REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

PROJET DE FIN D'ETUDES

présenté à

L'ECOLE NATIONALE POLYTECHNIQUE

pour l'obtention du

DIPLOME D'INGENIEUR D'ETAT EN AUTOMATIQUE

Intitulé du sujet

COMMANDE DU ROBOT AID PAR PC UTILISATION DE LA CARTE DSP FLEX MOTION

DIRIGE PAR :
M. BOUCHERIT

ETUDIE PAR :
REMICHI Fatima

Année universitaire 1997/1998

ECOLE NATIONALE POLYTECHNIQUE

DER GENIE ELECTRIQUE ET INFORMATIQUE

SPECIALITE : AUTOMATIQUE

PROJET DE FIN D'ETUDES

INTITULE

« COMMANDE DU ROBOT AID PAR PC »
« UTILISATION DE LA CARTE DSP FLEX MOTION »

DIRIGE PAR :
M. BOUCHERIT

ETUDIE PAR :
REMCHI FATIMA

1997/1998

A la mémoire de ma grand mère

A ma tendre mère

A mon adorable père

A mon frère Nabil

Et à l'amitié de tous ceux qui me l'ont témoigné durant ces cinq années formidables.

AVANT - PROPOS

Avant d'entreprendre l'exposé de ce travail, je souhaite remercier monsieur *BOUCHERIT* professeur à l'Ecole Nationale Polytechnique de m'avoir permis de travailler sur ce sujet.

Je tiens à souligner la part importante prise par monsieur *MENZOU* à la réalisation de ce mémoire, ses conseils et son expérience m'ont aidé à développer le thème de recherche sur de très bonnes bases, qu'il soit assuré de toute ma reconnaissance.

Je remercie également monsieur *REZINE* chef de l'UER Automatique, et monsieur *BOUSLIMANI* professeur à l'Ecole Militaire Polytechnique, de m'avoir accueilli au Laboratoire de Robotique et de Productique de l'EMP.

J'adresse également mes remerciements aux membres du jury qui ont accepté de juger ce travail.

Enfin je ne pourrait oublier monsieur *REMICHI*, mon cher père pour son aide si précieuse à la réalisation matérielle de ce mémoire.

RESUME :

Ce mémoire traite essentiellement de l'exploitation optimale d'une carte DSP appelée Flex Motion, pour la commande du robot AID-V5-EN.

La *programmation niveau effecteur* est l'objectif principal de ce travail, une application dans ce sens est présentée à la fin du mémoire pour illustrer les capacités certaines du travail en tandem de la carte DSP Flex Motion et du robot AID.

Les mots clés sont :

DSP Flex Motion, robot AID, programmation, commande, ressource, tache, effecteur.

Abstract :

This memoir deals essentially with the optimal exploitation of an electronic DSP card called FLEX MOTION, for the control of the robot AID-V5- EN.

The *effector level programming* is the principal objective of this work, an application in this sense is presented at the end of the memoir to illustrate capacities of the work in tandem of the card DSP FLEX MOTION and the robot AID.

The key words are:

DSP Flex Motion, AID robot, programming, control, resource, task, effector.

SOMMAIRE

« Notre ennemi dans l'étude c'est la suffisance, quiconque veut réellement apprendre doit commencer par s'en débarrasser. S'instruire sans jamais s'estimer satisfait et enseigner sans jamais se lasser, telle doit être notre attitude »

Mao Tse Toung.

INTRODUCTION

INTRODUCTION :

Le travail présenté dans ce mémoire a été réalisé au Laboratoire de Robotique et de Productique de l'Ecole Militaire Polytechnique (EMP).

Le problème abordé est celui de la *commande du robot AID-V5-EN via la DSP Flex Motion.*

La robotique née dans le courant des années soixante dix est devenue une science pluridisciplinaire par excellence. Elle met en oeuvre des disciplines multiples toutes à la pointe du progrès technologique : électronique, informatique, automatique, instrumentation électrotechnique, **CAO** (**C**onception **A**ssistée par **O**rdinateur), **CFAO** (**C**onception et **F**abrication **A**ssistée par **O**rdinateur), traitement d'images (7), etc...

La robotique se fixe donc l'objectif de construire des machines articulées dotées de la flexibilité opérationnelle de l'homme avec un pouvoir de décision plus ou moins étendu associés aux avantages que procure la mécanisation : la robustesse, la précision, la rapidité, la fiabilité et enfin l'insensibilité à la nature de l'environnement.

Le développement en matière de robotique puise sa dynamique dans le souci constant de l'homme de produire toujours plus de biens de consommation à moindre coûts, dépasser ses limