1
ENDWHILE.
```

### II.6.1.4 Les commandes COMMAND et SEND :

Une des utilisations les plus communes des PLCs est l'activation des programmes de mouvement et la mise en marche des moteurs aux moyens d'appels des procédures. Quelques actions de COMMAND devraient être suivies de la commande WHILE pour assurer que son effet est pris avant de procéder au reste du programme PLC ( c'est toujours vrai si une deuxième action de COMMAND exige que la première action de COMMAND est finie).

**Exemple :** Supposons que nous voulons qu'une entrée arrête n'importe quel mouvement dans un système en coordination, et commence le programme de mouvement 10 :

```
M187->Y:$0817,17,1        ; &1 bit d'entrée pour la fin de mouvement des moteurs.
OPEN PLC3 CLEAR
P11=0
IF (M11=1)                ; l'entrée est activée.
   IF (P11=0)
      P11=1
      COMMAND"&1A" ; quitté tout mouvement.
      WHILE (M187=0)  ; attendre que le mouvement est fini.
      ENDW
      COMMAND"&1B10R" ; commencer le programme 10.
   ENDIF
ELSE
     P11=0                ; ré- initialisé la variable P11.
ENDIF
CLOSE
```

DISPLAY "{message}" envoie le message vers un afficheur LCD.
SENDS"{message}" Transmet le message sur le port série.

**Remarque :** N'importe qu'elle commande SEND, COMMAND, ou DISPLAY devrait être faite seulement sur une condition a déclenchement de front d'impulsion, parce que le PLC fait un cycle plus rapide que ces opérations ainsi, elles ne peuvent traiter leurs informations.

### II.6.1.5 La temporisation :

Les commandes de synchronisation comme DWELL ou DELAY sont seulement réservées pour les programmes de mouvement et ne peuvent pas être employées pour la synchronisation des PLC.

A la place, nous disposons de deux temporisateurs de 24-bit dans lesquels on peut écrire, et compter (à raison d'une décrémentation par cycle Servo).

Habituellement on utilise les I- variables : Isx11 et Isx12 où x : représente le numéro du système en coordination x = (C.S.# - 1 à 9), s = 5 (pour C.S.# = 1 à 9), alors le PLC attend jusqu'à ce que le I- variable soit égale a 0.

**Exemple :**

```
M1=1                ; initialiser M1 a 1.
I5111=2259          ; initialiser le temporisateur a 1 seconde (2259 servo cycles)
WHILE (I5111 > 0)   ; attendre jusqu'à ce que le temporisateur soit égale a 0.
ENDWHILE
M1=0                ; effacer la sortie M1.
```

### II.6.2 LES PLC COMPILES ( PLCC ) :

Il est possible de compiler les programmes de PLC pour une exécution plus rapide, cette rapidité vient du fait que : d'abord, l'élimination du temps d'interprétation, et en second lieu, la possibilité du PLCC d'exécuter les opérations arithmétiques et logique des nombres entiers de 24-bits (on utilise les variables L).

Les opérations en virgule flottante ( *foalting-points* ) dans un PLCC s'exécutent 2 à 3 fois plus rapidement que dans des programmes PLC; les opérations des nombres entiers (booléen y compris) s'exécutent 20 à 30 fois plus rapidement dans les PLCC que dans les PLC.

### II.6.2.1 La compilation des PLCC :

La compilation du programme PLC n'est pas exécutée dans UMAC, elle est faite dans le PC, le code machine résultant est alors chargé à UMAC.

Dans le logiciel exécutif PEWIN, le compilateur est intégré dans la routine de téléchargement, seulement le code compilé est chargé à UMAC, par conséquent il est

suggère de sauvegarder le code source ASCII dans l'ordinateur séparément puisqu'il ne peut être rechargé par UMAC. Si plus d'un PLCC est programmé, tous les codes de PLCCs doivent appartenir au même fichier texte ASCII, car le téléchargement d'un seul code de PLCC effacera tous les codes précédemment présents dans la mémoire, par conséquent, il reste seulement le dernier code compilé. Le dispositif de chargement de multiple- fichier dans le menu PEWIN File (Show project manager) laisse avoir les codes de PLCC dans des différents fichiers, ils seront combinés et compilés par PEWIN32Pro dans le processus de téléchargement.

### II.6.2.2 Les variables L :

L'utilisation des variables L dans un programme de PLC est le signe au compilateur que les commandes doivent être exécutées en utilisant des opérations de nombre entier au lieu des opérations en virgule flottante.
Pour mettre en application l'arithmétique de nombre entier dans un PLC compilé, l'utilisateur doit définir toutes les variables L pour être utilisé et remplacé les variables qui ont été employées dans les programmes sous la forme interprétée (habituellement M-variables, leur définition est la même).

Dans leurs définitions {variable}={expression}, les L-variables ne peuvent être combinées avec d'autre type de variables; c'est-à-dire, s'il y a une commande d'égalité ou un test des conditions avec les variables L, toutes les variables dans la commande, de part et d'autre du signe d'égalité, doivent être des L-variables, ou bien des constantes appartenant dans la gamme 8.388.608 à 8.388.607 ( $-2^{23}$ à $2^{23}-1$)

Exemples des conditions correctes (légales) :
IF (L50 = 1)
WHILE (L75 < L76)
IF (L1&L2 | (L4+L5) > 0 AND P1 = Q2 OR L6 != 0)
OR (L3 & L5 = $11 AND P1 = P2 OR L1 = 0)

Exemples des conditions illégales :
IF (L50 = P1) , IF (L1 + P2 = 0)
WHILE (L75 < 10000000)

Le compilateur interprétera seulement les commandes contenant les variables L (correctement définies), et les constantes de nombre entier, comme des opérations à exécuter en utilisant les opérations arithmétiques des nombres entiers dans PLCCs. (Il est à noté que les variables L sont valables seulement pour le compilateur et qu'elles ne signifient rien à UMAC).

### II.6.2.3 Elaboration des programmes PLCC :

Les PLCCs sont programmés de la même façon que les programmes de PLCs, et gardent les mêmes caractéristiques que les PLCC0 et PLCC1 - PLCC31. Les seules opérations à faire sont :

- Changement de toutes les références PLC au PLC compiler (PLC à PLCC)
- On définie les variables L pour les opérations arithmétiques des nombres entiers.
- Combinent tous les programmes PLCs à compiler dans un fichier sur le PC.

On s'assurent que l'option de "Support MACRO's/PLCC's" est vérifiée avant le téléchargement (elle se trouve dans le menu Setup\ Generale setup and Options\ Download dans le logiciel PEWIN) .

Un PLCCs pourrait être supprimer en utilisant la commande DELETE PLCCn (on remplace n par le nombre approprié).

Exemple de la structure d'un PLCC:

```
CLOSE
DELETE GATHER
DELETE TRACE
OPEN PLCC n CLEAR
        {les fonctions PLCC}
CLOSE
ENABLE PLCC n
```

## II.7 CALCUL GEOMETRIQUE [8] :

La carte d'axes dont nous disposons (UMAC) fournie des structures facilitant la mise en œuvre et la manipulation des calculs géométriques complexes. Les calculs géométriques sont exigés quand il y a un rapport mathématique non linéaire entre les coordonnées du bout de l'outil et les positions articulaires des mécanismes typiques dans des géométries non cartésiennes. Ils sont généralement plus employés dans les applications en robotique, mais peuvent être employés avec d'autres types tels que les machines outils.

Les routines géométriques sont incluses dans le contrôleur par l'intégrateur, et fonctionnent invisiblement aux programmateur et aux opérateurs. Ces routines peuvent être figer ou paramétrées pour les machines, elles peuvent s'adapter aux changements normaux tels que des longueurs d'outil et de différentes extrémités d'effecteurs.

Les calculs " *forward-Kinematic* " (géométrique directe) emploient les positions articulaires comme entrées, et les convertissent en coordonnées de pointe. Ces calculs sont exigés au début des mouvements qui sont programmés dans des coordonnées de pointe pour établir les coordonnées initiales pour le premier mouvement.

Les calculs " géométrique - inverse" emploient les coordonnées de la positions cartésiennes de l'outil comme entrée, et les convertissent en coordonnées articulaires. Ces calculs sont exigés pour le point final pour chaque mouvement programmé dans les coordonnées cartésiennes.

### II.7.1 CREATION DES PROGRAMMES GEOMETRIQUE :

UMAC met en application l'exécution des calculs géométrique par des programmes spéciaux *forward-Kinematic* et *inverse-Kinematic* (géométrique directe et géométrique inverse). Chaque système en coordination peut avoir un de chacun de ces programmes, et les algorithmes eux même peuvent être exécutés automatiquement aux temps exigés, appelés comme des sous-programmes du programme de mouvement.

## II.7.2 ELABORATION DES PROGRAMMES GEOMETRIQUE DIRECTE (GD) :

&1 OPEN FORWARD
CLEAR
{les instructions & les fonctions du modèle }
CLOSE

Les instructions sont les mêmes que celle utiliser dans les programmes PLCs (sauf les suivantes ADDRESS, CMDx, and SENDx ).
Les fonctions du modèle décrivent les équations qui existent entre les coordonnées cartésiennes et articulaires.

Avant n'importe quelle exécution du programme géométrique direct, UMAC placera les positions actuelles pour chaque moteur xx, dans le système en coordination, dans la variable globale Pxx, le programme peut alors employer ces variables en tant qu'entrée dans les calculs.

Après n'importe quelle exécution du programme géométrique direct, UMAC prendra les valeurs dans Q1 - Q9 pour le système en coordination, et copie ces derniers dans les 9 registres d'axes contenus dans ce système en coordination.

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

**Tableau II.1 :** Correspondance des variables Q avec les axes

Le but de base du programme géométrique directe, est de prendre les valeurs des coordonnées articulaires trouvées dans P1 – P32 pour les moteurs utilisés dans le système en coordination, et calculent les valeurs cartésiennes correspondantes, et les placent dans les variables Q1- Q9.

### II.7.3 EDITION DES PROGRAMMES GEOMETRIQUES INVERSES (GI) :

```
&1 OPEN INVERSE
CLEAR
    {les instructions & les fonctions du modèle inverse}
CLOSE
```

Avant n'importe quelle exécution du programme géométrique inverse, UMAC placera les positions actuelles pour chaque axe dans le système en coordination dans les variables Q1– Q9, le programme peut alors employer ces variables en tant que " entre " dans les calculs.

Après n'importe quelle exécution du programme géométrique inverse, UMAC lira les valeurs dans les variables Pxx (P1– P32) correspondent aux moteurs xx, dont le commande de définition d'axe est # xx->1.

### II.7.4 PROGRAMME DE GEOMETRIE INVERSE POUR LE MODE PVT :

L'UMAC peut également faire la conversion des vitesses de l'espace cartésien a l'espace articulaire dans le programme de géométrie inverse, pour permettre l'utilisation du mode PVT avec des calculs cinématiques. Avec les mouvements du mode PVT, les calculs de position sont faits juste comme pour n'importe quel autre mode de mouvement, sauf qu'il y a un ensemble de calculs de conversion de vitesse qui doit être fait, où on peut étudier le profil des vitesses angulaires, (en donnant leurs équations mathématiques), pour un calcule de trajectoire optimale.

A l'exécution du mode PVT, UMAC placera automatiquement les valeurs de vitesse commandées des axes à partir des commandes PVT dans les variables Q11 – Q19, pour le système en coordination désiré, avant chaque exécution du programme géométrique inverse. L'unité des vitesses est définie par Isx90 (par exemple : mm/min ou deg/sec).

La table suivante montre la variable utilisée pour chaque axe:

| Vitesse d'axe ( variables Q associés) | Axes Associés | Vitesse d'axe ( variables Q associés) | Axes Associés | Vitesse d'axe ( variables Q associés) | Axes associés |
|---|---|---|---|---|---|
| Q11 | A | Q14 | U | Q17 | X |
| Q12 | B | Q15 | V | Q18 | Y |
| Q13 | C | Q16 | W | Q19 | Z |

**Tableau II.2 :** Correspondance des variables Q avec les axes

UMAC placera également Q10 à 1 en ce mode comme indicateur au programme géométrique inverse qu'il devrait employer ces valeurs de vitesses d'axes pour calculer des valeurs articulaires de vitesse pour le moteur.

En ce mode, après n'importe quelle exécution du programme géométrique inverse, UMAC lira les valeurs dans les variables P1xx (P101 – P132) pour chaque moteur xx défini en tant qu'axes géométrique inverse ( # xx->I). UMAC les emploiera en tant que valeurs de vitesse de moteur avec les valeurs de position dans Pxx pour créer un mouvement de PVT pour le moteur.

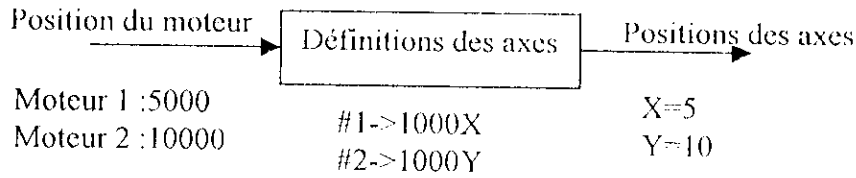
### II.7.5 EXECUTION DES PROGRAMMES GEOMETRIQUES :

Une fois que les programmes géométriques directs et inverses ont été crées pour un système en coordination, UMAC les exécutera automatiquement aux temps appropriés une fois que les calculs géométriques ont été activés (en plaçant Isx50 à 1). Aucune modification à un programme de mouvement n'est exigée pour l'accès aux programmes géométriques en temps voulu.

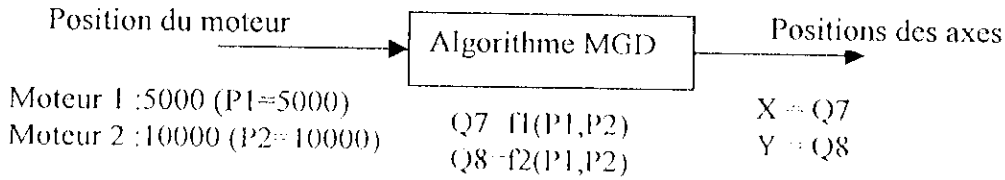
Le programme géométrique direct est exécuté automatiquement chaque fois que l'instruction R ou S (exécution étape par étape) est donnée au système en coordination (Bien sur que si Isx50 =1). Ceci est fait pour s'assurer que la position de commencement est correcte pour le calcul du mouvement initial.

## II.7.6 RECAPITULATIVE DE L'EXECUTION DE LA CONVERSION :

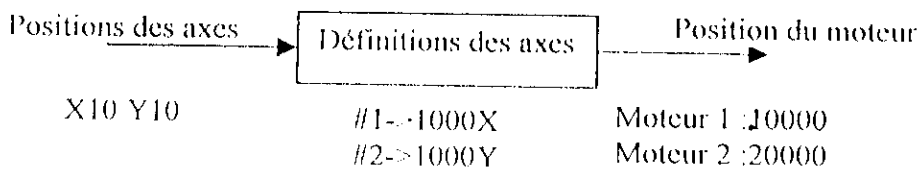
- **Conversion du moteur a l'axis sans le programme de GD (Forward-Kinematic):**



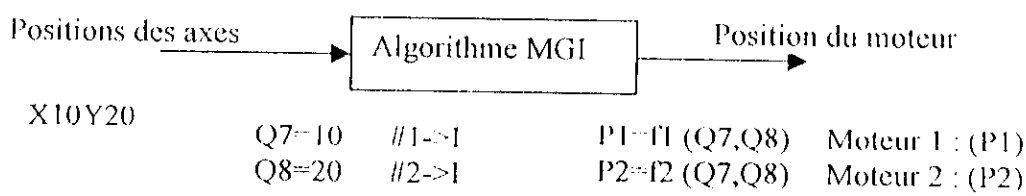

- **Conversion du moteur a l'axis avec le programme de GD (Forward-Kinematic):**




- **Conversion de l'axe au moteur sans le programme de GI (Inverse Kinematics) :**




- **Conversion de l'axe au moteur avec le programme de GI (Inverse Kinematics) :**



## II.8 CONCLUSION :

La description du langage de programmation de l'interface nous a permet de voir les grandes capacités de l'UMAC dans le domaine de programmation, elle peut supporter jusqu'à 32K programmes de mouvement et exécuter jusqu'à 256 programmes de mouvement simultanément (grâce à une extension des cartes Turbo PMAC2-3U CPU) avec une très grande précision. La disposition des variables et leurs classifications engendre un mode très flexible pour la configuration et la programmation de l'interface UMAC grâce au logiciel PEWIN32PRO développé par la firme **DELTA TAU** qui fournie un environnement interactif pour l'utilisateur permettant l'accès à toutes les variables internes de l'interface.

# *CHAPITRE III: Fonctionnement de l'UMAC*

## III.1 INTRODUCTION :

La flexibilité et la robustesse de l'interface UMAC que ce soit du point de vue logiciel ou de point de vue matériel se traduit par une architecture de fonctionnement complexe basée sur un processeur Motorola DSP 56303 avec une version récente de micros-programmes implémentées fournies par **DELTA TAU**, dont l'étude est assez intéressante.

## III.2 CHARGEMENT D'UN PROGRAMME A TRAVERS LE PORT SERIE [18] :

Le chargement d'un programme ou d'une commande en ligne est faite par le biais du code Bootloader de DSP56303, ce dernier permet à la DSP de charger le programme et les données d'une application par l'intermédiaire de l'interface série SCI aux mémoires X, Y et P ou de commencer à l'exécuter après le chargement. Il permet également de programmer les divers registres de contrôle du DSP avant d'exécuter le chargement du programme, le Bootloader est nécessaire pour les applications dans lesquelles l'utilisateur doit programmer des registres de contrôle ou télécharger des données a la mémoire X ou Y lors de l'initialisation de la DSP.

Comme le montre la Figure III.1, le chargement d'un programme ou d'une commande se produit dans trois étapes :

1. Le DSP est remise à l'état initial.
2. Le code Bootstrap s'exécute, c'est un code interne de démarrage du DSP, il est situé dans la mémoire interne ROM Bootstrap, ce dernier s'exécute pour charger le Bootloader du dispositif externe qui est un PC.
3. Le Bootloader s'exécute pour charger le programme dans la mémoire X, Y, P ou commence à exécuter le programme chargé.

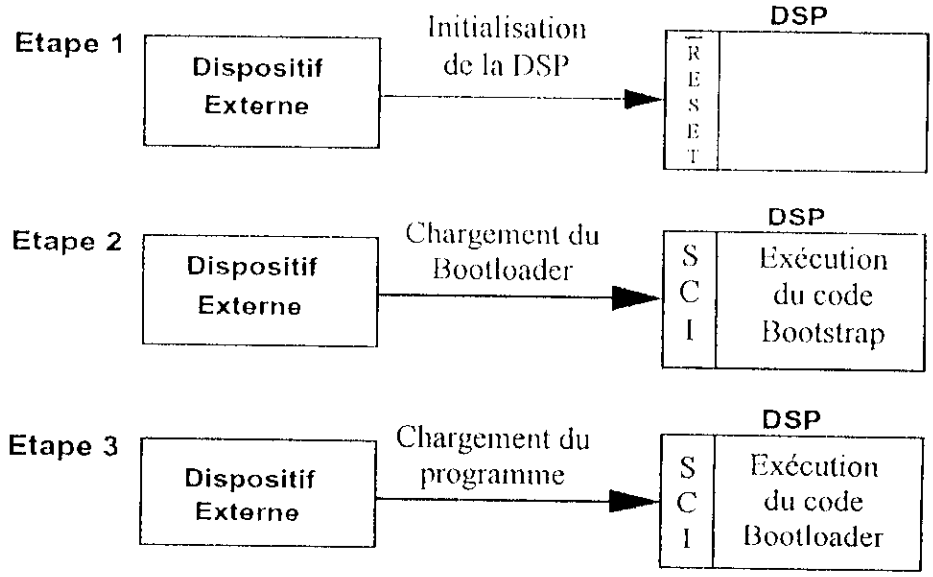



**Figure III.1 :** Etapes de chargement d'un programme à travers le port série

### III.2.1 INITIALISATION DU DSP :

Comme le montre la Figure III.1, le dispositif externe remet à l'état initial le DSP en activant la ligne RESET de ce dernier, puis le dispositif externe désactive le signal RESET, en réponse le DSP sort de l'état d'initialisation et charge les valeurs des broches de mode d'opération MODA, MODB, MODC et MODD dans les bits MA, MB, MC et MD du Registre de Mode d'Opération (OMR), ces configurations de bit déterminent le mode de fonctionnement et l'option du code Bootstrap que le DSP va utiliser par la suite.

Pour cette application, les broches de mode sont placées convenablement dans la configuration montrée dans le Tableau III.1, qui oriente le DSP vers l'amorçage par SCI, cette configuration dirige également le DSP pour ce brancher avec le code Bootstrap commençant à l'emplacement de mémoire de programme $FF0000 (ROM interne).

| Mode | Valeur |
|---|---|
| MODA | 1 |
| MODB | 0 |
| MODC | 1 |
| MODD | 0 |

**Tableau III.1 :** Mot de commande d'initialisation de l'OMR

### III.2.2 CHARGEMENT DE BOOTLOADER :

Comme il est présenté sur la Figure III.1, le dispositif externe charge le Bootloader au DSP, cette étape commence quand le DSP exécute le code Bootstrap, ce dernier lit les bits MA, MB, MC, et MD dans l'OMR et détermine quelle est la section du code à exécuter, si le mode d'opération est placé comme il a était décrit dans le paragraphe précédent, le DSP ce branche à la section qui charge le code du Bootloader par l'interface SCI, à ce moment le code Bootstrap s'attend à recevoir des mots de 3 octets (soit 24 bits). Le premier mot qu'il reçoit est le nombre de mots du programme à charger N (160), le deuxième mot est l'adresse de début DEB, après le code bootstrap s'attend à recevoir les mots du programme Bootloader, Quand il reçoit les N mots du Bootloader, il les place dans l'adresse mémoire programme commençant par DEB et commence à exécuter le programme Bootloader chargé. Réellement cette étape est terminer à l'ouverture d'une application de pilotage de l'interface UMAC sur le PC, ensuit le code Bootloader s'exécute continuellement.

### III.2.3 EXECUTION DU BOOTLOADER :

Le dispositif externe télécharge le programme d'application au DSP, comme il est présenté sur la Figure III.1, cette étape commence quand le DSP commence à exécuter le code Bootloader, la Figure III.2 montre les étapes du programme Bootloader, ceci s'explique par :

- Tout d'abord, le Bootloader reçoit un mot et transmet ce mot de nouveau au dispositif externe, la valeur réelle de ce mot n'est pas significative, le but de ce dernier est de lancer le transfert.
- Le Bootloader commence à recevoir des valeurs pour les registres de control du DSP, il reçoit des valeurs qui vont être écrits dans des registres appropriés :
  - Registre de Control ( PCTL )
  - Operating Mode Register ( OMR )
  - Register d'état ( SR )
  - Registres d'Attribution d'Adresse ( AAR0-3 )
  - Register de Control de Bus ( BCR )
  - Register de Control d'horloge ( SCCR )

- Encore le Bootloader reçoit un mot dont la valeur réelle n'est pas significative, le but de ce dernier est de relancer les transferts après que les registres de contrôle seraient écrits, en suite il reçoit trois mots indiquant quoi faire :

  1- Le premier mot décrit quelle action à prendre, ce mot devrait avoir une valeur entre zéro et trois, l'action pour chacune de ces valeurs est comme suit :

  - 0 = chargement à la mémoire P.
  - 1 = chargement à la mémoire X.
  - 2 = chargement à la mémoire Y.
  - 3 = exécution du code (commande en ligne).

2- Le deuxième mot que le Bootloader reçoit est le nombre de mots à charger N, ce mot est ignoré si le mot précédent demandait au Bootloader pour exécuter le code est une action = 3.

3- Le troisième mot que le Bootloader reçoit est l'adresse du début DEB, c'est l'adresse de mémoire où le Bootloader commence à charger les données ou a exécuter le code.

- Si une des actions de chargement est choisie, le Bootloader commence à recevoir les données, le Bootloader reçoit les N mots et les place dans l'espace mémoire approprié commençant au DEB.
- Au moment ou tous les N mots sont chargés, un mot contrôle est transmis de nouveau au dispositif externe, et le Bootloader se branche de nouveau à la section où l'action est reçue, ainsi plusieurs données peuvent être charger ou plusieurs programmes peuvent être exécuter
- Si le code d'action d'exécution est choisi, le Bootloader commence à exécuter le programme chargé depuis l'adresse DEB.

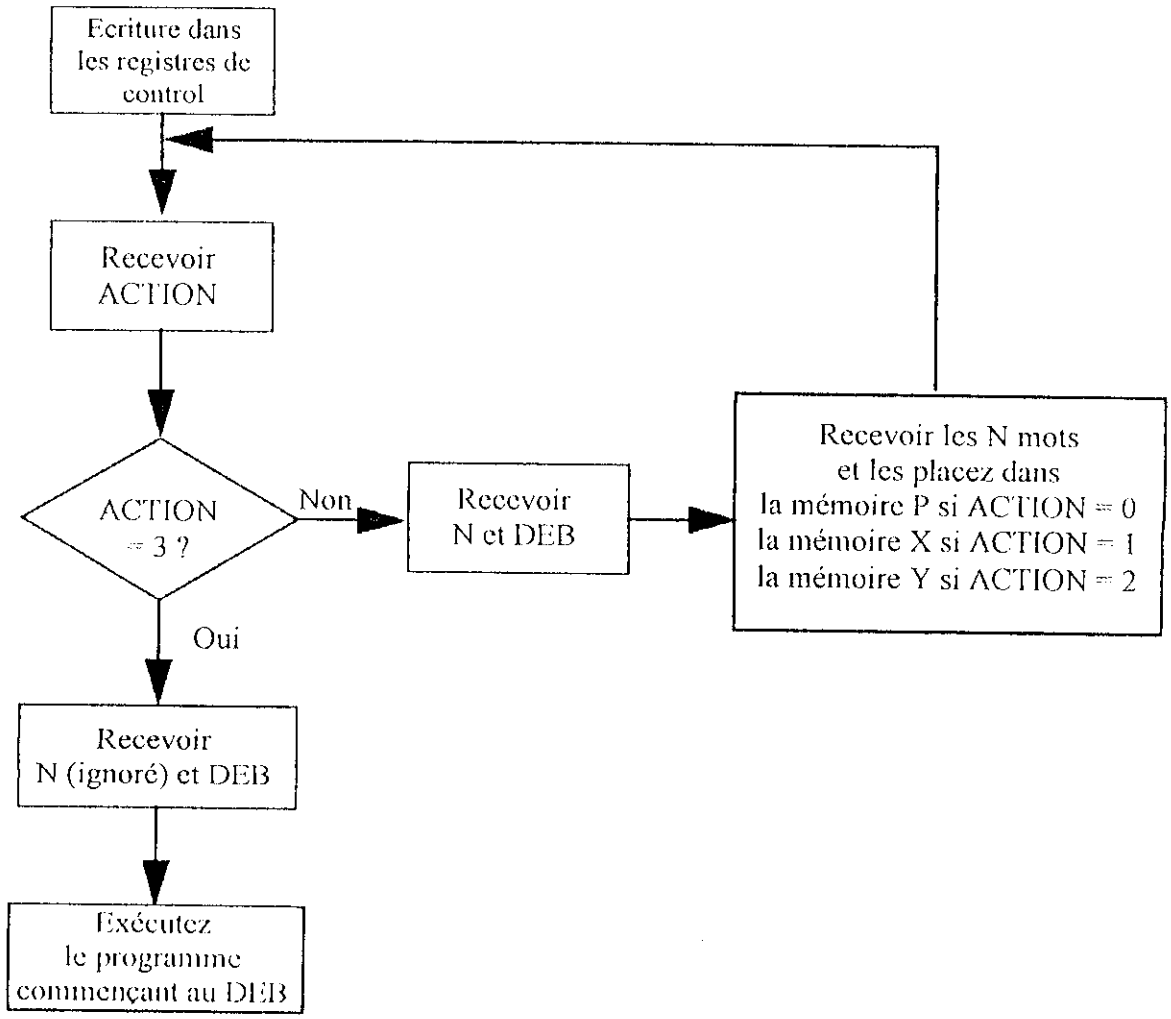



**Figure III.2** : Etape du programme Bootloader

## III.3 VALEURS NUMERIQUES TRAITES PAR LE PROCESSEUR [5] :

Le processeur peut enregistrer et traiter des valeurs numériques sous beaucoup de formes, avec des valeurs à virgule fixe et à virgule flottante, le processeur Motorola DSP 56303 qui agit en tant que l'unité centrale de traitement de l'UMAC est un processeur à virgule fixe avec la capacité de travaille en arithmétique de 24 bits et 48 bits (plus un accumulateur de 56-bit). Cependant, des micros-programmes de l'UMAC mettent en application un ensemble complet des sous-programmes à virgule flottante.

### III.3.1 FORMAT INTERNE DES DONNEES :

Tous les algorithmes de commande internes, les interpolations et les sous-programmes de commutation fonctionnent avec une arithmétique à virgule fixe de 8-bit et 24 bits pour une vitesse maximum.

Les programmes utilisateur de mouvement et PLC, utilisent une arithmétique à virgule flottante pour un maximum d'application, cependant lorsqu'il y'a une lecture ou une écriture aux registres de ces derniers, elle se fait à virgule fixe et les formats intermédiaires sont des valeurs à virgule flottante, la seule exception à cette règle c'est les programmes compilés PLCC contenant seulement des " variables L" et des constantes de nombre entier, le format intermédiaire dans ce cas est le nombre entier de 24 bits signé.

Le format à virgule flottante est un format long de 48 bits, avec une mantisse de 36 bits et un exposant de 12 bits, ceci fournit un intervalle de $\pm 2^{\pm 2047}$ (ou $\pm 3{,}233 \times 10^{\pm 616}$), qui est largement suffisant pour tous les usages prévisibles.

### III.3.2 RECEPTION DES VALEURS :

les constantes envoyées par le PC en tant qu'élément des lignes de commande sont envoyées comme un code texte ASCII, en valeurs décimales ou hexadécimales :

- Les valeurs décimales peuvent être positives ou négatives, et peuvent inclure des valeurs fractionnaires. L'interpréteur des valeurs numérique (dans les logiciels) ne peut pas supporter la notation exponentielle, et il est limité par des valeurs dans l'intervalle $\pm 2^{\pm 35}$ (ou $\pm 3{,}43 \times 10^{\pm 10}$), les valeurs en dehors de cet intervalle sont tronquées aux valeurs maximums ou minimums de ce dernier.
- Les valeurs hexadécimales doivent être précédées par un caractère $, elles doivent être non signées et elles ne peuvent pas inclure les valeurs fractionnaires.

### III.3.3 EMISSION DES VALEURS :

Le processeur émis des valeurs numériques au PC en tant qu'élément des lignes de réponse sous la forme décimale des textes en code d'ASCII (bien que des valeurs d'adresses puissent être émise sous la forme hexadécimale en code ASCII selon I9=2 ou 3). L'émetteur (dans l'interface ) de valeur est limité par des valeurs dans l'intervalle de $\pm 2^{\pm 47}$ (ou $\pm 1{,}41 \times 10^{\pm 14}$), les valeurs qui sont à l'extérieur de cet intervalle sont tronquées aux valeurs maximums ou minimums de ce dernier.

## III.4 ADRESSAGE DES MEMOIRES [5] :

Le processeur Motorola DSP56303 de l'interface UMAC utilise un bus d'adresses de 19 bits (18 bits de bus adresse + 1 ligne d'attribution d'adresse) pour pointer des zones mémoires contenant des mots de données de 24 bits dans les deux champs X et Y, ceci dit deux mots dans champs X et Y (respectivement) peuvent être associer pour former un long mot de 48 bits, où le mot X forme le LSB et le mot Y forme MSB, ce qui fait qu'on se retrouve dans une situation où on peut adresser des mots de données de 48 bits dans une zone mémoire allons de $000000 à 07FFFF.

## III.5 ADRESSES DES REGISTRES D'ENTREES/SORTIES [5] :

Les données de mesure de la vitesse et de la position ainsi que la sortie que reçoit l'amplificateur (a partir de DAC) sont situées dans des adresses bien spécifiques au type d'application traitée, de ce faite pour une application déterminée il faut spécifier :

- Ixx02 : c'est l'adresse de registre où la commande de sortie (commande en courant) sera écrite (en outre c'est le registre du DAC).
- Ixx03 : généralement c'est l'adresse où la mesure de la position sera lue et dans le cas où on utiliserait une Boucle Duel Feedback c'est l'adresse du registre qui contiennent la valeur du couple.
- Ixx04 : c'est l'adresse de registre ou la mesure de vitesse sera lue, cette mesure sera dérivée d'une façon numérique pour obtenir la valeur de la vitesse par conséquent ce registre contient la mesure de la position.

## III.6 MICRO-PROGRAMMES IMPLEMENTES :

La version 1.936 des micros-programmes fournis par **DELTA TAU** est très riche en applications, nous nous limiterons à la description des micros-programmes traitant la commande du mouvement que nous jugeons importants.

### III.6.1 ALGORITHME D'INTERPOLATIONS « 1/T SUB-COUNT » [5]:

C'est des opérations pour calculer la vitesse et la position d'une manière précise, chaque canal d'encodeur a deux registres de Timers, le premier temporisateur retient le temps T1 entre les deux dernières transitions A (CHA) et B (CHB) d'encodeur (Figure III.3), sur une interruption activée avec un front descendent de l'horloge SERVO, la vitesse est estimée en tant qu'étant inversement proportionnelle à ce temps, c'est une évaluation très précise en particulier à de basses vitesses.

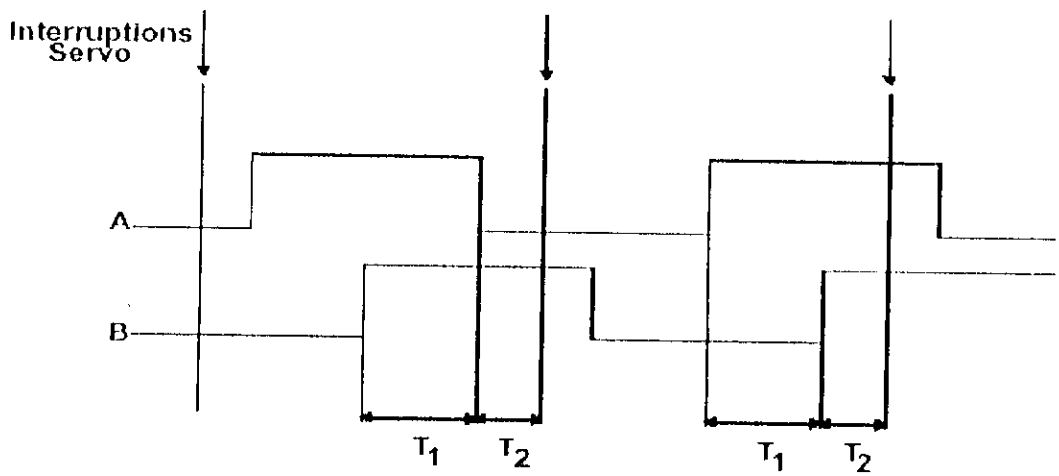



**Figure III.3** : Principe de mesure de T1 et T2 par les Timers d'entrées encodeur

Le deuxième Timer T2 retient le temps depuis la dernière transition, de même à chaque interruption SERVO la valeur du deuxième temporisateur se divise par la valeur du premier temporisateur pour obtenir la distance fractionnaire de la dernière transition qui doit

s'ajouter au nombre d'incrément compté par le compteur de l'encodeur pour aboutir à une mesure plus précise, ceci s'exprime par :

$$\begin{cases} Vn = \dfrac{K}{T_1} & ; Vn : \text{la vitesse d'encodeur au cycle SERVO n.} \\ & ; K : \text{constante fixe.} \\ Pn = \text{compte} \pm \dfrac{T_2}{T_1} & ; Pn : \text{la position d'encodeur au cycle SERVO n.} \end{cases} \quad \text{(III.1)}$$

**Remarque :** l'interface utilise cette méthode pour le calcule précis de la position seulement, pour le calcule de la vitesse il sera déduit par dérivation numérique de la position.

### III.6.2 TABLE DE CONVERSION D'ENCODEUR [5] :

Pour une mesure précise de la position, l'interface emploie un processus multiple-étape (Figure III.4) en exploitant l'information contenue dans les compteurs et les Timers moyennant une conversion pour trouver la position exacte à utiliser dans l'algorithme de commande.

La première étape dans le processus de conversion de la position est matérielle, les compteurs et les Timers associés à l'encodeur fonctionnent continuellement sans interposition directe du logiciel (bien qu'ils puissent être configurés par le logiciel), au-delà de ce point, le processus est contrôlé par le logiciel, au début de chaque cycle SERVO (front décodent), le signal d'interruption SERVO est envoyé pour verrouiller tous les registres qui contiennent les données non opérationnelles.

En ce moment, le processeur emploie une structure logicielle appelée "la Table de conversion d'encodeur" pour le traitement de l'information des registres verrouillés, cette table indique au processeur quel est le registre à traiter et comment les traiter.

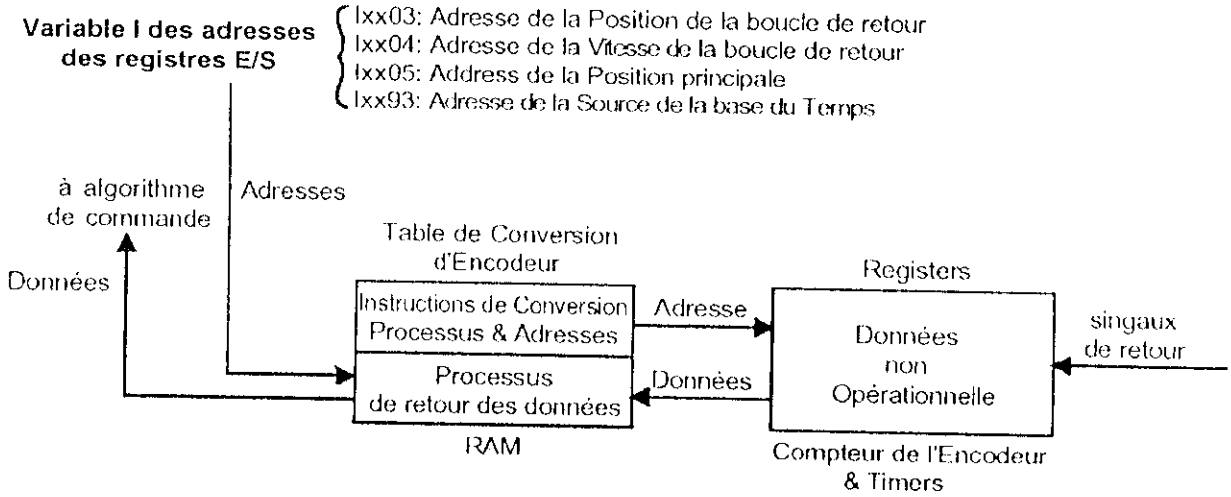



**Figure III.4** : Principe de la table de conversion d'encodeur

### III.6.2.1 Structure de la Table de conversion :

La Table de conversion d'encodeur à deux colonnes, une dans l'espace mémoire X et l'autre dans l'espace mémoire Y du processeur, la colonne X contient les données converties, alors que la colonne Y contient les adresses des registres sources et les méthodes de conversion employées sur les données dans chacun de ces registres source (du compteur et Timers). Fondamentalement, l'installation de cette table est faite par l'écriture de la colonne Y (par les variables 18000-18191), et le processeur emploie les données de la colonne Y pour remplir la colonne X à chaque cycle SERVO.

### III.6.2.2 Méthodes de Conversion :

Pour faire une conversion, les 4 derniers bits qui indiquent la méthode (ou le format) sont associés avec le bit20 du mode et les 19 bits premier de l'adresse pour remplir les 24-bit du mot Y dans la table de conversion.

Il y'a plusieurs méthodes qui peuvent être appliquées (de point de vue logiciel), mais selon le dispositif de mesure qui est présent (Encodeur incrémental), on peut utiliser que les méthodes disponibles pour les encodeurs incrémentaux qui sont $0, $8 et $C, ces trois méthodes de table de conversion utilisent les registres d'encodeur incrémental dans la DSPGATE, chaque

méthode fournit un résultat qui est traité avec une unité de 1/32 comptes (par rapport aux données qui se trouvent dans le registre du compteur), les 5 premiers bits du résultat fournissent la partie fractionnaire.

• **Extension 1/T :** Avec la méthode $0, la partie fractionnaire est calculée en appliquent la méthode d'interpolation « 1/T Sub-count », cette technique est connue sous le nom "extension 1/T", elle est utilisée part défaut (et même dans le cas notre application) et généralement c'est la méthode le plus appliquée.

• **Extension Parallèle** : Avec la méthode $8, la partie fractionnaire est calculée en lisant les 5 bits d'entrée indicateur à effet hall USERn, Wn, Vn, Un, et Tn en cas de présence de dispositif de mesure de ce type.

• **Aucune Extension** : Avec la méthode $C, la partie fractionnaire est forcée à zéro, ce qui signifie qu'il n'y a aucune extension du compte, cette configuration est employée principalement pour vérifier l'effet d'extension des deux méthodes décrites précédemment.

### II.6.3 ALGORITHMES DE COMMANDE IMPLEMENTES:

Du point de vue commande de mouvement cette interface offre une grande facilité d'utilisation, car il est possible de commander en boucle fermée comme en boucle ouverte, sans avoir recours à implémentation ou à l'écriture des algorithmes de commande, les algorithmes de commande sont déjà implémenter, et il reste seulement de déterminer les variables pour avoir les réponses souhaitables selon le cahier de charge, ceci dit, pour les différent moteur (à courant continu, alternatif synchrone ou alternatif asynchrone) les algorithmes de commande ne sont pas de la même structure.

#### III.6.3.1 Commande en boucle ouverte :

La commande en boucle ouverte se fait par rapport à la valeur du courant et avec la simple commande en ligne O (sur logiciel), par exemple pour une référence de courant qui est de 10% du courant maximale (0.3A), se fait par la simple commande : O10 <CR>.

### III.6.3.2 Commande en boucle fermée :

La commande en boucle fermée se fait tout d'abord par la commande j/ qui ferme la boucle de position vitesse, puis on affecte à l'axe de mouvement (X, Y, Z, ....) une valeur de déplacement, et à l'exécution du programme, l'algorithme de commande en boucle fermée s'exécute automatiquement (cela est détaillé dans la partie logicielle).

### III.6.3.3 Commande en cascade :

Pour la commande du mouvement des moteurs, la régulation de la position (respectivement la vitesse) et du couple (respectivement le courant) se font en cascade, comme présenté à la Figure III.5, la boucle la plus interne est la boucle de couple, le réglage de couple revient à un réglage de courant, car les lois de base des moteurs affirme que le couple est proportionnel au courant du moteur.

D'un autre coté la 2ème loi de newton déclare que le couple (ou la force) est proportionnel à l'accélération, ainsi par rapport à la boucle externe qui est une boucle de réglage de position, la commande que reçoit la boucle interne est une commande d'accélération.

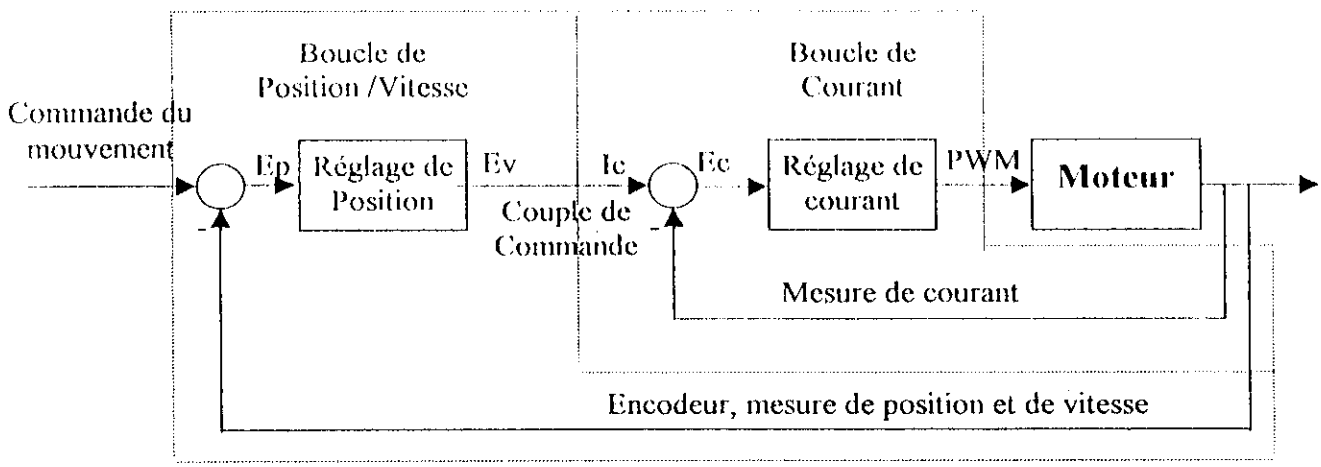



**Figure III.5** : schéma fonctionnelle Boucle d'asservissement en cascade

### III.6.3.4 Boucle de courant digital [6]:

Un algorithme de boucle de courant est implémenté, cet algorithme ferme la boucle de courant par des dispositifs de mesure de courant pour générér des commandes en tension PWM que doivent recevoir les amplificateurs, et comme le montre la Figure III.6, dans le cas des moteurs à induction cet algorithme applique les transformations de PARK directe et inverse pour le calcule du courant direct (courant de magnétisation fixée par Ix77) et du courant de quadrature (pour la commande du couple).

Pour cette boucle de courant, la sortie de commande de la DSPGATE est de type PWM (commande en tension) qui nécessite des amplificateurs de tension et des dispositifs de mesure de courant digital adapter à l'interface UMAC, ce qui n'est pas le cas pour le matériel que nous disposons, par conséquent cette boucle sera désactiver dans notre application.

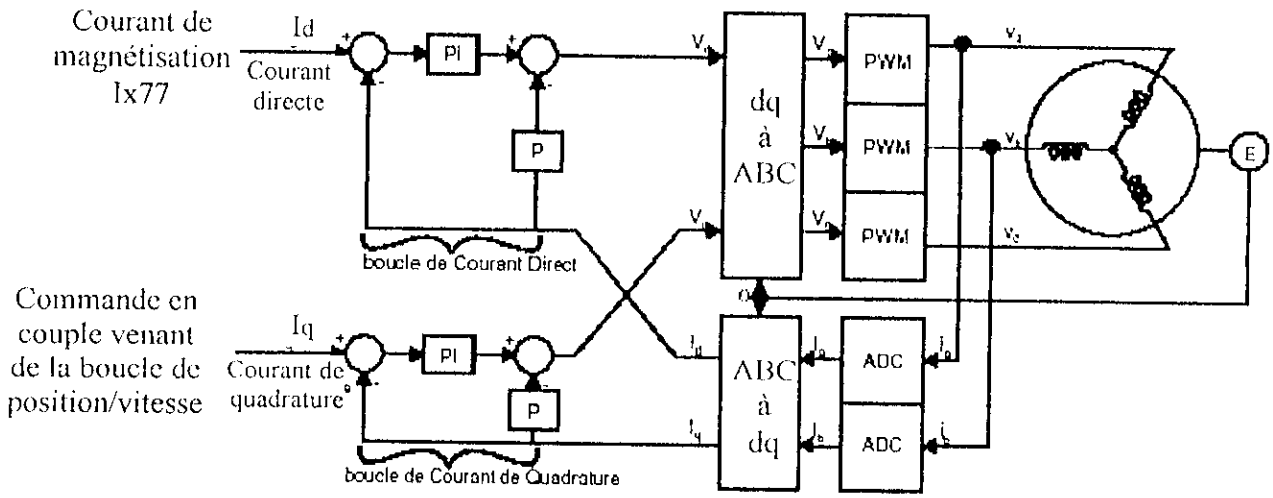



**Figure III.6** : Boucle digitale d'asservissement de courant
Pour un moteur triphasé

### III.6.3.5 Boucle de courant analogique [6] :

Dans le cas ou la boucle digitale est désactivée, les amplificateurs doivent recevoir une commande en courant (couple), la mesure et la fermeture de la boucle de courant sont faite à l'intérieur de l'amplificateur, dans ce cas on a affaire à une boucle de courant analogique. Ceci a l'avantage de faciliter l'utilisation et d'isolé les circuits numériques des

circuits de puissance, mais l'inconvénient de ne pas pouvoir changer les paramètres de cette boucle.

Pour notre application ce type de boucle de courant sera utiliser (car la disposition matérielle l'oblige) et elle sera considérée suffisante selon le cahier de charge.

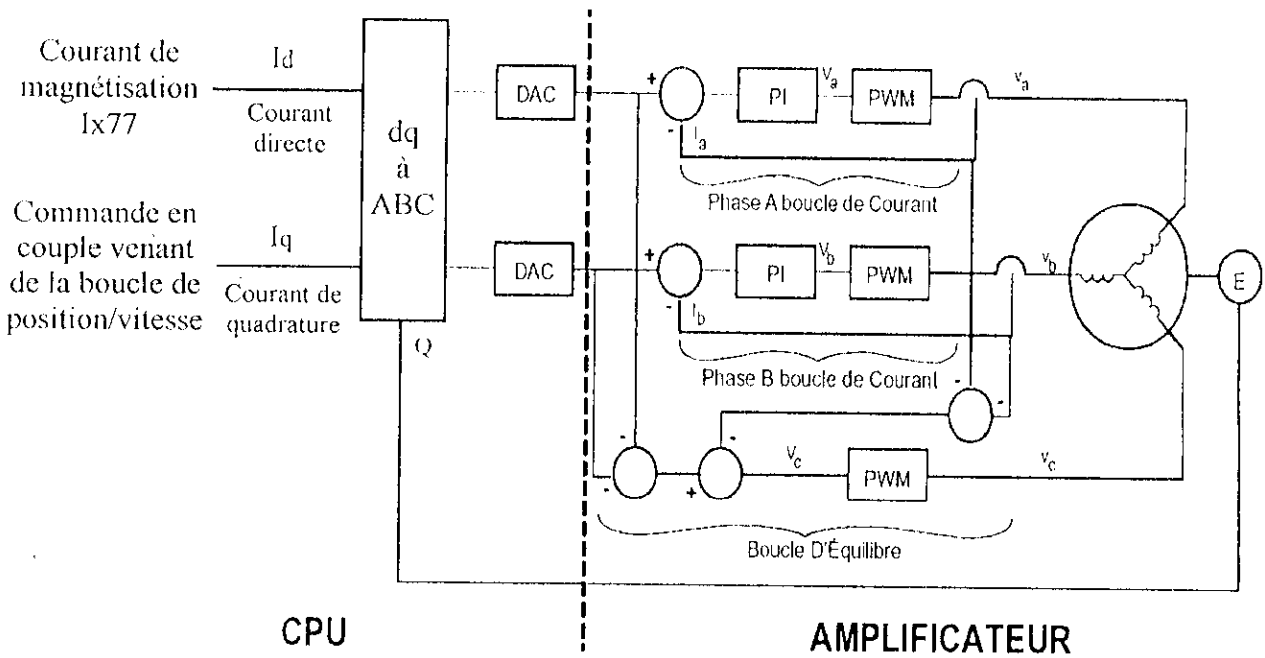



**Figure III.7** : Boucle analogique d'asservissement de courant pour un moteur triphasé

Comme le montre la Figure III.7, dans le cas des monteurs triphasé, les transformations de PARK fait pour le calcule de courant direct et quadrature, qui sont nécessaire pour la boucle de commande globale sont implémenté dans l'interface.

### III.6.3.6 Exécution des commutations des moteurs [4]:

La commutation est le changement de direction de mouvement d'un moteur, comme c'est le cas des moteurs à courant contenu où cette commutation corresponds à un changement de polarité des bornes du moteur. Et pour les moteurs à induction, cette commutation est un changement des phases (sens direct ou inverse).

L'exécution de cette fonction se passe soit à l'amplificateur s'il est possible (comme notre cas), ou soit dans le CPU par les algorithmes de commutation qui s'exécutent en parallèle avec l'algorithme de commande. Par conséquent la commutation sera désactivée, du moment qu'on dispose des amplificateurs qui exécutent la commutation automatiquement.

### III.6.3.7 Algorithme de commande PID implémenté [5] :

Un algorithme de commande standard PID (Proportionnelle Intégral Dérives) est implémenté dans l'interface, en association avec une boucle de compensation feedforward et un filtre réjecteur, cet algorithme est considéré suffisant pour la plupart des applications de contrôle de mouvement, il est facilement accessible pour imposer ou changer les variables afin d'optimiser la commande, ceci est fait en plaçant les variables I appropriées pour chaque moteur, le schéma fonctionnel de cette boucle est présenté dans la Figure III.8.

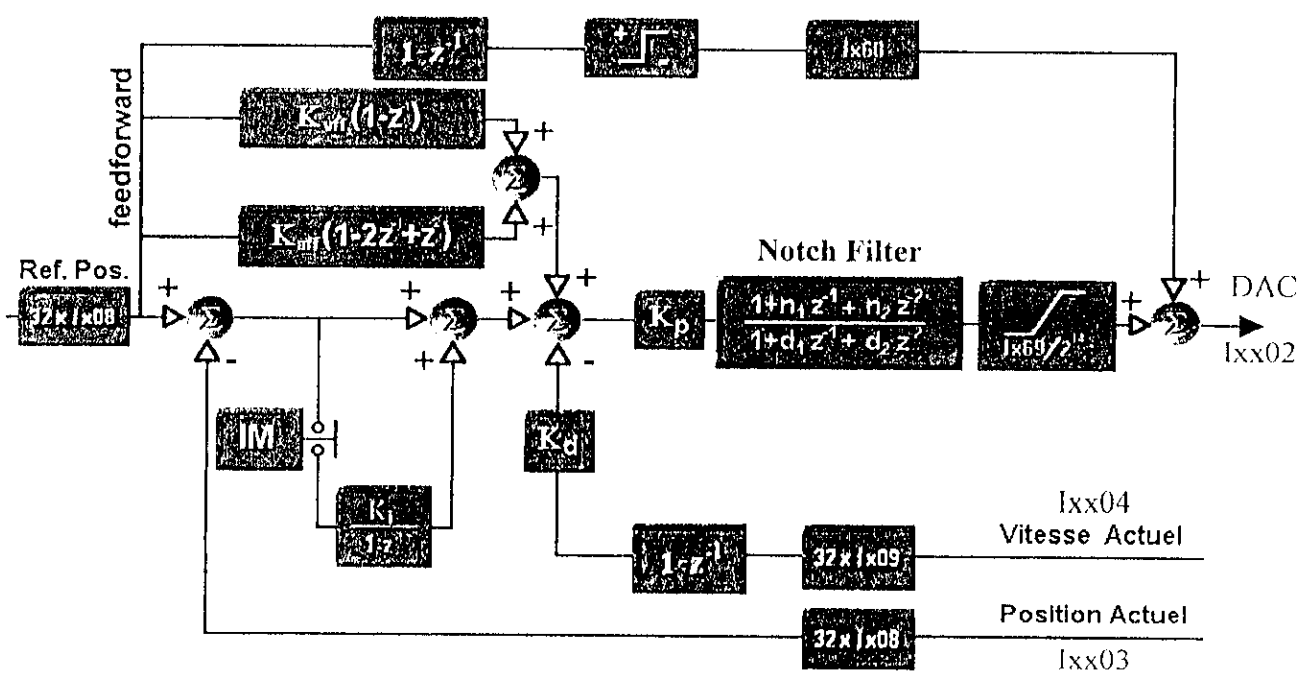



**Figure III.8** : Algorithme PID de la boucle de commande

### III.6.3.8.1 Termes PID présentes dans boucle de commende [5] :

• **Gain proportionnel Kp (Ixx30) :** Augmenter le gain proportionnel augmente la fréquence normale (pulsation propre) du système en boucle fermée, théoriquement l'augmentation du gain proportionnel aura comme conséquence l'amélioration de la commande en positionnement (rapidité de la réponse), mais souvent dans la pratique, l'augmentation du gain proportionnel augmente la sensibilité au bruit, c'est pour cela qu'il est toujours judicieux de commencer par la plus basse configuration qui est estimée à 2000 (par défaut) puis augmenter au fur et a mesure.

• **Gain dérivé Kd (Ixx31) :** Ce gain fonctionne comme un amortisseur, car il joue sur le coefficient d'oscillation du système, plus on augmente ce gain, plus il fournit une meilleure atténuation pour la stabilité. Ce gain empêche les dépassements mais rend le système lent, en outre dans le système digital la quantification des bruits est amplifiée quand le gain dérivé est appliqué, et dans le système en mouvement lent ce bruit pourrait contribuer de manière significative en la valeur de l'erreur.

• **Gain intégral Ki ( Ixx33 ) :** Pour les moteurs cette action agit pour éliminer l'erreur d'état d'équilibre (statique) due aux frottements, cependant une grande valeur de ce paramètre à un effet déstabilisateur du système.

• **Mode d'intégration IM ( Ixx34 ) :** En plaçant ce paramètre à 0 l'action intégrale sera activer tout le temps, et en le plaçant à 1 l'action intégrale sera activer seulement quand la valeur de la vitesse est égale à zéro.

### III.6.3.8.2 termes feedforward présentes dans boucle de commende:

• **Gain de vitesse feedforward Kvff ( Ixx32 ) :** Ce paramètre contribue à la réduction des erreurs à l'état d'équilibre, il est placé dans la boucle de régulation d'une manière non-causale. De ce fait, placer ce paramètre d'une manière arbitraire déstabilise le système, c'est pour cela qui est raisonnable et parfois même optimale de mettre ce gain égale au gain dérivé.

• **Gain d'accélération feedforward Kaff ( Ixx35 ) :** Kaff offre une limite de poursuite à une variation brusque de la valeur de la vitesse, la détermination de ce gain implique des calculs un peu complexes, mais il y a des voix intuitives pour appliquer ce gain, si la courbe de vitesse (mouvement d'un profil de vitesse parabolique) du système montre une mauvaise

poursuite à l'accélération ou à la décélération initiale, l'application de ce gain est indispensable.

• **Gain feedforward de frottement (Ixx68 ) :** Ce gain est activé quand l'erreur de la position n'est pas dans les limites de ce dernier à état d'équilibre, selon la direction requise pour la compensation, fixer cette limite trop haute peut déstabiliser le système.

### III.6.3.8.3 paramètres de sûreté :

• **DAC limite ( Ixx69 ) :** Ce paramètre est fixé de telle façon que le courant ne dépasse pas les limites qui ne peuvent pas être acceptées par l'amplificateur ou le moteur, afin que le courant excessif ne brûle pas le moteur ou l'amplificateur.

• **Limite fatal de l'erreur ( Ixx11 ) :** Ce gain agit quand l'amplitude de l'erreur excède Ixx11, dans ce cas le moteur x est désactivé.

**Remarque :** un algorithme de détermination de ces paramètres est proposé dans l'annexe 3.

### III.6.3.9 Commande résultant du Régulateur PID [5] :

On ne tenant compte que des termes PID décrit précédemment, la commande DAC qui est reçut par l'amplificateur est exprimer par la relation suivant :

$$DACout(n) = 2^{-19} \cdot Ix30 \left[ \left\{ Ix08 \cdot FE(n) + \frac{Ix32 \cdot CV(n) + Ix35 \cdot CA(n)}{128} + \frac{Ix33 \cdot IE(n)}{2^{23}} \right\} - \frac{Ix31 \cdot Ix09 \cdot AV(n)}{128} \right] \quad \text{(III.2)}$$

Ou :

- DACout(n) est la commande de sortie que reçoit l'amplificateur dans le cycle n avec une résolution de 16 bits (-32768 à +32767), qui est convertie en ±10V (limité par Ix69).
- Ix30 est le gain proportionnel du moteur x.
- Ix08 est une limite interne de graduation de position pour le moteur x (habituellement réglé à 96 ).
- Ix09 est une limite interne de graduation pour la boucle de vitesse du moteur x.
- FE(n) (*Felowing Error*) est l'erreur dans le cycle n, qui est la différence entre la commande de position et la position actuelle dans ce cycle : [ CP(n) - AP(n) ].

- AV(n) est la vitesse réelle dans le cycle n, qui est la différence entre les deux dernières positions réelles [ AP(n) - AP(n-1) ] dans ce cycle.
- CV(n) est la commande de vitesse dans le cycle n, qui est la différence entre les deux dernières positions de commande [ CP(n) - CP(n-1) ] dans ce cycle.
- CA(n) est l'accélération de commandée dans le cycle n, qui est la différence entre les deux dernières commandes de vitesse [ CV(n) - CV(n-1) ] dans ce cycle.
- IE(n) est l'erreur intégrée après le cycle n, qui est: $\sum_{j=0}^{n-1}[PE(j)]$

### III.6.3.10 *Notch Filters* [5] :

C'est un filtre d'anti-résonance (bande-rejeter) utilisé pour contrecarrer une résonance physique, il agit directement sur la sortie du PID. En général ce filtre est installé avec une bande de rejection d'environ 90% de l'amplitude à la fréquence de résonance et avec un filtre passe-bas de fréquence de coupure un peu plus grande que la fréquence de résonance pour réduire le gain à haute fréquence du filtre lui-même.

Le numérateur de ce filtre N(z) est de deuxième ordre et il constitue la partie anti-résonance. Le dénominateur D(z) est aussi de deuxième ordre et il constitue la partie passe-bas, les paramètres n1, n2, d1, d2 sont fixés par les variable Ix36, Ix37, Ix38, Ix39 respectivement, ils prennent leur valeur dans l'intervalle [ -2, 2 ].

### III.6.3.11 Systèmes à *Duels Feedback* [5] :

Dans la plupart des cas, le registre qui est utilisé pour la mesure de la vitesse est le même employé pour la fermeture de la boucle de position, ce qui signifie que les variables d'adresses Ix03 et Ix04 sont égales, cependant aujourd'hui le concept de systèmes à « *duel feedback* » devient de plus en plus populaire dans des systèmes de commande de mouvement, dans un tel système, il y a un capteur de position et un capteur qui mesure la charge appliquée sur le moteur.

Un capteur de la charge sur le moteur (souvent une échelle linéaire) fournit une mesure plus précise de la position qu'un capteur sur le moteur, parce que son exactitude n'est pas affectée

par les imperfections du couplage de moteur-charge. Cependant il peut également rendre l'axe moins stable, parce que ce dernier à des imperfections de couplage (jeu de denture) qui sont injecter à l'intérieur de la boucle. Tandis que le capteur de position sur le moteur est moins précis que le précédent, il fournit une meilleure stabilité parce que ces imperfections ne sont pas à l'intérieur de la boucle.

De ce fait, il est possible d'obtenir l'exactitude et la stabilité en utilisant un capteur de position et un capteur de la charge sur le moteur dans le système de l'interface, on emploie simplement l'encodeur de la charge pour fermer la boucle de position (pour l'exactitude), en utilisant la variable 1x03 pour adresser cet encodeur (par exemple l'adresse de ENC1 s'il est utilisé), et on emploie l'encodeur de position du moteur pour fermer la boucle de vitesse (pour la stabilité) en utilisant 1x04 pour adresser cet encodeur (par exemple l'adresse ENC2 s'il est utiliser).

### III.6.3.12 Algorithme de commande étendu [5] :

Pour les systèmes qui ont une dynamique plus difficile, telle que les systèmes qui on des résonances multiples ou des résonances de basse fréquence, un algorithme de commande étendu peut être utilisé au lieu de l'algorithme PID standard, la Figure ci-dessous montre la structure générale de l'algorithme ESA (*Extended Servo Algorithm*) sous une forme de schéma fonctionnel.

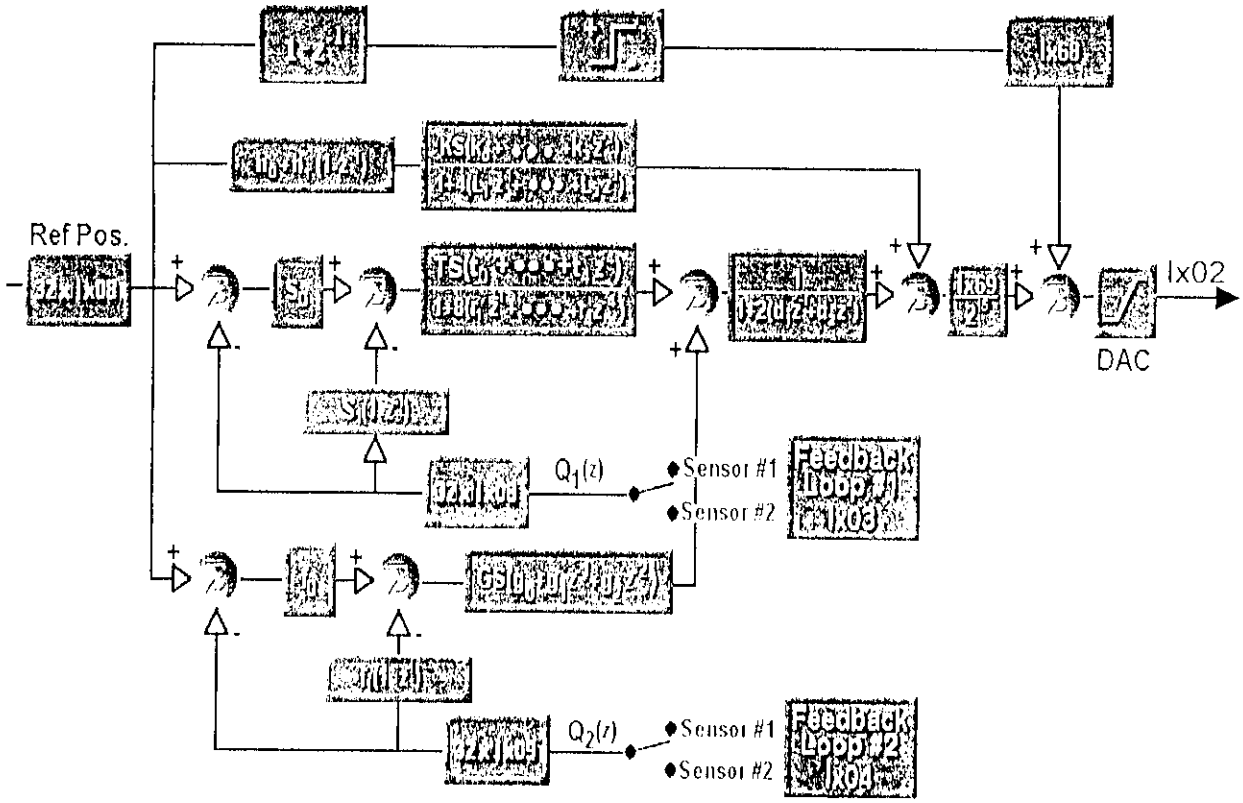



**Figure III.9** : Algorithme de commande étendu

Comme le montre la Figure III.9, il y a deux boucles de retour et une boucle feedforward, les données du retour 1 et retour 2 peuvent être soit en format position/vitesse, soit en format charge/vitesse pour les applications Duel Feedback. pour la boucle feedforward, il y a deux actions :

- La premier est dynamique, elle fonctionne directement par la position de référence du moteur (commande), cette action est utilisée pour compenser les retards dynamiques qui peuvent causer des erreurs.
- La deuxième action contient le signe de la commande de vitesse qui est non linéaire et qui est destiné pour la compensation des frottements.

## III.7 PRINCIPALE TACHES EXECUTER PAR LE PROCESSEUR [4], [5] :

Répartie sur chaque cycle SERVO, le processeur exécute en permanence cinq tâches principales que sont :

- Envoi et Réception d'un caractère Simple.
- Mise à jour de la commutation.
- Mise à jour Servo.
- Interruption en temps réel.
- Tâches de fond.

Ces tâches fonctionnent comme des interruptions ordonnées dans un arrangement prioritaire qui a été optimisé pour garder le fonctionnement des applications efficace et sans risque. Le diagramme suivant illustre le fonctionnement des tâches en arrangement prioritaire.

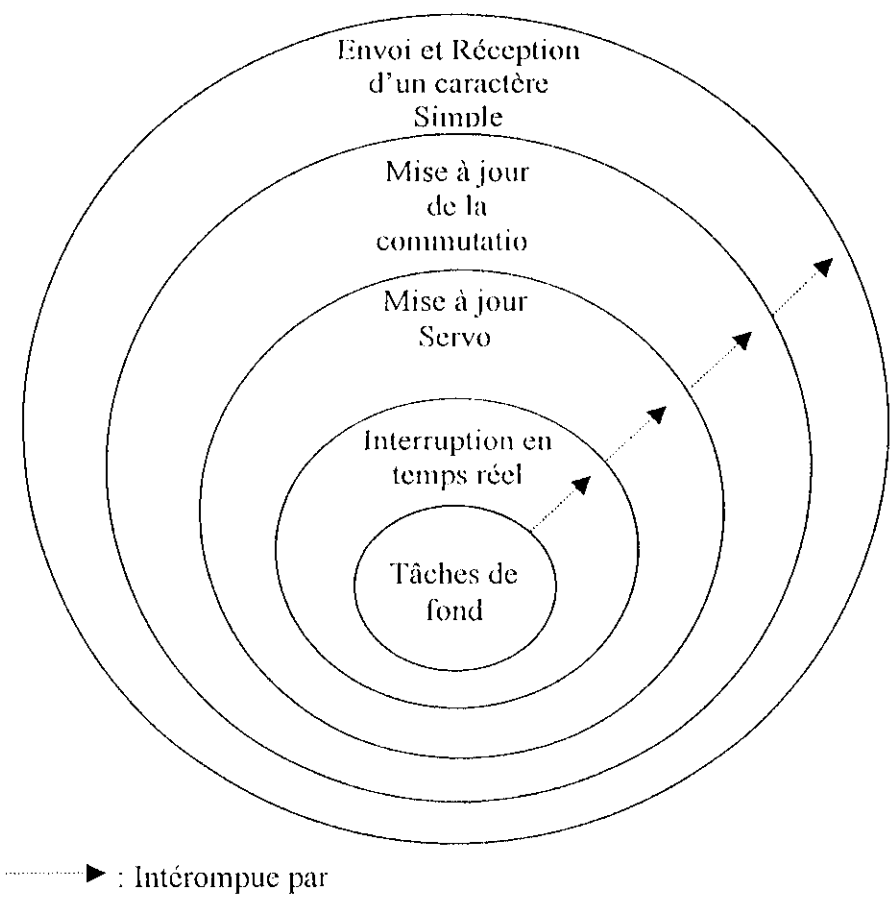



**Figure III.10** : Principe de fonctionnement en arrangement prioritaire

### III.7.1 ENVOI ET RECEPTION D'UN CARACTERE SIMPLE :

Le processeur peut communiquer avec le PC à tout moment, même au milieu d'un ordre des mouvements, l'interface peut recevoir une commande et prends la mesure appropriée en mettant la commande qui était entraîne de s'exécuter dans une mémoire tampon de programme pour une exécution postérieure, fournissant une réponse au PC, puis recommençant un mouvement de moteur etc. si la commande est incorrecte, elle envoi un message d'erreur au PC.

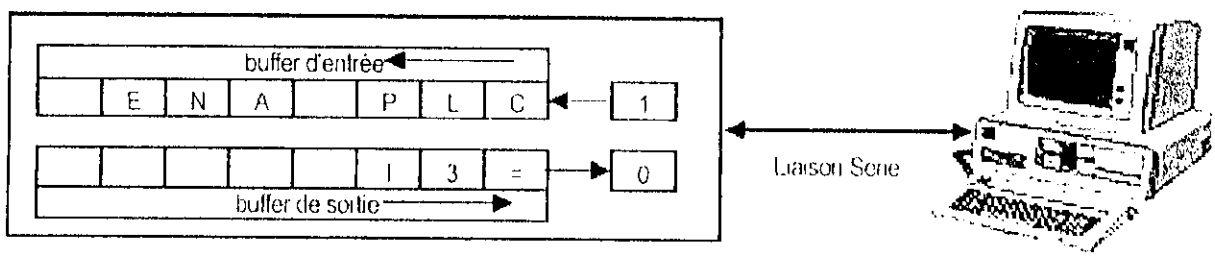



**Figure III.11** : Envoi et Réception d'un caractère Simple

L'opération de réception ou d'envoie d'un caractère simple par l'intermédiaire d'une liaison série, est de priorité la plus élevée dans l'interface, cette tâche dure à peu près quelques 200ns. Ce qui assure le réception tous les caractères émis par l'ordinateur.

### III.7.2 MISE A JOUR DE LA COMMUTATION (*Commutation Update*) :

Si UMAC est configurée pour exécuter la commutation pour un moteur multiphasé, il exécute automatiquement des mises à jour de commutation à une fréquence PHASE fixe, la commutation (ou la mise en phase) pour un moteur comprends la mesure de la position et/ou l'estimation de l'orientation du champ magnétique du rotor, puis la répartition de la commande qui a été calculée par la mise à jour Servo dans les différentes phases du moteur. Cette tâche se produit automatiquement.

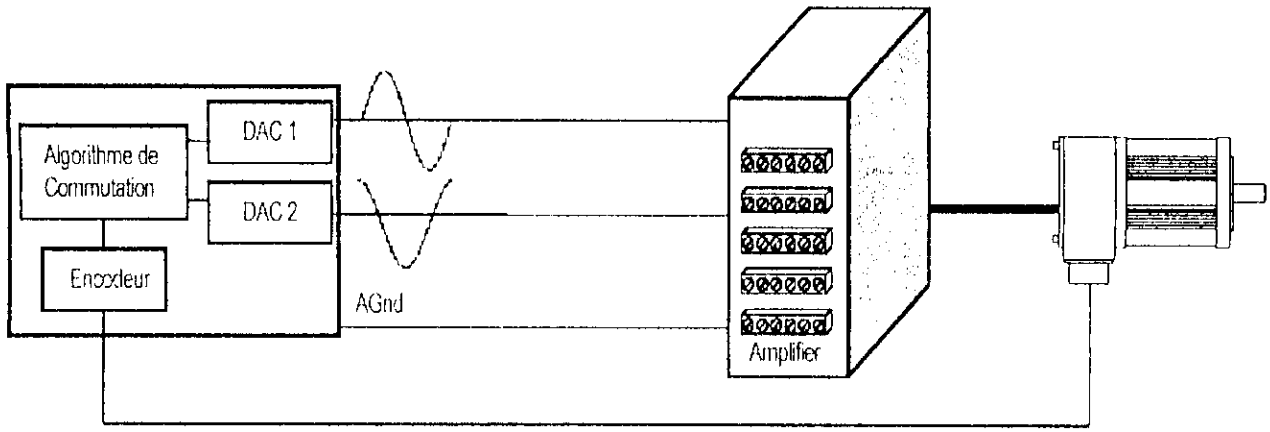



**Figure III.12** : Mise à jour de la commutation

La mise à jour de la commutation (mise en phase) est la deuxième priorité la plus élevée sur l'interface, cette tâche prends environ 3 µs par cycle pour chaque moteur commuté par l'interface (Ix01=1), la fréquence de mise à jour (PHASE) de défaut est 9 kHz (110 µsec par cycle). Par défaut, la commutation de chaque moteur prends approximativement 3% de temps de calcul.

### III.7.3 MISE A JOUR SERVO ( *Servo Update* ) :

C'est une tâche d'asservissement qui est invisible à l'utilisateur, l'interface exécute une mise à jour Servo pour chaque moteur à une fréquence SERVO fixe, en calculant la nouvelle commande de position, lisant la nouvelle position réelle et calculant le résultat de commande basé sur la différence entre les deux, elle se produit automatiquement et c'est la troisième priorité la plus élevée.

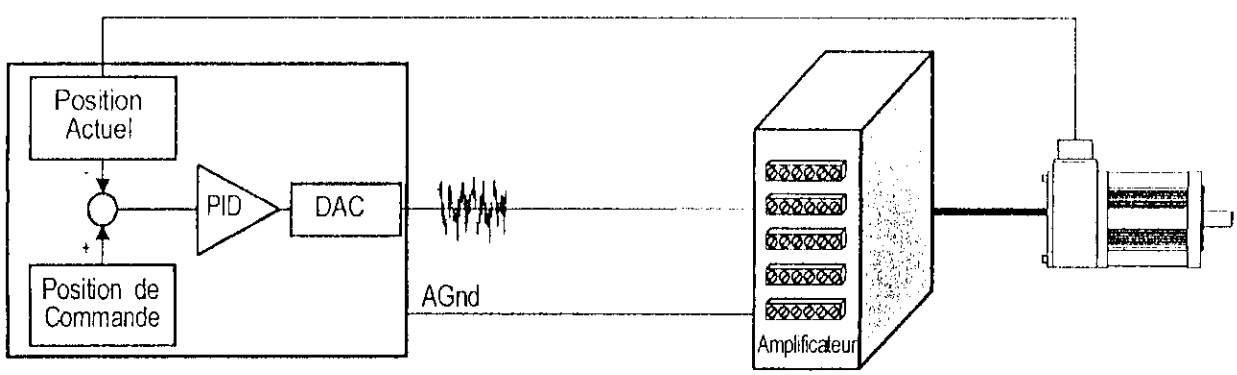



**Figure III.13** : Mise à jour Servo

Pour chaque moteur activé (Ix00=1), cette tâche prends environ 30 µs par cycle SERVO, plus environ 30 µs des tâches de commande générale telles que la table de conversion d'encodeur, la fréquence de mise à jour (SERVO) par défaut est 2,26 kHz (442 µsec par cycle), la mise à jour Servo de chaque moteur prend approximativement 7% de temps de calcul.

### III.7.4 INTERRUPTION EN TEMPS REEL RTI ( *Real-Time Interrupt Tasks* ) :

Les tâches d'interruption en temps réel (RTI) sont la quatrième priorité la plus élevée, elles se produisent immédiatement après les tâches de mise à jour Servo à une cadence contrôlée par le paramètre I8 (après chaque I8+1 cycle de mise à jour Servo ), Il y a deux tâches qui se produisent à ce niveau prioritaire : exécution PLC0 / PLCC0 et planification de mouvement par le programme de mouvement, l'enchaînement des taches RTI est présenté par l'organigramme suivant.

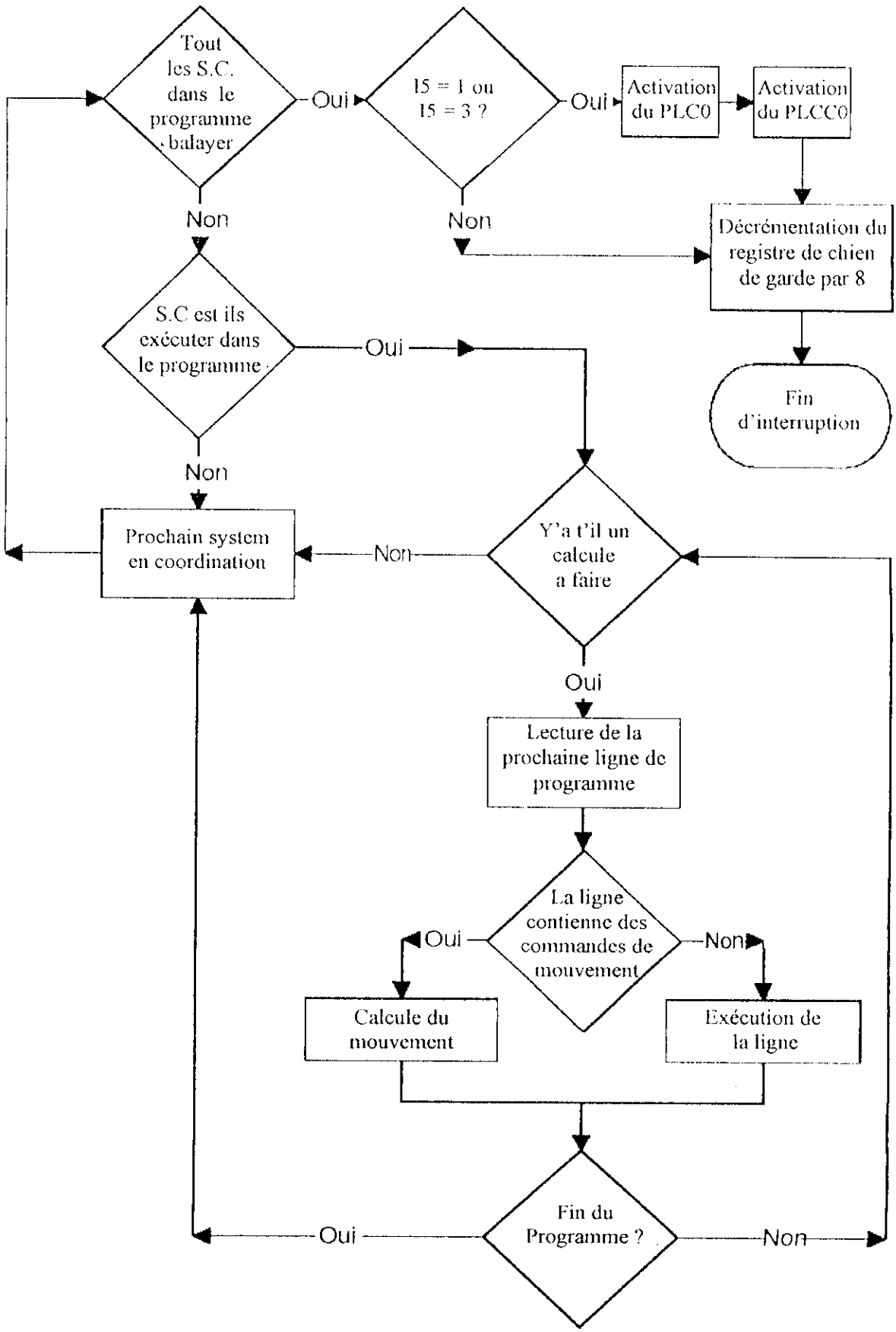



**Figure III.14** : Organigramme de la tâche RTI

### III.7.4.1 la Planification de mouvement :

Elle consiste à balayer chaque ligne du programme de mouvement fonctionnant dans les différents systèmes en coordination, on calculant le nombre nécessaire de commandes de référence.

Le nombre de commandes de mouvement à calculer dépends du type de trajectoire appliquée et du mode dans lesquels on exécute le programme. Les commandes qui ne sont pas de mouvement sont exécutées immédiatement quant il sont trouvées, le balayage de n'importe quel programme donné de mouvement s'arrêtera quant le nombre nécessaire de planifications de mouvements est calculé, et il reprends quand des commandes précédentes de mouvement sont terminées.

Dans l'exécution d'un programme de mouvement, si le processeur trouve deux sauts en arrière (vers le dessus par **ENDWHILE** ou **GOTO**) dans le programme tout en recherchant la prochaine commande de mouvement, il fera une pause et n'essayera pas de mélanger les mouvements ensemble en exécutent les autres tâches, et il continuera l'exécution du programme de mouvement dans un balayage postérieur.

### III.7.4.2 Exécution du programme PLC0/PLCC0 :

C'est un programme PLC (ou PLCC) spécial qui fonctionne à une priorité plus élevée que les autres PLCs, il est censé être utilisé seulement dans très peu de tâches qui sont délicates et qui doivent être faites fréquemment comme le partage du temps entre les Taches, le PLC0 s'exécute à chaque RTI après que la tâche précédente de ce dernier soit terminée. Quant il s'exécute, le registre de chien de garde (*watchdog Timers*) ce décrément par 8. Ce programme peut être désactiver par configuration (I5=0 ou 2), dans ce cas le registre de watchdog Timers ce décrémente par 8 immédiatement après la planification du mouvement.

### III.7.5 TACHES DE FOND (*Background Tasks* ) :

Dans l'absence de tâches de priorité élevée ou (et si I5=2 ou 3), le processeur exécute les tâches de fond (*Background Tasks*), entre outre Répondre aux commandes du PC, Exécution des programmes PLC/PLCC 1-31 et le ménage. La fréquence de ces tâches de fond est contrôlée par le volume des calculs sur le processeur, l'enchaînement des tâches dans ce cas est présenter par l'organigramme suivant :

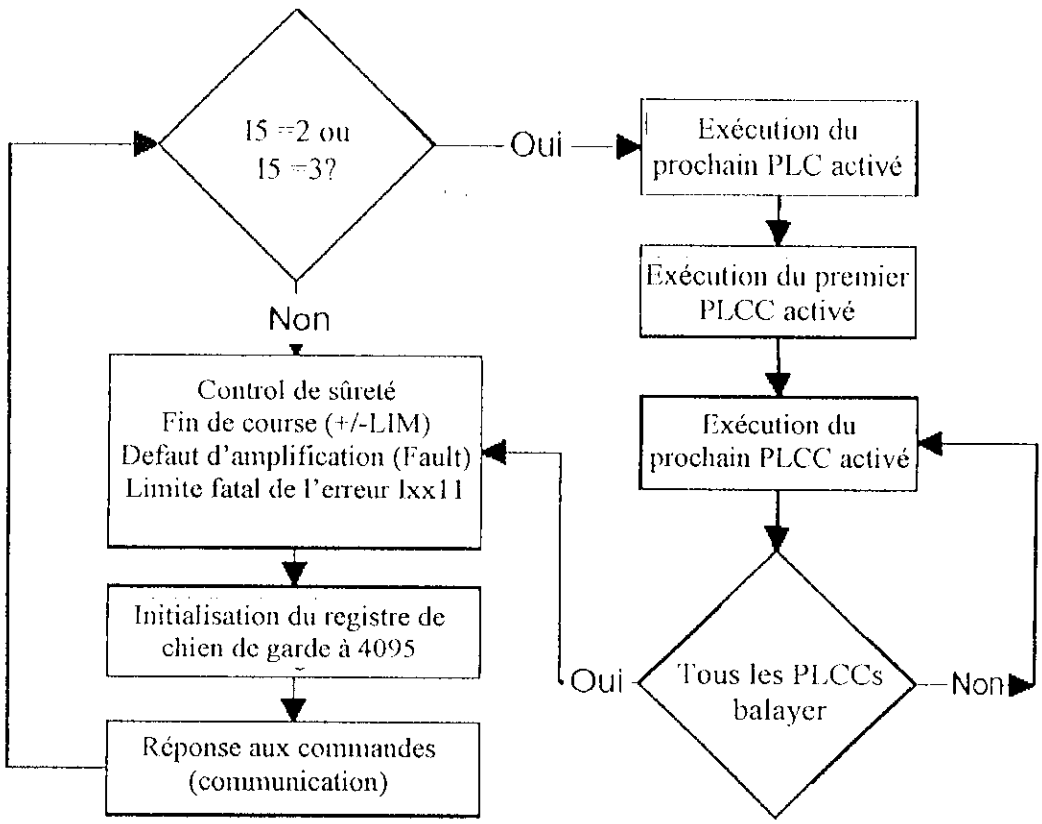



**Figure III.15** : Organigramme des Tâches de fond

#### III.7.5.1 Exécution des programmes PLC1-31 :

Un balayage de chaque programme PLC activé est exécuté (jusqu'à la fin ou à la rencontre d'un ENDWHILE), il n'est interrompu par n'importe quelle autre tâche de fond (bien qu'elle puisse être interrompue par des tâches prioritaires plus élevées), entre chaque programme PLC le processeur fera son ménage général, et réponds à une éventuelle commande de PC.

### III.7.5.2 Exécution des programmes compilés PLCC 1-31 :

Un balayage exécuté pour tous les programmes de PLCC existant (jusqu'à la fin ou la rencontre d'un ENDWHILE) à partir du PLCC numéroté du plus bas au plus haut, ils sont non interrompus par aucune autre tâche de fond (bien qu'elle puisse être interrompue par des tâches prioritaires plus élevées), les programmes PLCCs sont exécutés après l'exécution du premier programme PLC.

### III.7.5.3 Ménage général ( *housekeeping* ) :

Entre chaque balayage des programmes PLC, le processeur exécute ses fonctions de ménage pour assurer que le système est en bon état de marche, ces tâches incluent les contrôles de sûreté, tels que des limites fatales de l'erreur (Ixx11), les limites de dépassement de matériel (±LIM), les limites de dépassement de logiciel et le défaut d'amplification (FLAUT), elles incluent aussi l'initialisation du temporisateur de chien de garde.

Bien que ceci se produise à une basse priorité, une fréquence minimum est assurée par le chronométreur de chien de garde qui se déclenche en arrêtant la carte, si cette fréquence devient trop basse.

### III.7.5.4 Répondre aux commandes du PC :

La réception d'un caractère de commande de n'importe quel port est un signal au processeur qui devrait répondre par une commande, le caractère de commande le plus commun est le retour de chariot (< CR >), qui indique au processeur qu'il doit traiter tous les caractères alphanumériques précédents comme commande en ligne, d'autres caractères de commande ont leurs propres significations, indépendamment de autres caractères alphanumériques reçus.
Ici le processeur prendra les mesure appropriée à la commande, ou si c'est une commande illégale, il reportera une erreur au centre serveur.

## III.8 TEMPS PRIS PAR CHAQUE TACHE [4] :

Par apport au cycle Servo qui a une période 440 µsec de défaut, l'estimation du pourcentage du temps pris par chaque tâche est présentée par le diagramme circulaire suivant :

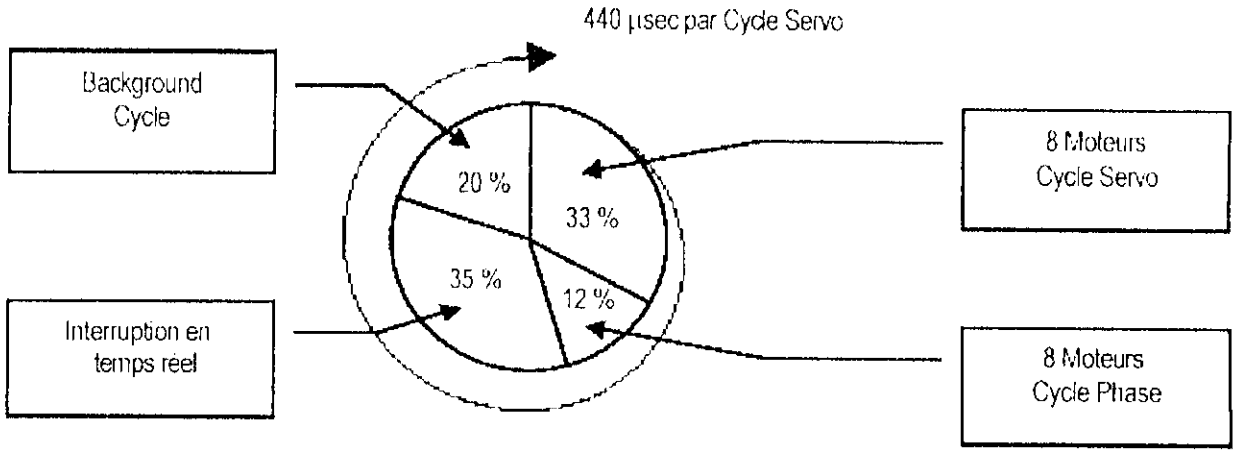



**Figure III.16** : Temps pris par chaque tâche

Pour mieux voir l'enchaînement des tâches exécuté par le processeur, un exemple de huit cycles SERVO (Figure III.17) est donné, dans le cas ou il y'a quatre systèmes en coordination et la commutation est désactivé pour les quatre moteurs.

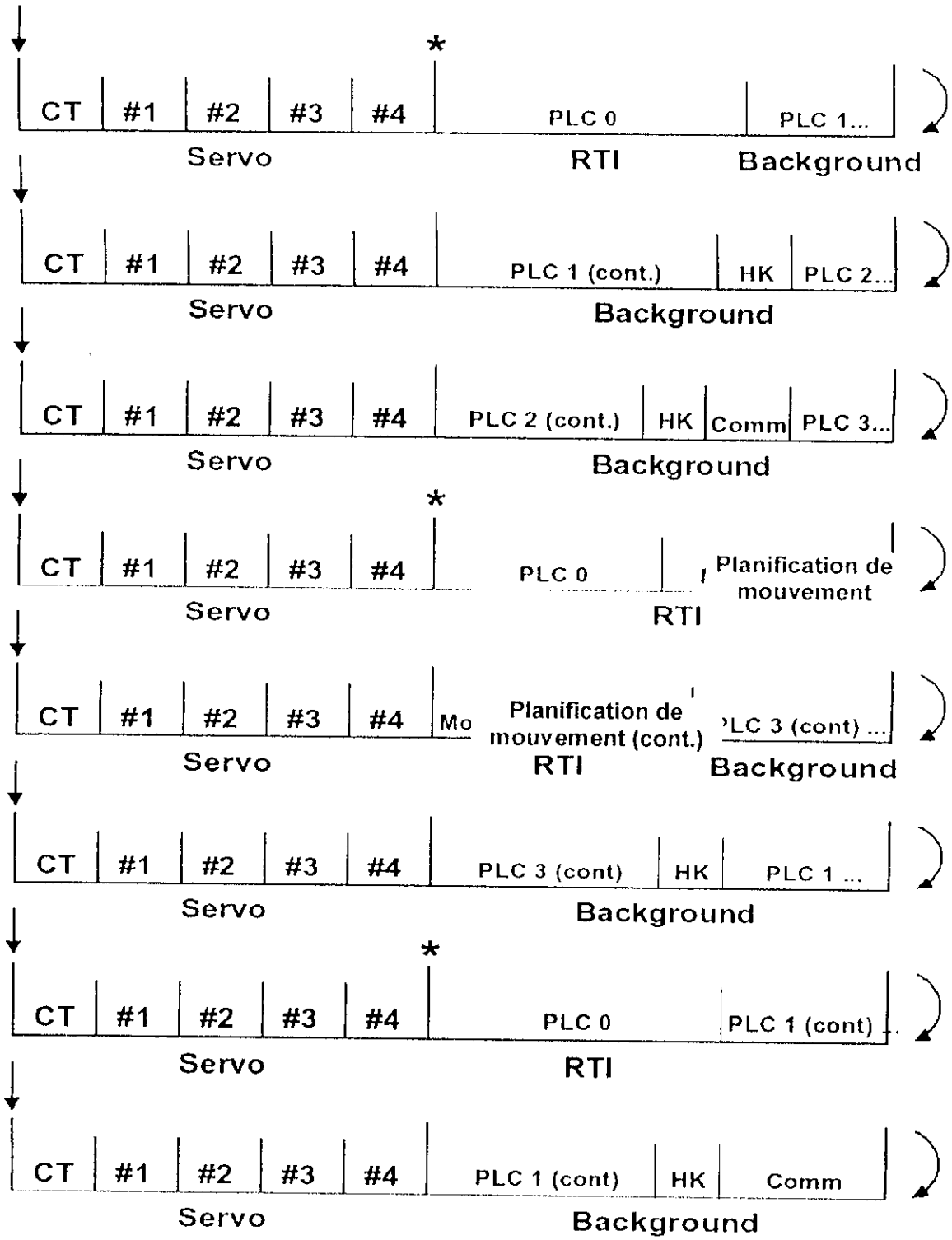



**Figure III.17** : Exemple d'enchaînement des tâches

## III.9 ROLE DU CHIEN DE GARDE [4] :

Si la fréquence de RTI passe au-dessous de 50 hertz environ ou une tâche de fond n'a pas était exécuté au moins à chaque 512 cycles RTI, le chronométreur se déclencherait.
Le but de cette commande en deux parties du temporisateur est de s'assurer qu'on exécute tous les aspects du logiciel.

## III.10 CONCLUSION :

Dans le présent chapitre, on voit bien que l'interface UMAC possède un modèle de fonctionnement assez structuré et robuste, permettent de réalisé des applications sans avoir à se soucier du mauvais fonctionnement interne du système.

# *CHAPITRE IV: Réglage et mise en fonction de l'interface UMAC*

## IV.1 INTRODUCTION :

Après avoir bien détailler les différentes parties matérielles et logiciels de notre interface d'axes UMAC, nous exposerons dans ce présent chapitre la configuration des différents paramètres logiciel apportés à l'UMAC afin de réaliser correctement notre application, ainsi que quelques fonctionnalités typiques de l'interface, tout en se familiarisant avec les logiciels de ce dernier.

## IV.2 INSTALLATION DU PROGRAMME EXECUTIF DE L'INTERFACE UMAC « PEWIN32PRO » :

Avant d'établir la communication « interface UMAC – ordinateur », on doit choisir le quel des deux types de protocole de transmission série disponible dans l'interface (RS-232 ou RS-422) à utiliser. La connexion par défaut devrait être la RS-232, pour changer la connexion on doit reconfigurer les Jumpers E17 et E18 (voire annexe 1).

## IV.3 ETABLIR LA COMMUNICATION AVEC PC :

Pour l'installation ou la désinstallation de l'UMAC sur PC on a besoin d'aucune application, y compris PEWIN32, car le UMAC est considéré comme n'importe quelle périphérique de PC (carte vidéo, carte son, modem, imprimante etc. .), donc son installation se fait simplement par ajout d'un nouveau matériel dans le panneau de configuration, ce logiciel est très récent car c'est la dernière version produite par la firme DELTA TAU, il est compatible avec WINDOWS 98/ME et 2000.

Les différantes étapes à suivre pour l'installation :

- Ajout d'un nouveau matériel dans le panneau de configuration.
- Sélectionné '' Non, le périphérique ne figure pas dans la liste''.
- Sélectionné '' Non, je veux choisir le matériel à partir d'une liste''.
- Sélectionné ''Autres périphériques''.
- Sélectionné ''Delta Tau Data Systems Inc,'' et ''PMAC Serial Port''.
- Le chemin d'accès du pilote est : Windows\System32\Drivers\pmacser.sys.

Si le périphérique est bien installé, on aperçoit le message :

'' a device is configured successfully ''

A présent on re- boot le PC.

**Remarque :** Notons qu'il faut fermer toutes applications développées par Delta Tau lors de la configuration, si elles en existent sur le PC.

## IV.4 CONFIGURATION DU PORT SERIE PAR PEWIN32PRO :

Après l'installation, il faut configurer l'UMAC dans PEWIN32PRO :
Déterminer le PMAC (carte mère de UMAC) principale et les PMACs secondaires s'il en existe d'autres, configurer les propriétés du port série :

- Numéros du port utiliser.
- Vitesse de transmission 38400 bits/sec.
- Parité non.
- Sélectionnée CTS (clear to send).
- Dans le contrôle de RTS (ready to send) sélectionnée enable ect.
- Pour le temps de sortie time out : character 2000 ms.
  Flush 30 ms.

## IV.5 CONFIGURATION NECESSAIRE POUR L'APPLICATION :

Avant de programmé n'importe qu'elle application il est suggéré de faire une configuration matérielle et logiciel associé a l'application désirée.
En ce qui concerne la configuration matérielle, les jumpers sont laissés tel quel sont dans notre application, car ils sont configurer de manière a ce que les cartes fonctionne pour les modes les plus communs.

L'annexe 4 apporte les schémas du câblage associé au branchement des moteurs PITTMAN 9236, qui sont des moteur à courant continue contenant des encodeur incrémentale de 500 CPR (compte par révolution).

Nous présenterons dans ce qui suit, les principales variables pour les applications les plus commune :

### IV.5.1 LES PRINCIPALES VARIABLES I CONCERNANT LES MOTEURS [7] :

Ixx00 : Activation du moteur numéro xx. (Pour activé le moteur 1: I100 =1).

Ixx01 : Activation de la commutation et/ou la boucle digitale du courent pour le moteur xx,
Ixx01 = 0 pas de commutation par UMAC du moteur xx.
Ixx01 = 1 exécuté la commutation (faite pour les moteurs multi-phase), dans ce cas il faut bien configurer le rang des variables Ixx70 - Ixx83, sinon peut importe leur valeur.

Ixx02 : Spécifie l'adresse du registre pour le quel PMAC écrit la commande de sortie pour le moteur. La valeur par défaut contient l'adresse du registre A dans la DSPGATE1.

Ixx03 : Spécifie l'adresse du registre de feedback de la position pour le moteur xx.

Ixx04 : Spécifie l'adresse du registre de feedback de la vitesse pour le moteur xx, c'est la même valeur de Ixx03 si on utilisent pas un dual feedback.
Le calcule de la vitesse ce fait en suite par dérivation numérique de la position.

Ixx24 : C'est une variable très utile pour la configuration et la gestion des interruptions existantes dans l'interface.
le lecteur est invité à voir sa définition bit par bit au fichier de description des variables I (voir annexe 2).

Ixx25 : Spécifie l'adresse du registre des interruptions : d'entrée (+LIMn, -LIMn, HMFLn, FALTn), de sortie (AENA)
Ixx24 spécifie quel types d'interruption est utiliser, et son mode d'activation (front d'activation).

Ixx80 : Spécifie l'état du moteur pour la prochaine mise sous tension, la valeur 0 indique que le moteur ne bouge pas jusqu'à une nouvelle instruction du programmeur
(Usuellement J/,A ou CTRL+A pour Ixx01=0, ou bien $ pour Ixx01=1).

Ixx69 : Détermine la valeur en volt de la sortie du convertisseur DAC
Ixx69 =20480 c'est à dire que le DAC peut envoyer le signale de position dans la gamme ± 6.25 volts.
Ixx69 = 32767 la gamme de sortie du DAC est ± 10 volts
La résolution du DAC est sur 14 bits.

## IV.5.2 LES PRINCIPALES VARIABLES I CONCERNANT LES ENCODEURS [7] :

DSPGATE (IC) m :2-9 Chanel n :1-4

I7mn0 : Contrôle le signe et l'amplitude du comptage de l'encodeur. la majorité des utilisateurs

emploi le décodage x4 pour une meilleure résolution.

I7mn2 : Détermine quel est le signale d'entrée ou la combinaison des signales du canal n, et le front actif, pour la capture de la position HOME.

Ixx97 doit être égale a 0 pour spécifier une capture matérielle.

I7mn3 : Si I7mn2 détermine une utilisation des interruptions matérielles, I7mn3 détermine quelle est l'entrée interruption (généralement on choisie HMFLn ).

I7mn6 : sélectionne le mode de sortie.

I7mn6 = 0 : la sortie A&B est en mode PWM

I7mn6 = 1 : la sortie A&B est en mode DAC

L'application Turbo Setup pro accompagnons le logiciel PEWIN32PRO nous aiderons plus efficacement pour l'initialisation et la configuration des différantes paramètres suivant le matériel et les accessoires de la carte mère Turbo PMAC2, ainsi que les moteurs utilisés.

## IV.5.3 CONFIGURATIONS NECESSAIRES POUR LES MOTEURS PITTMAN 9132:

Le moteur PITTMAN 9132 est à courant continue de plus les amplificateurs que nous disposons travail avec une boucle de courant analogique qui exécute les commutations automatiquement, de ce faite pour placer quatre moteurs PITTMAN 9132 les configurations suivantes sont nécessaires :

I100 = I200 = I300 = I400 = 1 ; activé les quatre premiers moteurs

I101 = I201 = I301 = I401 = 0 ; désactivé la commutation les quatre premiers moteurs.

I7216 = I7226 = I7236 = I7246 =1 ; configuré les quatre canaux de la première DSPGATE en sortie DAC pour que les amplificateurs reçoivent des commandes en courant.

I7210 = I7220 = I7230 = I7240 = 3,ou 7 ; si l'erreur de Limite fatale de l'erreur ( Ixx11 ) se déclenche alors dans ce cas on affaire à une contre réaction positive et il faut inverser la configuration du sens du comptage (3 à 7 ou 7 à 3).
I124 = I224 = I324 = I424 = $20001 ; : si +LIM, -LIM ne sont pas utilisé et ils ne sont pas branché à la masse, il est nécessaire de faire cette configuration, sinon les fins de course se déclenchent et les moteurs ne tourneront pas

Les autres configurations logiciel sont laissés affectées à leur valeur par défaut car ce n'est pas nécessaire pour le fonctionnement normal du système.

## IV.6 UTILISATION DE L'APPLICATION PMAC TUNING PRO :

PMAC Tunig Pro est utilisé pour une configuration et amélioration des différentes boucles d'avertissement de l'interface, il sert à contrôler les divers aspects du réglage des moteurs. Pour l'utilisation de ce dernier, on procédera par les étapes suivantes :

### IV.6.1 CALIBRAGE DE LA SORTIE DU CONVERTISSEUR DAC :

Dans le menu : Position Loop \ DAC Calibration.
Le but de cette option est de fournir pour l'utilisateur la capacité de déterminer la valeur du biais pour le DAC qui affectera les variables Ixx29 et Ixx79. Ce réglage apporte plus de précision a l'offset réglé manuellement à partir des amplificateurs.

### IV.6.2 TESTE DE LA REPONSE DU MOTEUR :

Dans le menu : Position Loop \ Open Loop Test.
On peut tester la réponse du moteur en boucle ouvert par un signale de référence carrée périodique dont on peut déterminer le nombre de répétitions.

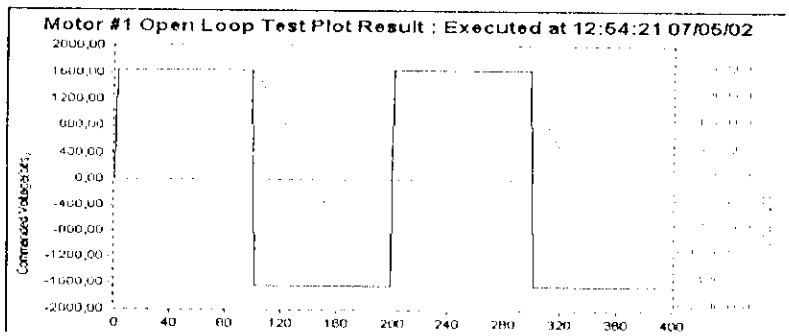



**Figure IV.1 :** Réponse du moteur en boucle ouverte

### IV.6.3 AUTO ESTIMATION DES PARAMETRES DU REGULATEUR PID :

Dans le menu Position Loop \ Regular PID \ Auto Tuning.

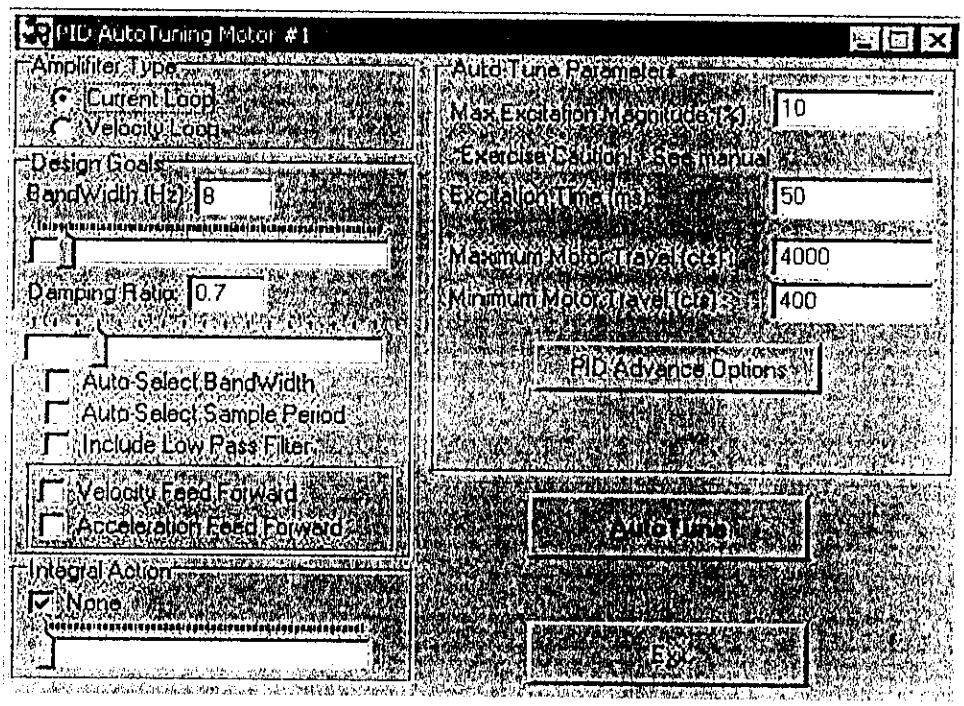



**Figure IV.2 :** Fenêtre PID Auto Tuning

- Sélectionné Auto-select Bandwidth si vous ne savez pas choisir une valeur raisonnable pour évaluez la largeur de bande de la boucle fermé.
- Sélectionné Damping Ratio entre 0.8 à 1.0.
- A présent Sélectionné Auto Tune, cette routine vas faire une première estimation des valeurs optimale des paramètres du régulateur PID.

- Implémenter en suite ces valeurs dans l'interface UMAC, ceci affectera une gamme des variables I : Ixx30 – Ixx35.

### IV.6.4 SIMULATION DES DIFFERANTE TRAJECTOIRES :

Dans le menu Position Loop \ Regular PID \ Interactive Tunig.

Après un test de l'auto régulation, avec Step position par exemple, on remarque que la position commandée ne suit pas exactement la consigne.

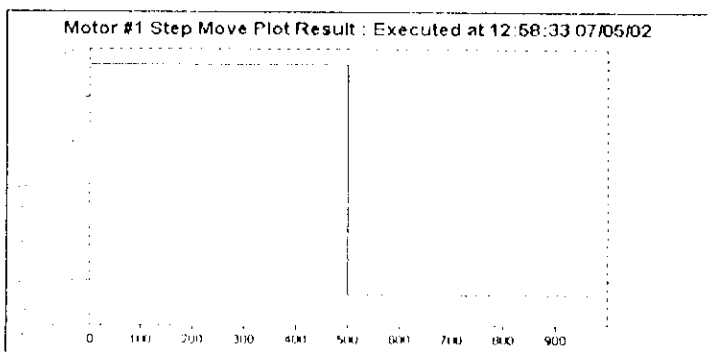



**Figure IV.3 :** Réponse du moteur a un échelon en boucle fermé avec un auto réglage

Donc on procédera à la régulation interactive (selon l'Annexe 3), jusqu'à avoir des réponses satisfaisantes, puis implémenté les paramètres dans l'interface a l'aide de la fenêtre du réglage '' PID Interactive Tuning'' suivante :

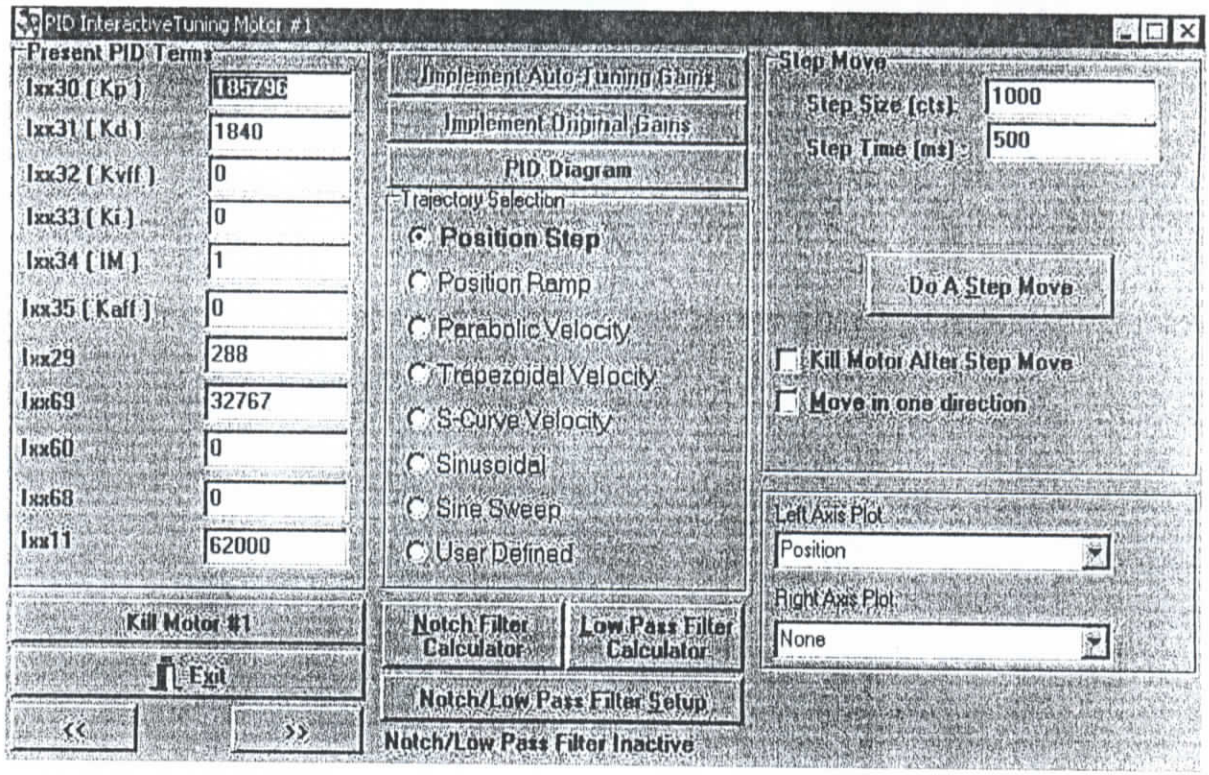



**Figure IV.4 :** Fenêtre PID Interactive Tuning

On va changer les valeurs des paramètres Kp, Kd, Ki, Kvff, et Kaff suivant la méthode illustrée dans le chapitre précédent :

- Tout d'abord on règle les paramètres Kp, Kd, Ki avec le test du Position Step, d'où on a abouti au résultat suivant :

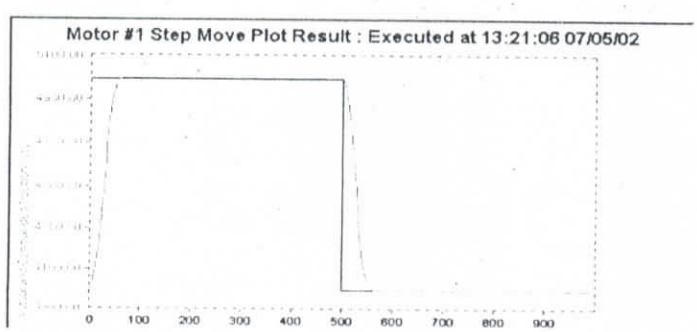



**Figure IV.5 :** Réponse du moteur a un échelon après réglage interactive

Le régalage de ces paramètres du correcteur PID n'est pas suffisant pour avoir une bonne poursuite de la trajectoire rampe – erreur en vitesse – le modèle mathématique du moteur a courant continue n'est pas de classe 1 (ne possède pas un intégrateur propre).

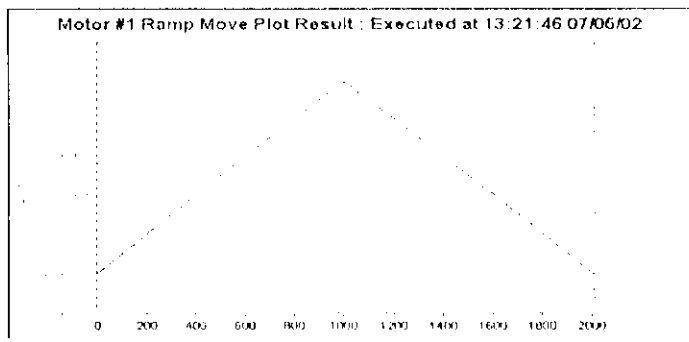



**Figure IV.6 :** Réponse du moteur a une rampe avant l'ajustement du régulateur FeeForword

- Les coefficients Kvff, et Kaff vont être estimé avec en utilisant la trajectoire Parabolic Velocity, cet ajustement va augmenté la précision de la poursuite.

Kvff réduction des erreurs en régime d'équilibre (erreur en vitesse ).
Kaff offre une bonne poursuite a une variation brusque de vitesse.
En faite ces coefficients sont intégré pour un réglage feedforward, qui est un réglage de compensation des retards dynamiques (effectué sur la vitesse et l'accélération), et qui consiste a introduire la variation de ces dernier à la commande d'une façon non- causale 'passé directement a la sortie', sans avoir a comparer avec la position ou la vitesse du feedback.

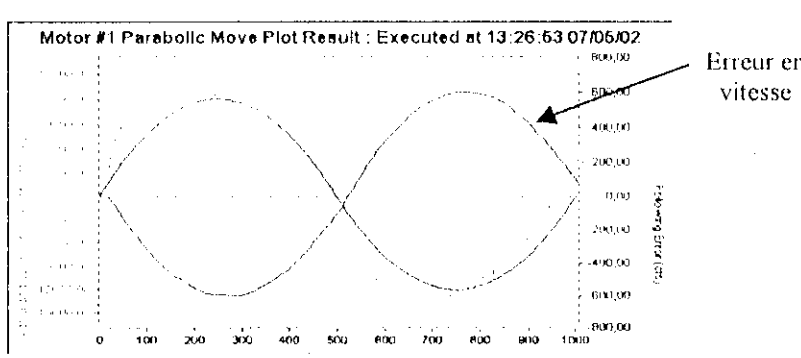



**Figure IV.7 :** Réponse du moteur avant réglage de Kvff , Kaff à une consigne de vitesse sinusoïdale

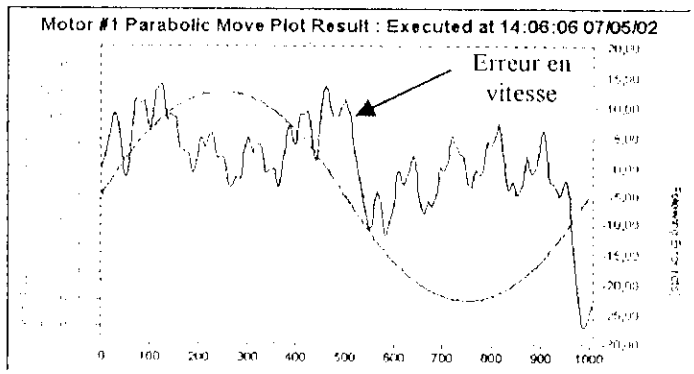



**Figure IV.8 :** Réponse après réglage interactive des coefficient FeedForward

On remarque que la poursuite de la variation de vitesse est de très grande précision, avec une variation d'erreur de moyenne nulle.

A présent, la poursuite d'une rampe est presque sans erreur :

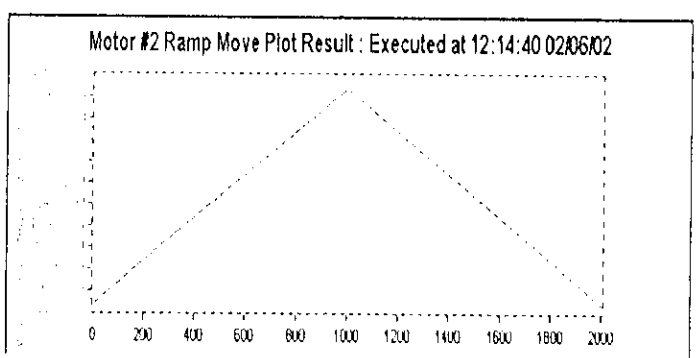



**Figure IV.9 :** Réponse du moteur a une rampe après l'ajustement du régulateur FeeForword

On peut tester toutes les trajectoires disponibles dans cette section, y compris la génération des trajectoires usuelles tel que : LINEAR, Spline1, PVT.

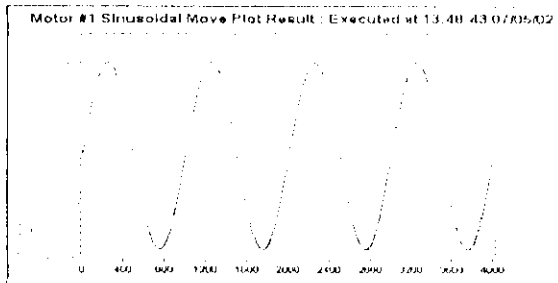



**Figure IV.10.a :** Mouvement sinusoïdale

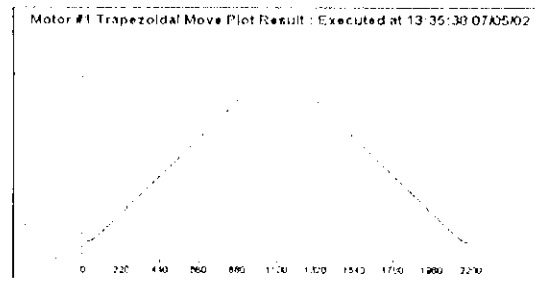



**Figure IV.10.b :** Mouvement trapézoïdale

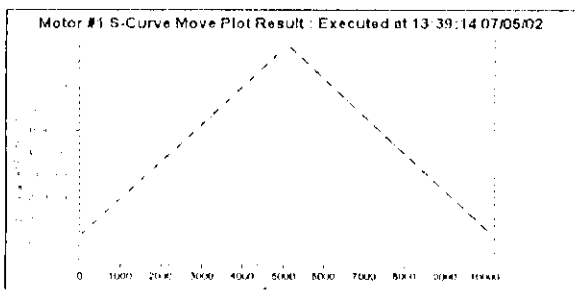



**Figure IV.10.c :** Mouvement S- courbe

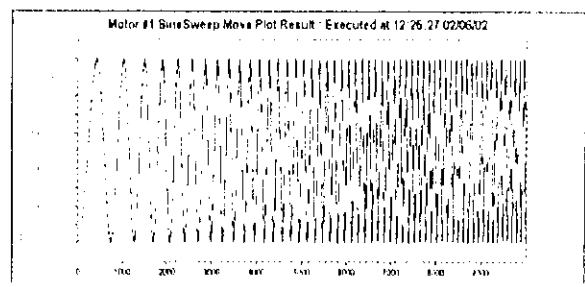



**Figure IV.10.d :** Mouvement sinsweep

Le mouvement SineSweep consiste à augmenter la fréquence au fur et mesure que le mouvement évolue, dans notre cas on à augmenter la fréquence de 10 fois.

## IV.6.5 FILTRE REJECTEUR ( NOTCH FILTER) ET LE FILTRE PASSE BAS ( LOW PASS FILTER ) :

Ces deux filtres sont utilisé pour enlever la fréquence de résonance mécanique, et rejeter les fréquences de perturbation, qui sont généralement dans les hautes fréquences, donc ils doivent se trouver apprêt la fréquence de coupure du filtre passe bas.

Mais, étant donnée que la structure mécanique du robot n'est pas utiliser, on ne peut pas connaître la fréquence de résonance mécanique de la structure, alors on n'a pas utilisé ces filtres.

## IV.7 UTILISATION DU LOGICIEL PEWIN32PRO :

### IV.7.1 EDITEUR DE PROGRAMME DE MOUVEMENT ET PLC :

Dans le menu File \ New.

Dans ce menu, on a accès a l'édition de texte du programme de mouvement ou du programme PLC, qui est télécharger en suite a UMAC par la commande Download qui se trouve dans le menu de l'éditeur de texte.

### IV.7.2 LE CONTENUE DU MENU VIEW :

- Une section Terminal dont on peut faire entrer les commandes en ligne.
- Il permet la visualisation de variation de la position en temps réel.
- Des fenêtres de visualisation des registres d'état des moteurs.
- Il contient aussi une section pour l'essai du moteur 'Jog Ribbon'

  Cette dernière section nous permet d'essayer tout les moteurs branchés a l'interface, ainsi de les arrêter indépendamment les un des autres 'Abort' ou bien au même moment 'Kill'.

### IV.7.3 LA FONCTION PMAC PLOT32 PRO :

Dans le menu de PEWIN32PRO : Tools\ Pmac Plot.

Pour faire analyser les trajectoires programmées, on utilise l'application Pmac Plot32 Pro, cette fonction nous assure le traçage de mouvement effectué par chaque articulation.

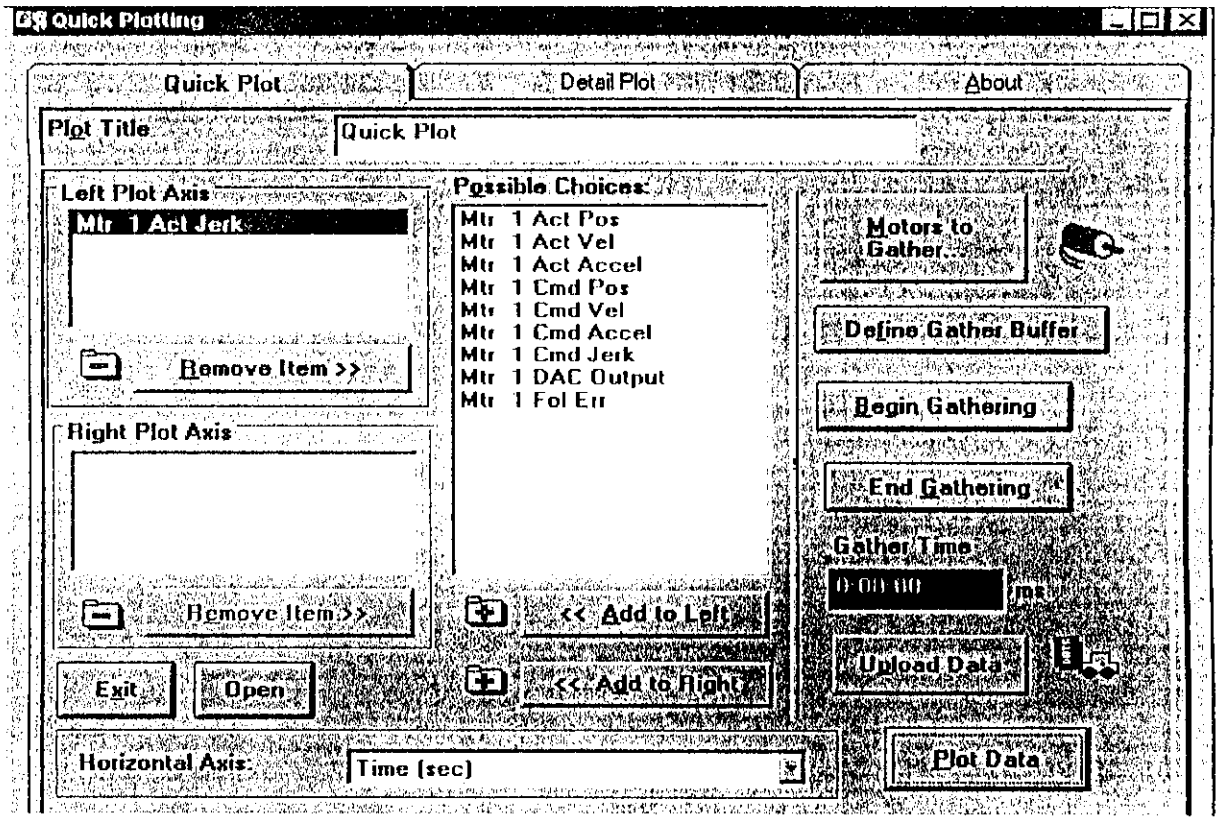



**Figure IV.11 :** Fenêtre Quick Plotting

- Tout d'abord il faut sélectionner quel profil de vitesse est choisie (position, vitesse, ect) qui sont sélectionnés dans zone : Possible Choices, la trajectoire sélectionnée est ajoutée à gauche par <<Add to Left.
- Sélectionné : Motors to gather
- Sélectionné : Define Gather Buffer
- Sélectionné : Begin Gathering
- A présent il faut exécuter le programme de mouvement dans la fenêtre Terminale depuis PEWIN32PRO avec la commande &n Bm R :
- Où n : numéro du système en coordination.

  m : numéro du programme a exécuter.

  Bm : ce brancher au début du programme m.

  R : exécuter le programme m.
- Sélectionné : End Gathering
- Sélectionné : Upload data (ramener les donnée au PC)
- Sélectionné : Plot Data (tracer la trajectoire)

On peut apercevoir maintenant le profil de la trajectoire à analyser.

**Remarque :** On a bien préciser le numéro du système en coordination pour l'exécution du programme, cela nous amène à dire que les moteurs doivent être affecté chacun a son système en coordination, cela peut être fait dans la Barre d'outil du logiciel PEWIN32PRO au menu : Configure\ coordinate systems.

## IV.8 EXEMPLE EXECUTION DES SYSTEMES EN COORDINATION :

Maintenant, on illustre la signification explicite de l'exécution des systèmes en coordination à travers l'exemple de programme suivant :

Supposons que les moteurs #1 et #2 sont dans le système en coordination &1
et que les moteurs #3 et #4 sont dans le système en coordination &2
On peut utiliser le programme suivant pour les deux système en coordination :

```
CLOSE
DELETE GATHER
UNDEFINE ALL
#1-> 363.89X
#2-> 363.89Y
OPEN PROG 1 CLEAR
LINEAR      ; mode linéaire
INC         ; mode incrémantale
TA100       ; temps d'accélération = 100 msec.
TS30        ; temps de courbure au bout du profil de vitesse = 30 msec.
TM1000      ; temps de mouvement = 1 sec.
X90Y75      ; le moteur #1 fait une rotation de 90° et #2 fait 75°, au méme moment
            ; (car ils sont dans la méme ligne).
CLOSE
```

L'exécution du programme 1 pour chaque système en coordination donne les résultats suivants :

&1B1R : le moteur #1 fait une rotation de 90°
le moteur #2 fait une rotation de 75°
&2B1R : le moteur #3 fait une rotation de 90°
le moteur #4 fait une rotation de 75°

Notons que les deux commandes d'exécution sont indépendantes, ils peuvent s'exécuter simultanément « &1B1R&2B1R » ou l'un après l'autre « &1B1R <CR>, &2B1R »<CR>.

## IV.9 IMPLEMENTATION DU MODELE GEOMETRIQUE D'UN ROBOT :

### IV.9.1 EXEMPLE D'IMPLEMENTATION DU MODELE GEOMETRIQUE DIRECTE D'UN ROBOT A DEUX DEGRES DE LIBERTE (MGD) :

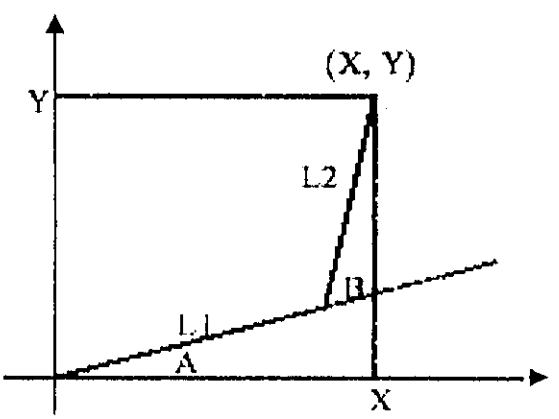



**Figure IV.12 :** Représentation cartésienne du robot

Les coordonnées géométriques de la position de l'effecteur (la pointe du bras de robot) sont obtenues par la projection des axes du robot sur le plan XY, ils sont représentés par le système d'équations suivant :

$$\begin{cases} X = L_1 \cos(A) + L_2 \cos(A+B) \\ Y = L_1 \sin(A) + L_2 \sin(A+B) \end{cases} \qquad (IV.1)$$

Suivant les équations du système (IV.1), le programme d'implémentation du modèle est le suivant :

```
I15 = 1          ; les calcules Trigonométriques en degré
&1               ; Adressé SC 1
Q91= 400         ; longueur L1 (mm)
Q92=300          ; longueur L2 (mm)
Q93= 363.889     ; Compte par degré (pour A et B)

&1 OPEN FORWARD
CLEAR                                              ; effacer le contenue précèdent
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
```

### IV.9.2 EXEMPLE DU MODELE GEOMETRIQUE INVERSE DU ROBOT (MGI) :

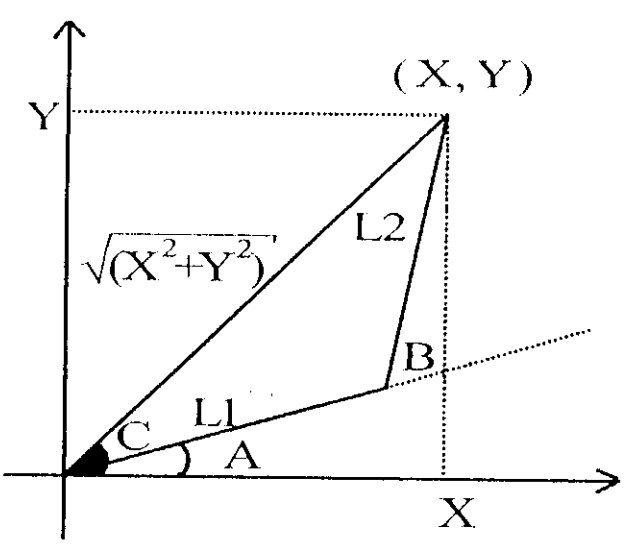



**Figure IV.13 :** Représentation angulaire du robot

Les équations de l'espace articulaire en fonction des coordonnées cartésiennes sont :

$$\begin{cases} B = \cos^{-1}\left(\dfrac{X^2 + Y^2 - L_1^2 - L_2^2}{2L_1L_2}\right) \\ A + C = a\tan 2\,(Y,X) \\ C = \cos^{-1}\left(\dfrac{X^2 + Y^2 + L_1^2 - L_2^2}{2L_1\sqrt{X^2 + Y^2}}\right) \\ A = (A + C) - C \end{cases} \qquad (IV.2)$$

Avec : Atang2 (y, x) = sin (y) / cos (x).

Le programme du modèle géométrique inverse du bras de robot qui correspond au système décrit par l'équation (IV.2) est le suivant :

```
&1
#1->1          ; Moteur 1 affecté au modèle géométrique cinématique inverse (MGI) dans SC 1
#2->1          ; Moteur 2 affecté au MGI dans SC 1
M5182->Y:$00203F,22,1          ; SC 1 bit indicateur du temps de calcule
; les constantes du système
Q94=Q91*Q91+Q92*Q92 ; L1^2 + L2^2
Q95=2*Q91*Q92 ; 2*L1*L2
Q96=Q91*Q91-Q92*Q92 ; L1^2 - L2^2

&1 OPEN INVERSE
CLEAR
Q20=Q7*Q7+Q8*Q8                ; X^2+Y^2
Q21=(Q20-Q94)/Q95              ; cos (B)
IF (ABS(Q21)<0.9998)           ; si la variation de B est dans le voisinage [-1°,1°]
                               ; ou [-179°,179°] (le bras est tendu au maximum ?
  Q22=ACOS(Q21)                ; B (deg)
  Q0=Q7                        ; X dans cos argument for ATAN2
  Q23=ATAN2(Q8)                ; A+C = ATAN2(Y,X)
  Q24=ACOS((Q20+Q96)/(2*Q91*SQRT(Q20)))      ; C (deg)
  Q25=Q23-Q24                  ; A (deg)
```

```
  P1=Q25*Q93                ; Moteur 1 = 363.889*A
  P2=Q22*Q93                ; Moteur 2 = 363.889*B
ELSE
  M5182=1                   ; déclencher erreur de temps de calcule
ENDIF
CLOSE
```

Cet exemple arrête le programme dans le cas où le bras du robot serait tendu au maximum (cos (B) ≈ 0 donc X=L1 et Y=L2). Il fait ceci en activant le bit d'état "*run-time error*" qui corresponde à une erreur d'exécution, pour le système en coordination a 1, qui cause l'arrêt automatique de l'exécution du programme de mouvement (en activant la commande Abort : commencé à décéléré immédiatement).

### IV.9.3 EXECUTION D'UN EXEMPLE DE PROGRAMME DE MOUVEMENT POUR LE ROBOT :

Une fois que les modèles MGD et MGI seront implémenté dans l'interface, et que le calcule géométrique est activé (par la variable I5150 = 1), n'importe quelle programme exécuté pour le système en coordination &1 subira cette transformation géométrique. On exécute maintenant l'exemple suivant :

```
CLOSE
DELETE GATHER
UNDEFINE ALL
#1->100X
#2->100Y
OPEN PROG 1 CLEAR
LINEAR      ; mode linéaire
INC         ; mode incrémentale
TA100       ; temps d'accélération = 100 msec.
TS30        ; temps de courbure au bout du profil de vitesse = 30 msec.
TM1000      ; temps de mouvement = 1 sec.
X0.1Y1      ; la position consigne à suivre pour le bras du robot est (0.1,1)
CLOSE
```

Après l'exécution de ce programme pour le système en coordination &1, le bras du robot fait la translation appropriée jusqu'à la position (0.1,1).

Cela dit, l'utilisateur doit donner les destinations de l'outil finale suivant l'équation $X^2 + Y^2 < (L_1+L_2)^2$ qui indique l'espace opérationnelle du robot, sinon l'exécution du programme vas être erroné (le bit d'état *run-time error* est activé).

**Remarque :** Le point initial de la trajectoire est critique car les articulations sont à la position zéro, alors pour l'exécution du programme on doit ajouter un offset aux articulations de telles façons qu'ils sortent de la zone critique.

### IV.9.4 EXEMPLE D'EXECUTION DU PROGRAMME DE ROBOT SOUS MODE PVT :

Si on prend le même exemple précèdent, on a les expressions des vitesses angulaires suivantes :

$$\begin{cases} \dot{A} = \dfrac{L_2\cos(A+B)\dot{X} + L_2\sin(A+B)\dot{Y}}{L_1 L_2 \sin B} \\ \dot{B} = \dfrac{\left[-L_1\cos A - L_2\cos(A+B)\right]\dot{X} + \left[-L_1\sin A - L_2\sin(A+B)\right]\dot{Y}}{L_1 L_2 \sin B} = \dfrac{-\dot{X}\,\dot{X} - \dot{Y}\,\dot{Y}}{L_1 L_2 \sin B} \end{cases} \quad (IV.3)$$

Notons que les vitesses deviennent infinies quand l'angle B approche 0° ou 180° (les points singuliers du robot), dans l'exemple suivant, une erreur d'exécution est créée si on est trop près d'une singularité.

```
&1
OPEN INVERSE CLEAR
;le calcule de position est fait dans le programme précèdent MGD
IF (Q10=1)                  ; mode PVT?
  Q26=SIN(Q25)              ; sin (B)
  IF (Q26>0.0175)           ; n'est pas proche de la singularité ?
    Q27=Q91*Q92*Q26         ; L1*L2*sinB
    Q28=COS(Q25+Q22)        ; cos (A+B)
    Q29=SIN(Q25+Q22)                    ; sin (A+B)
```

```
    Q30=(Q92*Q28*Q17+Q92*Q29*Q18)/Q27  ; dA / dt
    Q31=(-Q7*Q17-Q8*Q18)/Q27           ; dB / dt
    P101=Q30*Q93          ; vitesse du moteur #1
    P102=Q31*Q93          ; vitesse du moteur #2
  ELSE                    ; proche de la singularité
    M5182=1               ; déclencher erreur de temps de calcule
  ENDIF
ENDIF
CLOSE
```

Ce programme permet un meilleur ajustement pour les vitesses angulaires, cela est fait en mentionnant les équations mathématiques des vitesses angulaires dans le programme du modèle inverse du robot, et en ajoutant les conditions sur les points de singularités.

## IV.10 CONCLUSION :

Pour pouvoir exécuter un programme de gestion de la trajectoire d'un robot dans l'espace cartésien, on doit d'abord écrire, un et un seul, programme du modèle géométrique direct et inverse du robot, et l'implémenter dans la carte d'axes, sans oublier d'activer le processus du calcul automatique des coordonnées géométriques (Isx50 1).

A présent, n'importe quel programme de mouvement peut effectuer le déplacement du robot dans l'espace cartésien, sauf qu'il reste à déterminer la graduation correcte des axes selon les longueurs et les dimensions réels du robot, pour avoir un déplacement cartésien gradué dans les unités désiré.

## Conclusion Générale :

L'objectif de notre travail est l'étude et la mise en marche d'une interface d'axes industriels en vue de la commande d'un bras de robot manipulateur.

L'identification de la documentation fournie sur le site web de la firme de fabrication de l'interface **DELTA TAU,** était très difficile principalement à cause des appellations commerciales de chaque carte composant l'interface UMAC, et à la nature des documents disponibles de l'interface souvent des manuels à vocation commerciale ne proposant pas d'explication technique détaillée sur les produits de la firme.

La détermination des différentes cartes composant l'interface UMAC et leurs fonctionnalités a été rendue possible grâce au seul document trouvé sur Internet contenant la description des produits de la firme **DELTA TAU**, ce qui nous a considérablement aidé dans notre étude.

L'exploration des fonctionnalités de l'interface UMAC nous a permis de commander quatre moteurs dans les différents modes de trajectoires possibles et d'implémenter un modèle géométrique direct et inverse d'un robot à deux degrés de liberté.

Vu les capacités et les performances de l'interface, en perspective nous proposons des études plus poussées des fonctionnalités de l'interface UMAC particulièrement la manipulation des Contrôleurs Programmables Logiques (PLC) dans la gestion de l'interface, et l'utilisation de la carte d'entrées/sorties digitale IOGATE implémentant un automate de gestion de l'environnement du robot.

# *ANNEXE 1: Référence matérielle de l'interface UMAC*

## Connecteur RS232/422 [2] :

**J7: RS-232/422 Serial Port Connector**

(26-pin Header at Location A-2)


Front View

| Pin # | Symbol | Function | Description | Notes |
|---|---|---|---|---|
| 1 | CHASSI | Common | Turbo PMAC2-3U Common | |
| 2 | S+5V | Output | +5VDC Supply | Deactivated by "ES" |
| 3 | RD-/RXD- | Input | Receive Data | Diff. low TRUE / low TRUE |
| 4 | RD+ | Input | Receive Data | Diff. high TRUE |
| 5 | SD-/TXD- | Output | Send Data | Diff. low TRUE / low TRUE |
| 6 | SD+ | Output | Send Data | Diff. high TRUE |
| 7 | CS+/CTS | Input | Clear to Send | Diff. high TRUE / high TRUE |
| 8 | CS- | Input | Clear to Send | Diff. low TRUE |
| 9 | RS+/RTS | Output | Request to Send | Diff. high TRUE / high TRUE |
| 10 | RS- | Output | Request to Send | Diff. low TRUE |
| 11 | DTR | Bidirect | Data Terminal Ready | Shorted to DSR |
| 12 | INIT/ | Input | Turbo PMAC2-3U Reset | Low is "RESET" |
| 13 | GND | Common | Turbo PMAC2-3U Common | |
| 14 | DSR | Bidirect | Data Set Ready | Shorted to DTR |
| 15 | SDIO- | Bidirect | Special Data | Diff. I/O low TRUE |
| 16 | SDIO+ | Bidirect | Special Data | Diff. I/O high TRUE |
| 17 | SCIO- | Bidirect | Special CTRL. | Diff. I/O low TRUE |
| 18 | SCIO+ | Bidirect | Special CTRL. | Diff. I/O high TRUE |
| 19 | SCK- | Bidirect | Special Clock | Diff. I/O low TRUE |
| 20 | SCK+ | Bidirect | Special Clock | Diff. I/O high TRUE |
| 21 | SERVO- | Bidirect | Servo Clock | Diff. I/O low TRUE |
| 22 | SERVO+ | Bidirect | Servo Clock | Diff. I/O high TRUE |
| 23 | PHASE- | Bidirect | Phase Clock | Diff. I/O low TRUE |
| 24 | PHASE+ | Bidirect | Phase Clock | Diff. I/O high TRUE |
| 25 | GND | Common | Turbo PMAC2-3U Common | |
| 26 | +5V | Output | +5VDC Supply | Power Supply Out |

**Figure A1.2 : Connecteur du port série RS-232/422**

## Connecteur d'UBUS [12] :

Ce connecteur contient 96 Pin décomposé sur 3 rangées, d'un format 3U, permettent l'interfaçage avec l'UBUS.

Les Pins d'entrées / sorties du connecteur sont illustrés dans la table suivante :

| Pin # | Row A | Row B | Row C |
|---|---|---|---|
| 1 | +5Vdc | +5Vdc | +5Vdc |
| 2 | GND | GND | GND |
| 3 | BD01 | RFU* | BD00 |
| 4 | BD03 | RFU* | BD02 |
| 5 | BD05 | RFU* | BD04 |
| 6 | BD07 | RFU* | BD06 |
| 7 | BD09 | RFU* | BD08 |
| 8 | BD11 | RFU* | BD10 |
| 9 | BD13 | RFU* | BD12 |
| 10 | BD15 | RFU* | BD14 |
| 11 | BD17 | RFU* | BD16 |
| 12 | BD19 | RFU* | BD18 |
| 13 | BD21 | RFU* | BD20 |
| 14 | BD23 | RFU* | BD22 |
| 15 | BS1 | RFU* | BS0 |
| 16 | BA01 | RFU* | BA00 |
| 17 | BA03 | RFU* | BA02 |
| 18 | BX/Y | RFU* | BA04 |
| 19 | CS3- | BA06 | CS2- |
| 20 | BA05 | BA07 | CS4- |
| 21 | CS12- | BA08 | CS10- |
| 22 | CS16- | BA09 | CS14- |
| 23 | BA13 | BA10 | BA12 |
| 24 | BRD- | BA11 | BWR- |
| 25 | BS3 | MEMCS0- | BS2 |
| 26 | WAIT- | MEMCS1- | RESET |
| 27 | PHASE+ | IREQ1- | SERVO+ |
| 28 | PHASE- | IREQ2- | SERVO- |
| 29 | ANALOG GND | IREQ3- | ANALOG GND |
| 30 | -15Vdc | PWRGUD | +15Vdc |
| 31 | GND | GND | GND |
| 32 | +5Vdc | +5Vdc | +5Vdc |

RFU* : Habituellement déterminé à la carte mère, et réservé pour un futur usage.

## Turbo PMAC2 3U CPU [2] :

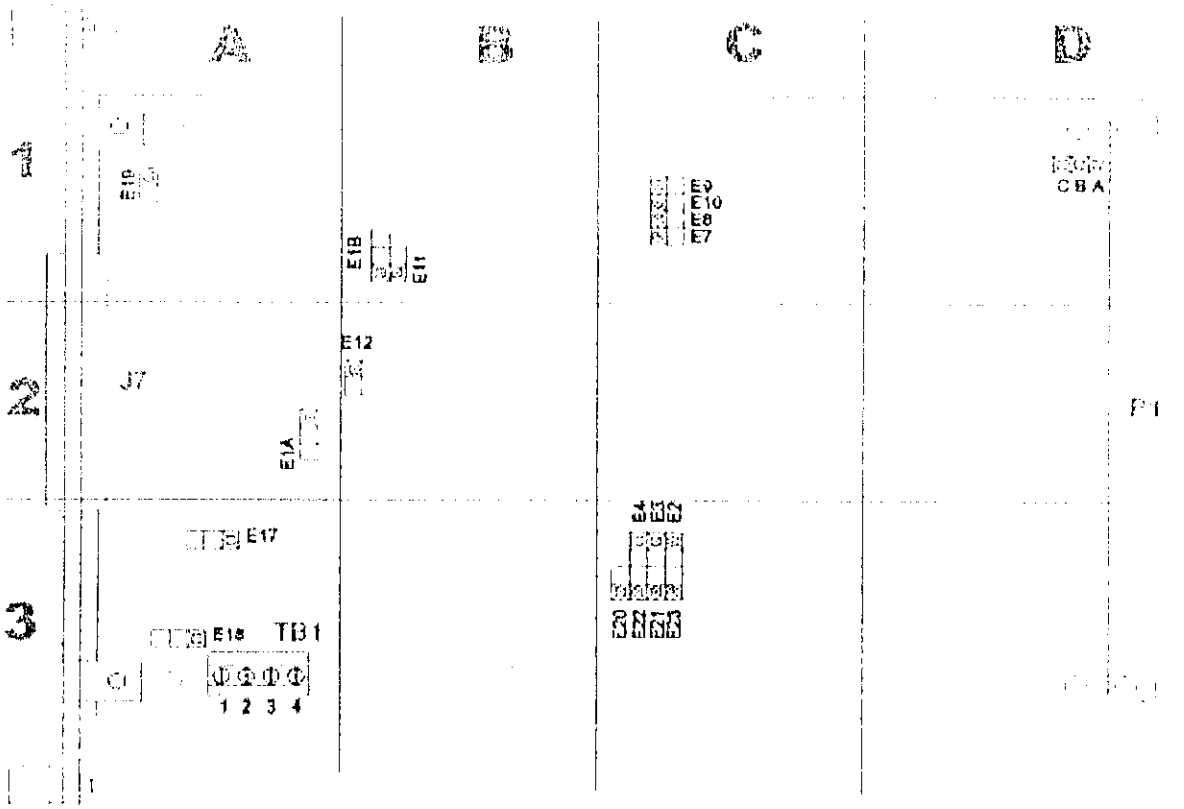



**Figure A1.1 : Répartition des Jumers sur la carte Turbo PMAC2**

| Jumper | Location | Description | Par défaut |
|---|---|---|---|
| E0 | | Utilisation de fabriquant seulement | |
| E1A | A – 2 | - Connectez les bornes 1 et 2 pour que Turbo UMAC transmet le signaux d'horloges SERVO et PHASE à l'extérieur sur le port série J7. E1B devrait relier les bornes 2 et 3.<br>- Connectez 2 et 3 pour que Turbo UMAC reçoive ses signaux d'horloge de SERVO et de PHASE sur le port série J7. E1B devrait relier les bornes 1 et 2. | 1 et 2 relier |
| E1B | B – 1 | - Connectez 1 à 2 pour obtenir l'horloge de PHASE et de SERVO du port J7 (d'une source extérieure telle qu'un autre UMAC)<br>- Connectez 2 à 3 pour obtenir l'horloge de PHASE et de SERVO du connecteur de la carte mère P1 (d'un ACC-24E2x par exemple) | 2 et 3 relier |
| E2 | | - Réservé pour de futur usage | |

| | | | |
|---|---|---|---|
| E3 | C – 3 | - Enlevez le jumper pour le mode reset normale.<br>- Connectez 1a 2 pour la ré-initialisation de tous les paramètres lors d'un reset. | Pas de connexion |
| E4 | | - Réservé pour de futur usage | |
| E7, E8, E9, E10 | C – 1 | - Utiliser pour un autre accessoire | |
| E12 | B – 2 | - Connectez 1 à 2 pour attacher la référence de masse numérique GND à celle de la référence analogue AGND<br>- Enlevez le jumper pour maintenir les tensions de référence séparée | 1 et 2 relié |
| E17 | A – 3 | - Connectez 1 à 2 pour indiquer que le port série est de type RS-232.<br>- Connectez 2 à 3 pour le port RS-422 | 1 et 2 relié |
| E18 | A – 3 | - Connectez 1 à 2 pour indiquer que le port série est de type RS-232.<br>- Connectez 2 à 3 pour le port RS-422 | 1 et 2 relié |
| E19 | A – 1 | - Connectez 1 à 2 pour déactiver le chien de garde<br>- Enlevez le jumper pour activer le chien de garde | Pas de connexion |
| E20, E21 E22 | C – 3 | Pour charger la mémoire active de flash IC lors d'un reset<br>- Enlevez le jumper E20.<br>- Connectez 1 à 2 pour E21<br>- Connectez 1 à 2 pour E22.<br>D'autres combinaisons sont pour l'usage de fabriquant seulement; la carte ne fonctionnera pas dans aucune autre configuration. | Pas de connexion pour E20<br><br>1et 2 relié pour E21, E22 |
| E23 | C – 3 | - Connectez 1 à 2 pour recharger des progiciels du port série.<br>- Enlevez le jumper pour des opérations normales. | Pas de connexion |

**Block terminal TB1 situé dans A – 3 :**

| Pin | Symbole | fonction | Note |
|---|---|---|---|
| 1 | GND | Commune | |
| 2 | +5 v | Entrée | Alimentation de tous les circuits numériques de la carte mère Turbo Pmac2. |
| 3 | +12 v | Entrée | +12 à +15 v, n'est pas nécessaire pour la carte, utiliser pour d'autres accessoires |
| 4 | - 12 v | Entrée | -12 à –15 v, utiliser pour d'autres accessoires |

Ce block terminal alimente la carte Turbo Pmac2 dans le cas où elle ne serait pas connectée à l'UBUS ; sinon ces les alimentations viennent automatiquement par le connecteur UBUS, dans ce cas-ci, ce TB ne devrait pas être employé.

# DSPGATE ACC-24E4A [11] :

**Block Terminale TB1 coté encodeur 1 :**

| Pin | Symbole | Description | Note |
|---|---|---|---|
| 1 | CHA1+ | Entrée positive du canal A de ENC1. | |
| 2 | CHA1- | Entrée négative du canal A de ENC1. | |
| 3 | CHB1+ | Entrée positive du canal B de ENC1. | |
| 4 | CHB1- | Entrée négative du canal B de ENC1. | |
| 5 | CHC1+ | Entrée positive du canal C de ENC1. | Canal d'index |
| 6 | CHC1- | Entrée négative du canal C de ENC1. | Canal d'index |
| 7 | VCC de ENC | Sortie d'alimentation digitale | Alimentation pour ENC1 |
| 8 | GND | Référence | |
| 9 | CHU1+ | Entrée indicateur supplémentaire U ou direction + pour ENC1 | En outre, entrée direction |
| 10 | CHV1+ | Entrée indicateur supplémentaire V ou direction – pour ENC1 | En outre, entrée direction |
| 11 | CHW1+ | Entrée indicateur supplémentaire W ou impulsion + pour ENC1 | En outre, entrée d'impulsion |
| 12 | CHT1+ | Entrée indicateur supplémentaire W ou impulsion – pour ENC1 | En outre, entrée d'impulsion |

CHA1-, CHB1-, CHC1- : sont utilisé pour une référence d'encodeur différentiel (la tension de référence est la différence entre CHA1+ et CHA1-), la configuration est asymétrique tel quelle, la tension de référence est la différence entre CHA1+ et GND)

**Block Terminale TB3 sortie EQU :**

| Pin | Symbole | Fonction | Description |
|---|---|---|---|
| 1 | GND | Commune | Référence du voltage |
| 2 | BEQU1 | Sortie | Sortie de comparaison 1 |
| 3 | BEQU2 | Sortie | Sortie de comparaison 2 |

Les sorties (EQU Comparer- égale ) ont une utilisation consacrée à fournir un signal quand une position d'encodeur atteint une valeur pré- définie ( certaine position ).

```
M103->X:$C003,0,24,S  ; registre de position (24-bit) a compare avec la valeur cts
                      ; ( M103 = cts )
M112->X:$C000,12,1    ; contrôle activation de sortie matériclle bit
M113->X:$C000,13,1    ; contrôle front de sortie bit
```

**Block Terminale TB1 coté Amplificateur 1 :**

| Pin | Symbole | Description | Notes |
|---|---|---|---|
| 1 | DAC1A+ | Sortie analogique de la phase A | +/-10V, par apport a AGND |
| 2 | DAC1A- | Sortie analogique de la phase A | +/-10V, par apport a AGND |
| 3 | DAC1B+ | Sortie analogique de la phase B | +/-10V, par apport a AGND |
| 4 | DAC1B- | Sortie analogique de la phase B | +/-10V, par apport a AGND |
| 5 | AE_NC_1 | Activé (Enable) l'amplificateur 1 | Normalement fermé |
| 6 | AE_COM_1 | Activé l'amplificateur 1 | |
| 7 | AE_NO_1 | Activé l'amplificateur 1 | Normalement ouvert |
| 8 | AFAULT_1+ | Entrée depuis l'amplificateur signalent un défaut d'amplification. | |
| 9 | AFAULT_1- | Entrée depuis l'amplificateur signalent un défaut d'amplification. | |
| 10 | AA+15V | (entrée ou sortie)<br>Alimentation analogique positive | Enlevez les jumpers E85, E87, E88. |
| 11 | AA-15V | (entrée ou sortie )<br>Alimentation analogique négative | Enlevez les jumpers E85, E87, E88. |
| 12 | AGND | Référence analogique | Enlevez les jumpers E85, E87, E88. |

**Block Terminale TB3 Alimentation Analogique :**

| Pin | Symbole | Description | Notes |
|---|---|---|---|
| 1 | AGND | Référence analogique | Enlevez les jumpers E85, E87, E88. |
| 2 | AA+15V | (entrée ou sortie)<br>Alimentation analogique positive | Enlevez les jumpers E85, E87, E88. |
| 3 | AA-15V | (entrée ou sortie )<br>Alimentation analogique négative | Enlevez les jumpers E85, E87, E88. |

**Connecteur TB1 coté limites interruptions ( LIMITS 1) :**

| Pin | Symbole | Description | Notes |
|---|---|---|---|
| 1 | USER1 | Entrée indicateur pour une utilisation générale | Front montant ou descendant |
| 2 | PLIM1 | Entrée indicateur de la limite positive | Front montant ou descendant |
| 3 | MLIM1 | Entrée indicateur de la limite négative | Front montant ou descendant |
| 4 | HOME1 | Entrée indicateur de la position Home | Front montant ou descendant |
| 5 | FLG_1_RET | Entrée retour de tous les indicateurs (masse de tous les indicateurs) | +(12 à 24 ) v, ou 0v |

**ACC – 24E2A canal 1 & 2 :**

| Jumper | Disposition Physique | Description | Défaut |
|---|---|---|---|
| E1A | 1 – 2 | - Aucun Jumper pour l'entrée TTL de l'indicateur CHU1<br>- Le Jumper 1-2 pour la sortie DIR1+ dans le mode pas à pas | Pas de connexion |
| E1B | 1 – 2 | - Aucun Jumper pour l'entrée TTL de l'indicateur CHV1<br>- Le Jumper 1-2 pour la sortie DIR1- dans le mode pas à pas | Pas de connexion |
| E1C | 1 – 2 | - Aucun Jumper pour l'entrée TTL de l'indicateur CHW1<br>- Le Jumper 1-2 pour la sortie PUL1+ dans le mode pas à pas | Pas de connexion |
| E1D | 1 – 2 | - Aucun Jumper pour l'entrée TTL de l'indicateur CHT1<br>- Le Jumper 1-2 pour la sortie PUL1- dans le mode pas à pas | Pas de connexion |
| E2A | 1 – 2 | - Aucun Jumper pour l'entrée TTL de l'indicateur CHU2<br>- Le Jumper 1-2 pour la sortie DIR2+ dans le mode pas à pas | Pas de connexion |
| E2B | 1 – 2 | - Aucun Jumper pour l'entrée TTL de l'indicateur CHV2<br>- Le Jumper 1-2 pour la sortie DIR2- dans le mode pas à pas | Pas de connexion |
| E2C | 1 – 2 | - Aucun Jumper pour l'entrée TTL de l'indicateur CHW2<br>- Le Jumper 1-2 pour la sortie PUL2+ dans le mode pas à pas | Pas de connexion |
| E2D | 1 – 2 | - Aucun Jumper pour l'entrée TTL de l'indicateur CHT2<br>- Le Jumper 1-2 pour la sortie PUL2- dans le mode pas à pas | Pas de connexion |
| E5 | 1-2-3 | - Connectez 1-2 pour Turbo Pmac 2<br>- D'autre configuration pour d'autre carte mère | 1 te 2 relié |
| E13 | 1 – 2 | Connectez 1-2 pour recevoir l'horloges de PHASE et de SERVO.<br>Connectez 1-2 pour transmettre l'horloges de PHASE et de SERVO. | 1 te 2 relié |
| E85 | 1 – 2 | - Connectez 1-2 pour recevoir +15v de la carte mère.<br>- Aucun jumper pour une alimentation externe +15v. | 1 te 2 relié |
| E87 | 1 – 2 | - Connectez 1-2 pour avoir AGND par carte mère.<br>- Aucun jumper pour un AGND externe. | 1 te 2 relié |
| E88 | 1 – 2 | - Connectez 1-2 pour recevoir -15v de la carte mère.<br>- Aucun jumper pour une alimentation externe -15v. | 1 te 2 relié |
| E111 | 1 – 2 | Aucun jumper pour le mode direct de l'axe 1.<br>Connectez 1-2 pour le mode impulsion et direction de l'axe1. | Pas de connexion |
| E112 | 1 – 2 | Aucun jumper pour le mode PWM de l'axe 1.<br>Connectez 1-2 pour le mode impulsion et direction de l'axe 1. | Pas de connexion |
| OPT1 | 1 – 2 | Utilisation du fabriquant | |
| OPT2 | 1 – 2 | Utilisation du fabriquant | |

**ACC – 24E2A canal 3 & 4 :**

| Jumper | Disposition Physique | Description | Défaut |
|---|---|---|---|
| E1A | 1 – 2 | - Aucun Jumper pour l'entrée TTL de l'indicateur CHU3<br>- Le Jumper 1-2 pour la sortie DIR3+ dans le mode pas à pas | Pas de connexion |
| E1B | 1 – 2 | - Aucun Jumper pour l'entrée TTL de l'indicateur CHV3<br>- Le Jumper 1-2 pour la sortie DIR3- dans le mode pas à pas | Pas de connexion |

| | | | |
|---|---|---|---|
| E1C | 1 – 2 | - Aucun Jumper pour l'entrée TTL de l'indicateur CHW3<br>- Le Jumper 1-2 pour la sortie PUL3+ dans le mode pas à pas | Pas de connexion |
| E1D | 1 – 2 | - Aucun Jumper pour l'entrée TTL de l'indicateur CHT3<br>- Le Jumper 1-2 pour la sortie PUL3- dans le mode pas à pas | Pas de connexion |
| E2A | 1 – 2 | - Aucun Jumper pour l'entrée TTL de l'indicateur CHU4<br>- Le Jumper 1-2 pour la sortie DIR4+ dans le mode pas à pas | Pas de connexion |
| E2B | 1 – 2 | - Aucun Jumper pour l'entrée TTL de l'indicateur CHV4<br>- Le Jumper 1-2 pour la sortie DIR4- dans le mode pas à pas | Pas de connexion |
| E2C | 1 – 2 | - Aucun Jumper pour l'entrée TTL de l'indicateur CHW4<br>- Le Jumper 1-2 pour la sortie PUL4+ dans le mode pas à pas | Pas de connexion |
| E2D | 1 – 2 | - Aucun Jumper pour l'entrée TTL de l'indicateur CHT4<br>- Le Jumper 1-2 pour la sortie PUL4- dans le mode pas à pas | Pas de connexion |
| E85 | 1 – 2 | - Connectez 1-2 pour recevoir +15v de la carte mère.<br>- Aucun jumper pour une alimentation externe +15v. | 1 te 2 relié |
| E87 | 1 – 2 | - Connectez 1-2 pour avoir AGND par carte mère.<br>- Aucun jumper pour un AGND externe. | 1 te 2 relié |
| E88 | 1 – 2 | - Connectez 1-2 pour recevoir -15v de la carte mère.<br>- Aucun jumper pour une alimentation externe –15v. | 1 te 2 relié |

# IOGATE ACC-12E [10] :

**Connecteur TB1 : Entrées digitales :**

| Pin | Symbole | Description |
|---|---|---|
| 1 | IN00 | Entrée 1 peut être activé au front mentent ou front descendant |
| 2 | IN01 | Entrée 2 peut être activé au front mentent ou front descendant |
| 3 | IN02 | Entrée 3 peut être activé au front mentent ou front descendant |
| 4 | IN03 | Entrée 4 peut être activé au front mentent ou front descendant |
| 5 | IN04 | Entrée 5 peut être activé au front mentent ou front descendant |
| 6 | IN05 | Entrée 6 peut être activé au front mentent ou front descendant |
| 7 | IN06 | Entrée 7 peut être activé au front mentent ou front descendant |
| 8 | IN07 | Entrée 8 peut être activé au front mentent ou front descendant |
| 9 | IN08 | Entrée 9 peut être activé au front mentent ou front descendant |
| 10 | IN09 | Entrée 10 peut être activé au front mentent ou front descendant |
| 11 | IN10 | Entrée 11 peut être activé au front mentent ou front descendant |
| 12 | IN11 | Entrée 12 peut être activé au front mentent ou front descendant |

Le connecteur TB2 a les mêmes fonctions que TB1, sauf que les numéros des entrées sont différents : IN00-IN11 pour TB1

IN12-IN23 pour TB2

**Connecteur TB3 : Références des signaux d'entrée :**

| Pin | Symbole | Description |
|---|---|---|
| 1 | REF1 | Référence du voltage pour les entrées 1-8<br>(12-24V pour front descendent / 0V pour le front montent) |
| 2 | REF2 | Référence du voltage pour les entrées 9-16<br>(12-24V pour front descendent / 0V pour le front montent) |
| 3 | REF3 | Référence du voltage pour les entrées 17-24<br>(12-24V pour front descendent / 0V pour le front montent) |

**Connecteur TB1 : Sorties de commande :**

| Pin | Symbole | Description |
|---|---|---|
| 1 | OUT00 | Sortie 1 peut être activé au front mentent ou front descendant |
| 2 | OUT01 | Sortie 2 peut être activé au front mentent ou front descendant |
| 3 | OUT02 | Sortie 3 peut être activé au front mentent ou front descendant |
| 4 | OUT03 | Sortie 4 peut être activé au front mentent ou front descendant |
| 5 | OUT04 | Sortie 5 peut être activé au front mentent ou front descendant |
| 6 | OUT05 | Sortie 6 peut être activé au front mentent ou front descendant |
| 7 | OUT06 | Sortie 7 peut être activé au front mentent ou front descendant |
| 8 | OUT07 | Sortie 8 peut être activé au front mentent ou front descendant |

| | | |
|---|---|---|
| 9 | OUT08 | Sortie 9 peut être activé au front mentent ou front descendant |
| 10 | OUT09 | Sortie 10 peut être activé au front mentent ou front descendant |
| 11 | OUT10 | Sortie 11 peut être activé au front mentent ou front descendant |
| 12 | OUT11 | Sortie 12 peut être activé au front mentent ou front descendant |

Le connecteur TB2 a les mêmes fonctions que TB1, sauf que les numéros des entrées sont différents : OUT00-OUT11 pour TB1

OUT12-OUT23 pour TB2

**Connecteur TB3 : Références des signaux d'entrée :**

| Pin | Symbole | Description |
|---|---|---|
| 1 | REF1 | Référence du voltage pour les sorties 1-8<br>(0V pour front descendent / 12-24V pour le front montent) |
| 2 | REF2 | Référence du voltage pour les sorties 9-16<br>(0V pour front descendent / 12-24V pour le front montent) |
| 3 | REF3 | Référence du voltage pour les sorties 17-24<br>(0V pour front descendent / 12-24V pour le front montent) |

Les jumpers E1-E4 sont utiliser pour la configuration de l'adresse des registres de transfert pour IOGATE.

Le jumper E5 est utiliser pour configurer l'horloge du SERVO

| Jumper | Adresse par défaut |
|---|---|
| E1 | $078C00 |
| E2 | $078D00 |
| E3 | $078E00 |
| E4 | $078F00 |
| E5 | Horloge SERVO |

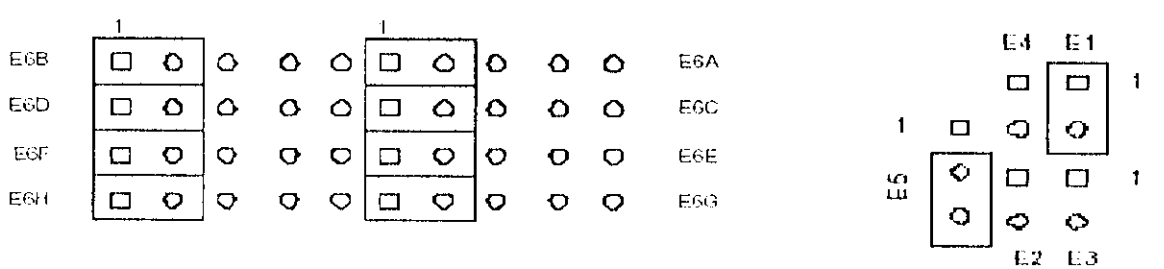



**Figure A1.2 : Connections par défaut des Jumpers**

| Jumper | Différents Cas d'installation | Description |
|---|---|---|
| E6A-E6H | 1-2 | Utiliser les bits 0-7 pour les six mémoires consécutives (48bits) |
| E6A-E6H | 2-3 ou 3-4 | Utiliser les bits 8-15 pour les six mémoires consécutives (48bits) |
| E6A-E6H | 4-5 | Utiliser les bits 16-23 pour les six mémoires consécutives (48bits) |

## Amplificateur AMP-2 [13] :

**Block Terminale TB1 : sortie de commande pour les moteurs :**

| Pin | Symbole | Description |
|---|---|---|
| 1 | AMPOUT1 | Sortie alimentation + pour le moteur 1 |
| 2 | AMPOUT1/ | Sortie alimentation - pour le moteur 1 |
| 3 | AMPOUT2 | Sortie alimentation + pour le moteur 2 |
| 4 | AMPOUT2/ | Sortie alimentation - pour le moteur 2 |
| 5 | AMPOUT3 | Sortie alimentation + pour le moteur 3 |
| 6 | AMPOUT3/ | Sortie alimentation - pour le moteur 3 |
| 7 | AMPOUT4 | Sortie alimentation + pour le moteur 4 |
| 8 | AMPOUT4/ | Sortie alimentation - pour le moteur 4 |

**Block Terminale TB2 : coté ACC – 24E2A :**

| Pin | Symbole | Fonction | Description |
|---|---|---|---|
| 1 | DAC1+ | Entrée | Signale de commande 1 |
| 2 | AENA1- | Entrée | Activé (Enable) l'amplificateur 1 |
| 3 | FAULT1- | Sortie | Défaut d'amplification de Amp 1 |
| 4 | DAC2+ | Entrée | Signale de commande 2 |
| 5 | AENA2- | Entrée | Activé l'amplificateur 2 |
| 6 | FAULT2- | Sortie | Défaut d'amplification de Amp 2 |
| 7 | AGND | | Référence analogique |
| 8 | DAC3+ | Entrée | Signale de commande 3 |
| 9 | AENA3- | Entrée | Activé l'amplificateur 3 |
| 10 | FAULT3- | Sortie | Défaut d'amplification de Amp 3 |
| 11 | DAC4+ | Entrée | Signale de commande 4 |
| 12 | AENA4- | Entrée | Activé l'amplificateur 4 |
| 13 | FAULT4- | Sortie | Défaut d'amplification de Amp 4 |

Ce connecteur apporte jusqu'à quatre signaux de commande analogiques et actifs les quatre amplificateurs, il envoie également le signal de défaut d'amplificateur au contrôleur.

**Connecteurs TB3 & TB4 : Alimentation externe :**

| Symbole | Fonction | Description |
|---|---|---|
| PGND | Entrée | Référence d'alimentation externe |
| A+40V | Entrée | Alimentation externe |

**Connecteurs TB5 ( port JFAN ) : Alimentation des circuits logiques de l'amplificateur :**

| Pin | Symbole | Fonction | Description |
|---|---|---|---|
| 1 | FAN | Sortie | Pour le ventilateur |
| 2 | FAN | Sortie | Pour le ventilateur |
| 3 | AGND | Entrée / Sortie | Référence logique |
| 4 | A+15 v | Entrée | Pour alimentation logique ( Habituellement de UBUS ) |
| 5 | AGND | Entrée / Sortie | Référence logique |
| 6 | A-15 v | Entrée | Pour alimentation logique ( Habituellement de UBUS ) |

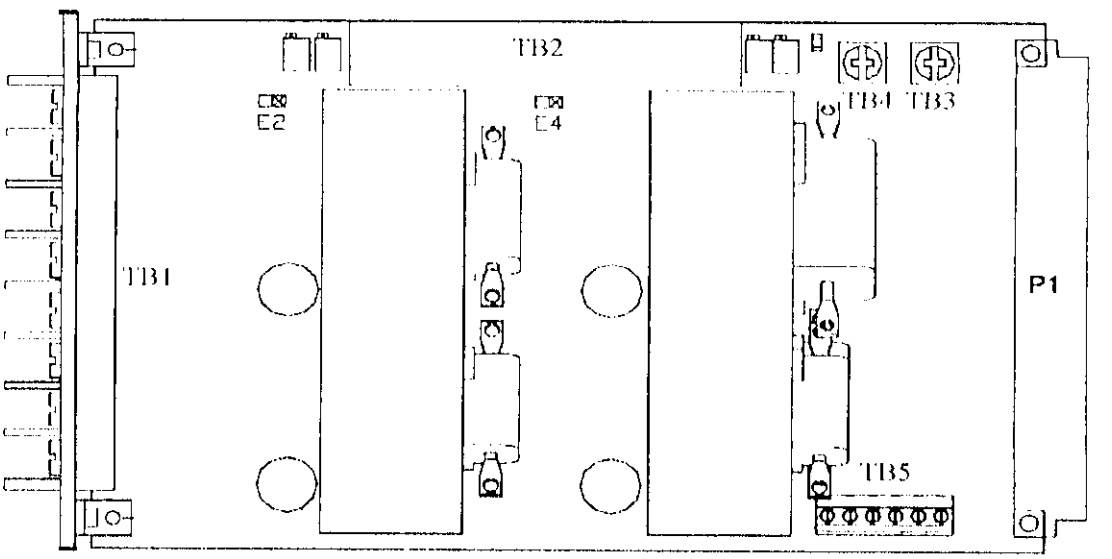



**Figure A1.3 : Répartition des Jumpers et des Connecteurs sur l'amplificateur**

**Les Jumpers d'installation de courant maximum :**

| Jumper | Description | Défaut |
|---|---|---|
| E1 | Connectez 1 à 2 pour avoir un facteur de transductance de 0.25A/V pour l'amplificateur 1<br>Pas de connexion pour avoir un facteur de 0.5A/V | Pas de connéction |
| E2 | Connectez 1 à 2 pour avoir un facteur de transductance de 0.25A/V pour l'amplificateur 2<br>Pas de connexion pour avoir un facteur de 0.5A/V | Pas de connéction |
| E3 | Connectez 1 à 2 pour avoir un facteur de transductance de 0.25A/V pour l'amplificateur 3<br>Pas de connexion pour avoir un facteur de 0.5A/V | Pas de connéction |
| E4 | Connectez 1 à 2 pour avoir un facteur de transductance de 0.25A/V pour l'amplificateur 4<br>Pas de connexion pour avoir un facteur de 0.5A/V | Pas de connéction |

# *ANNEXE 2: Table des codes d'erreurs*

# Table des codes d'erreurs [4]

| Le code de l'erreur | Problème | Solution |
|---|---|---|
| ERRO 01 | Commande non permise pendant l'exécution du programme | devrait stopper l'exécution du programme avant d'exécuter la commande |
| ERRO 02 | Erreur de mot de passe | devrait entrer le mot de passe approprié |
| ERRO 03 | Erreur de données ou commande non reconnue | devrait corriger la syntaxe de la commande |
| ERRO 04 | Caractère illégal: mauvaise valeur (> 127 ASCII) ou erreur parité sérielle | Devrait corriger le caractère et/ou vérifier le bruit sur le câble série |
| ERRO 05 | Commande non permise à moins que le buffer soit ouvert | d'abord il faut ouvrir le buffer |
| ERRO 06 | Aucun espace dans le buffer pour la commande | devrait laisser plus d'espace pour le buffer -- DELETE ou CLEAR d'autres buffer |
| ERRO 07 | buffer déjà en service | devrais CLOSE le buffer ouvert actuellement |
| ERRO 08 | erreur de lien | le registre X:$0798 tient la valeur d'erreur |
| ERRO 09 | Erreur structurale de programme (par exemple IF sans ENDIF ) | corriger la structure du programme |
| ERRO 10 | les deux limites de dépassement pour un moteur dans le système en coordination S.C sont déclenché | corriger ou neutraliser des limites |
| ERRO 11 | Mouvement précédent non accompli | Quitté le mouvement (Abort) ou lui permettre de le finir |
| ERRO 12 | Un moteur dans le S.C est dans une boucle ouverte | fermer la boucle sur le moteur (avec #xxJ/) |
| ERRO 13 | Un moteur dans le S.C n'est pas activé | placer Ixx00 à 1 ou enlever le moteur de S.C. |
| ERRO 14 | Aucuns moteurs dans le S.C | définir au moins un moteur S.C |
| ERRO 15 | Ne pas se pointé au bon programme | employer la commande B d'abord |
| ERRO 16 | Exécuter le programme incorrectement structuré (par exemple ENDWHILE manquant ) | corriger la structure du programme |
| ERRO 17 | Essai de reprendre après / ou \ avec des moteurs hors la position d'arrêtée. | employer 'J =' pour remettre des moteurs en position arrêtée |

# *ANNEXE 3: Détermination des paramètres de boucle de régulation*

## Détermination des paramètres de la boucle de régulation :[4]

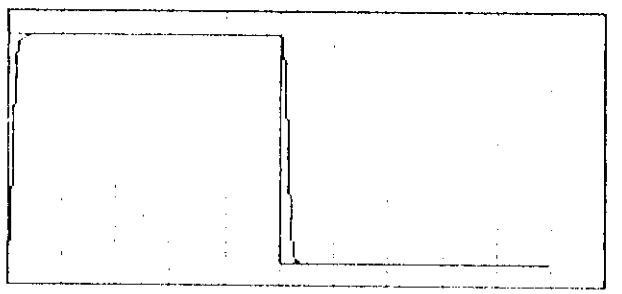
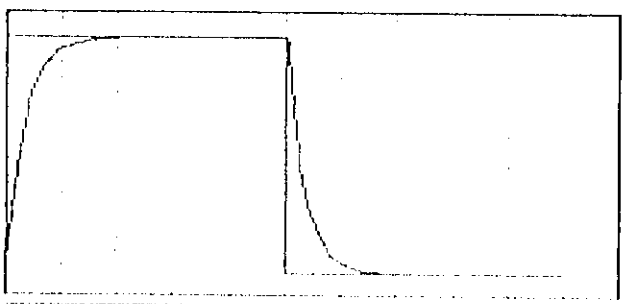
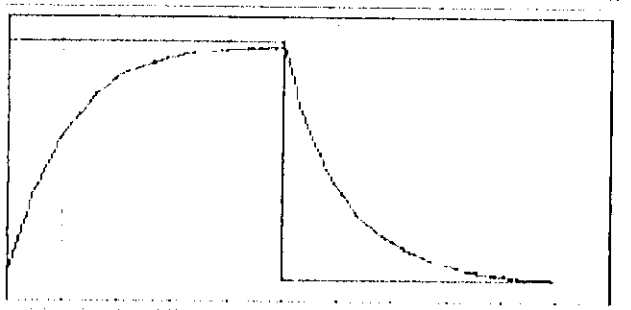
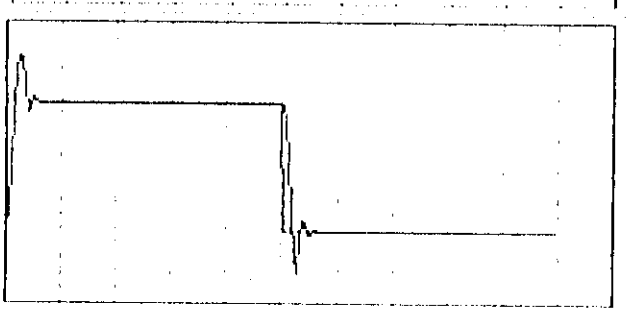
Pour la détermination des paramètres du PID et de Notch Filters, il y'a deux façons, la premier est de laisser l'utilisateur fixer les paramètres selon sont choix, la deuxième est proposer par le constructeur de l'interface, elle est composer de plusieurs étape, et sa facilité d'utilisation fait qu'on va l'utiliser dans notre application.

**1er étape :**

La première étape est d'utiliser le logiciel Pmac Tuning Pro fourni par DELTA TAU qui fait une première estimation du Gain proportionnel Kp de la boucle de régulation pour une première amélioration de la réponse a partire des tests effectués en boucle ouverte.

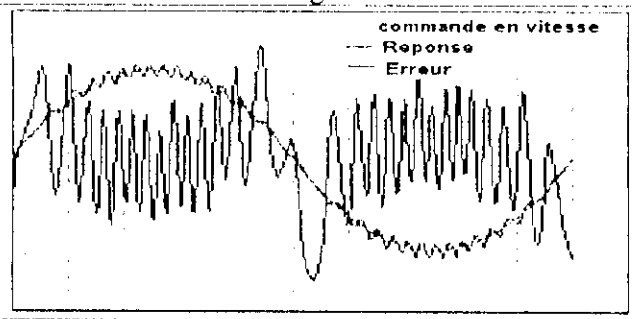
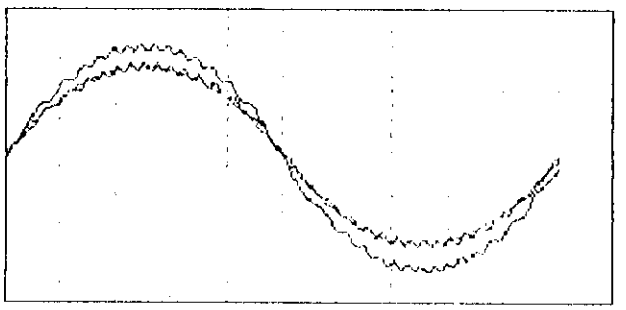
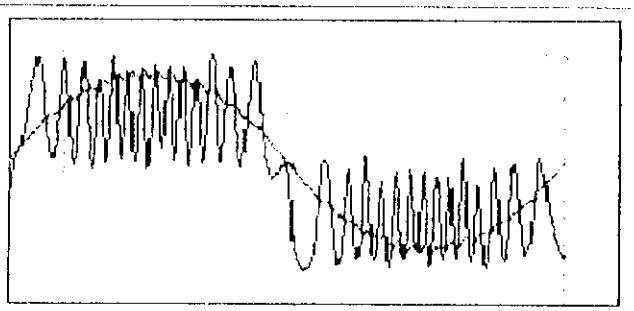
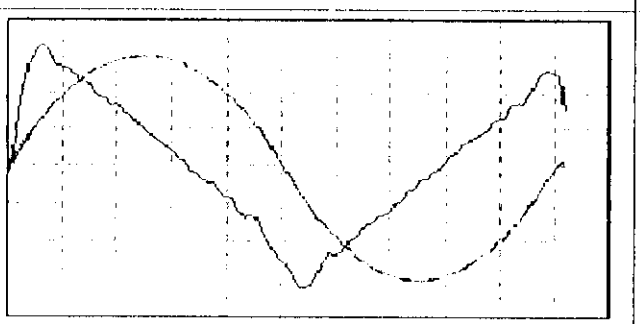
**2éme étape :**

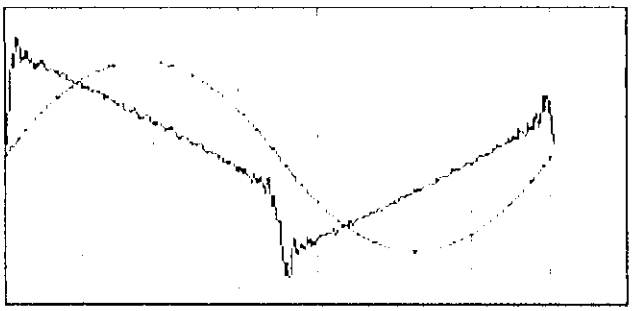
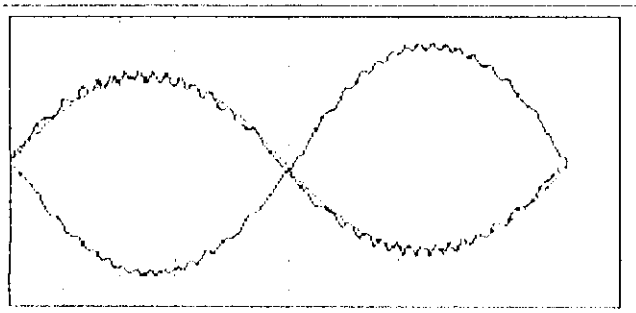
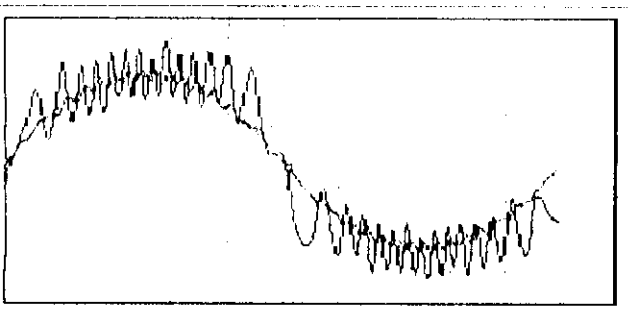
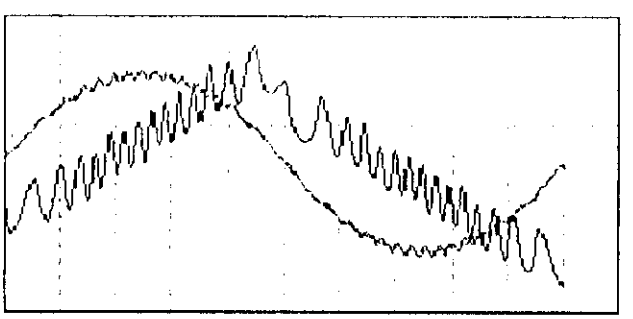
La 2éme étape est de réajustée ces paramètres pour améliorée la réponse à un échelon, cela est fait en visualisant la réponse à un échelon et d'apporter les modifications présenter dans ce tableau :

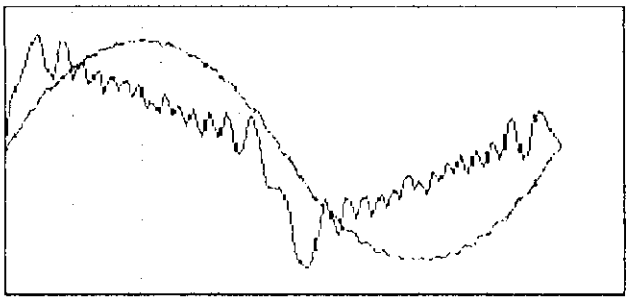
| Figure | Cause | Modification |
|---|---|---|
|  | Cas idéale | Aucune modification n'est envisagée. |
|  | Frottement ou limitation de constante de force / système | Augmentation Ki ( Ix33 ) et peut être une augmentation de Kp ( Ix30 ). |
|  | Réponse Lente due à un gain proportionnel très faible. | Augmentation du Kp (Ix30) ou diminution du Kd (Ix31). |
|  | Dépassez et oscillation due à gain proportionnel très élevé | Diminution du Kp (Ix30) ou augmentation du Kd (Ix31) |

**3éme étape :**

Une fois les paramètres Kp, Kd, Ki ajuster, la troisième étape est semblable à deuxième sauf que pour ce cas on réajuste les paramètres Kvff et Kaff pour améliorer le mouvement parabolique ( éliminer l'erreur en vitesse ), ceci est présenter le tableau suivant :

| Figure | Cause | Modification |
|---|---|---|
|  | Cas Idéal<br>L'erreur suivante est réduite au minimum et elle est concentrée au centre | Aucune modification n'est envisagée |
|  | Corrélation élevée vitesse \ erreur due à l'atténuation | Augmentation le gain feedforward en vitesse Kvff (Ix32) |
|  | Corrélation élevée vitesse \ erreur due au frottement | Augmentation du gain intégral Ki (Ix33) ou du gain feedforward de limitation (Ix68) |
|  | Corrélation élevée accélération \ erreur Retard de l'action intégrale | Augmentation du gain feedforward accélération Kaff (Ix35) |

| Figure | Cause | Modification |
|---|---|---|
|  | Corrélation élevée accélération \ erreur du limitations physiques de système | Utilisation de moins d'accélération |
|  | Corrélation vitesse \ erreur négative due à un gain de vitesse feedforward Trop élever | Diminution du gain feedforward en vitesse Kvff (1x32)) |
|  | Corrélation vitesse \ erreur élevée due à l'atténuation et les frottements | Augmentation du gain feedforward en vitesse Kvff (1x32) |
|  | Corrélation accélération \ erreur élevée due Trop d'accélération feedforward | Diminution du gain d'accélération gain Kaff ( 1xx35 ) |

| Figure | Cause | Modification |
|---|---|---|
|  | corrélation vitesse \ erreur et accélération \ erreur élevée due au Retard et à l'intégration frottement | Augmentation du gain d'accélération feedforward Kaff (Ix35) |

**4éme étape:**

Si le système ne réagit pas aux bruit et il n'y a pas de résonnasse la quatrième étape n'est pas nécessaire, par contre si ce n'est pas le cas alors il suffie simplement d'utiliser le logiciel Pmac'Tuning Pro et de donner les paramètres nécessaires comme la fréquence de coupure et la fréquence qu'on veut rejeter et le logiciel calcule automatiquement les paramètres de ce filtre. Pour notre cas, on n'a pas constaté une résonance ou une réaction au bruit, et par conséquent ce filtre ne sera par utiliser.

# *ANNEXE 4: Câblage de l'UMAC*

## 1 Connections des Blocks Terminales de ACC-24E2A ( DSPGATE ) : : [10]

### 1.1 Coté Encodeur :

Le moteur dont nous disposons contient un encodeur de haute résolution, qui envois a la DSPGATE le code de la position sur 3 lignes, les deux lignes restantes sont pour son alimentation. Le branchement de l'encodeur avec l'accessoire ACC-24E2A est illustrer dans la figure suivante :

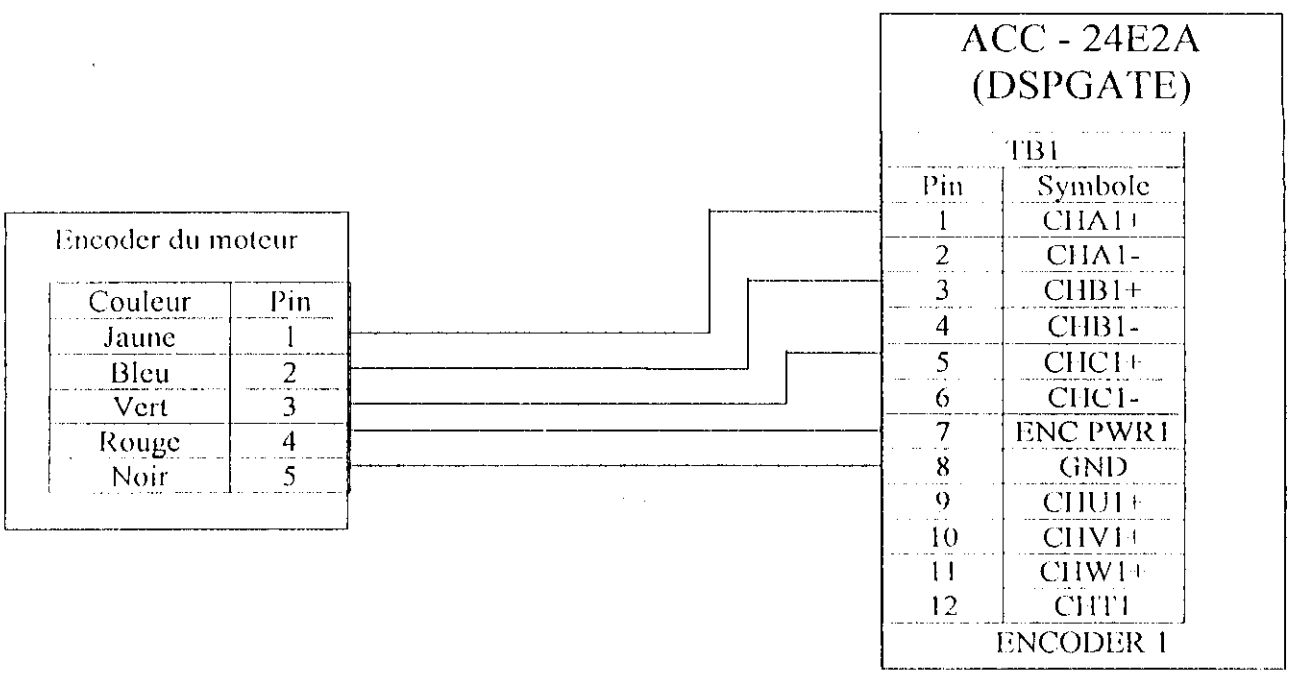



**Figure A.C.1 : Diagramme de câblage ACC24E2A avec Encodeur**

### 1.2 Coté Amplificateur : [13]

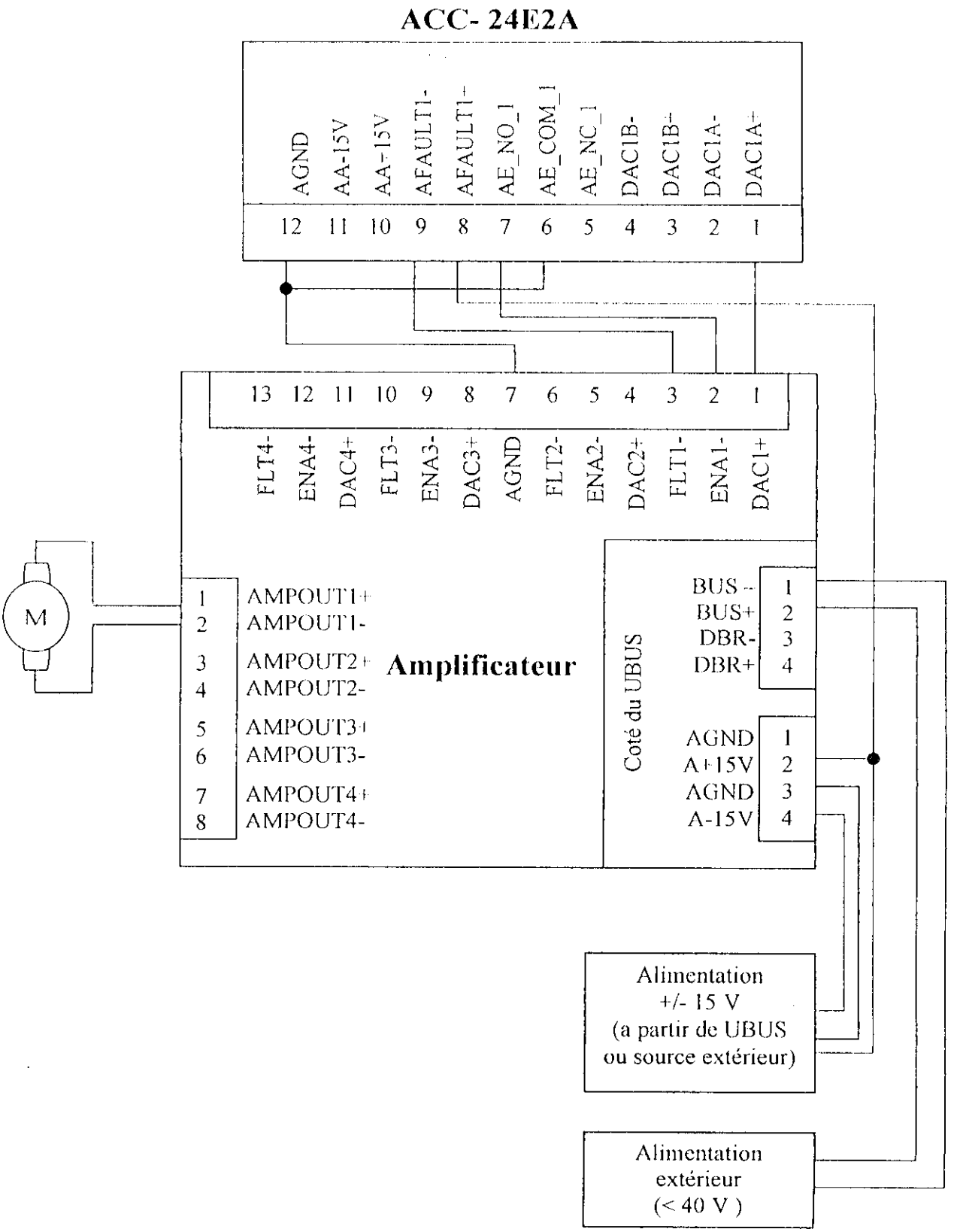
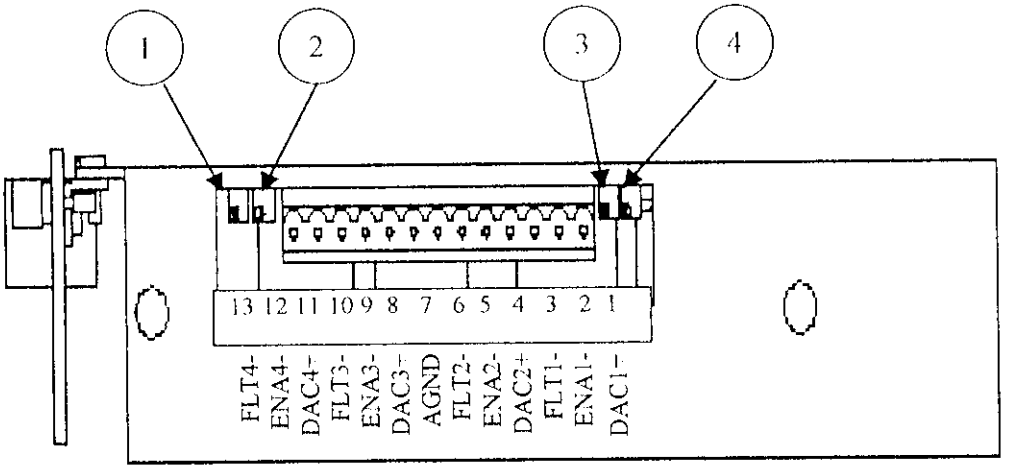



**Figure IV.2. Diagramme de câblage ACC24E2A avec Amplificateur**

## 2 Présentation des régulateurs manuel d'offset pour les amplificateurs : [13]




1 : accès a un réglage d'offset manuel pour le premier moteur #1.

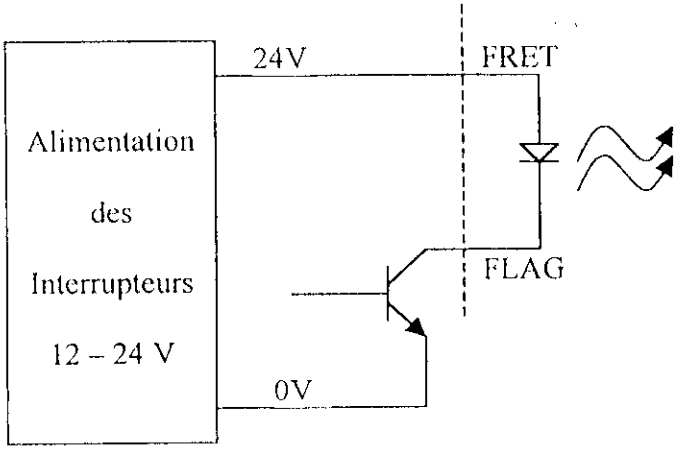
2 : réglage d'offset du moteur #2.

3 : réglage d'offset du moteur #3.

4 : réglage d'offset du moteur #4.

## 3 Branchement des entrée fin de course ( les butés des articulations ) : [10]

### 3.1 Pour un front descendant ( Sinking ) :

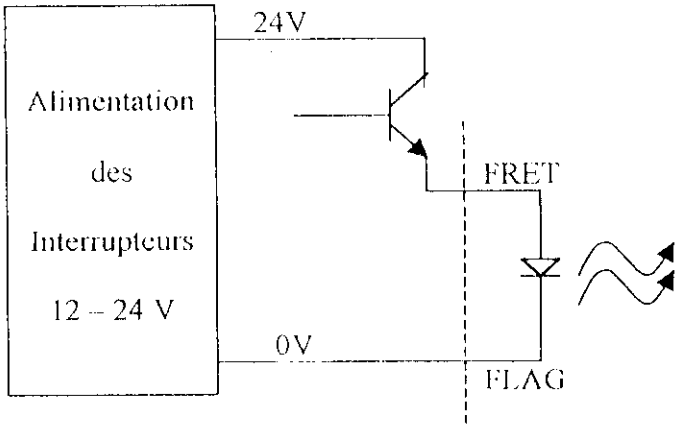



**ACC – 24E2A**

| PIN | Symbole |
|---|---|
| 1 | USER |
| 2 | PLIM |
| 3 | MLIM |
| 4 | HOME |
| 5 | FRET |

**Entrées interruptions**

### 3.2 Pour un front montant ( Sourcing ) :




**ACC – 24E2A**

| PIN | Symbole |
|---|---|
| 1 | USER |
| 2 | PLIM |
| 3 | MLIM |
| 4 | HOME |
| 5 | FRET |

**Entrées interruptions**

# *REFERENCES BIBLIOGRAPHIQUES*

[1] **DELTA TAU DATA SYSTEMS, INC. Delta Tau FTP Site,** http://www.deltatau.com.

[2] **DELTA TAU DATA SYSTEMS, INC. UMAC Products Guide version 2.1, Disponible en format pdf de** ftp://ftp.deltatau.com/NewIdeasInMotion/umacpg.pdf, **Janvier 2001.**

[3] **DELTA TAU DATA SYSTEMS, INC. PMAC2 and Turbo PMAC2 Products Guide, Disponible en format pdf de**ftp://ftp.deltatau.com/NewIdeasInMotion/p2t2pglr.exe, **Septembre 2000.**

[4] **DELTA TAU DATA SYSTEMS, INC. PMAC(1) Family Quick Reference Manual version 2.1 Disponible en format pdf de** ftp://ftp.deltatau.com/NewIdeasInMotion/pmacqref.pdf, **Juin 2000.**

[5] **DELTA TAU DATA SYSTEMS, INC. PMAC1 USER'S MANUAL, Disponible en format pdf de** ftp://ftp.deltatau.com/NewIdeasInMotion/pmac1usr.pdf, **Janvier 2000.**

[6] **DELTA TAU DATA SYSTEMS, INC. PMAC2 USER'S MANUAL, Disponible en format pdf de** ftp://ftp.deltatau.com/NewIdeasInMotion/pmac2usr.pdf, **Janvier2000.**

[7] **DELTA TAU DATA SYSTEMS, INC. Turbo PMAC software manual (version 1.936) , Disponible en format pdf de** ftp://ftp.deltatau.com/PMACManual/t1t2sw.exe, **Septembre 2001.**

[8] **DELTA TAU DATA SYSTEMS, INC. Turbo PMAC version 1.936 software addendum manual, Disponible en format pdf de** ftp://ftp.deltatau.com/NewIdeasInMotion/t1936add.pdf, **Janvier 2000.**

[9] **DELTA TAU DATA SYSTEMS, INC. UMAC Turbo (3U Turbo PMAC2) Hardware Reference Manual, Disponible en format pdf de** ftp://ftp.deltatau.com/UMAC/umactbhw.pdf, **Août 2001.**

[10] **DELTA TAU DATA SYSTEMS, INC. UMAC ACC-24E2A, the analog +/- 10 Volts axes board, Disponible en format pdf de** ftp://ftp.deltatau.com/UMAC/acc24e2a.pdf, **Octobre 2001.**

[11] **DELTA TAU DATA SYSTEMS, INC. UMAC I/O Accessory, HIGH POWER OPTO 24 INPUT/24 OUTPUT BOARD, Disponible en format pdf de** ftp://ftp.deltatau.com/UMAC/acc12e.pdf, **Décembre 2000.**

[12] **DELTA TAU DATA SYSTEMS, INC. UMAC UBUS Preliminary Specification, Disponible en format pdf de** ftp://ftp.deltatau.com/UMAC/ubus.pdf, **Avril 2000.**

[13] **DELTA TAU DATA SYSTEMS, INC. AMP-2, four axes analog PWM amplifier, 48 VDC 3/5 A, Disponible en format pdf de** ftp://ftp.deltatau.com/UMAC/amp-2.pdf, **Avril 2001.**

[14] **DELTA TAU DATA SYSTEMS, INC. UMAC Power supply: 20 A @ 5 V, 1 A @ +/- 15 V , Disponible en format pdf de** ftp://ftp.deltatau.com/UMAC/acce2.pdf, **Février 2002.**

[15] **MOTOROLA, SEMICONDUCTOR PRODUCT INFORMATION. DSP56303 24-Bit General-Purpose Digital Signal Processor Data Sheet, Disponible en format pdf de** http://e-www.motorola.com/brdata/PDFDB/docs/DSP56303DS.pdf, **Janvier 2002.**

[16] **MOTOROLA, SEMICONDUCTOR PRODUCT INFORMATION. DSP56303 24-Bit Digital Signal Processor User's Manual, Disponible en format pdf de** http://e-www.motorola.com/brdata/PDFDB/docs/DSP56303UM.pdf, **Janvier 2001.**

[17] **MOTOROLA, SEMICONDUCTOR PRODUCT INFORMATION. Advance Information. 4-BIT DIGITAL SIGNAL PROCESSOR, Disponible en format pdf de** http://e-www.motorola.com/brdata/PDFDB/docs/DSP56303P.pdf, **Juillet 2001.**

[18] **T.Redheendran. Booting DSP563xx Devices Through the Serial Communication Interface (SCI), Disponible en format pdf de** http://e-www.motorola.com/brdata/PDFDB/docs/AN1781.pdf, **Novembre 98.**

[19] **Barbara Johnson.Software Differences Between the DSP56002 and the DSP56303, Disponible en format pdf de** http://e-www.motorola.com/brdata/PDFDB/docs/AN1829.pdf, **Octobre 2001.**

[20] **Barbara Johnson.DSP56303 Hardware Differences Between the DSP56002 and the DSP56303, Disponible en format pdf de** http://e-www.motorola.com/brdata/PDFDB/docs/AN1830.pdf, **Octobre 2001.**

[21] **National Semiconductor. LMD 18200 H-Bridge Data Sheet, Disponible en format pdf de** http://www.nsc.com/ LMD18200.pdf, **Décembre 1999.**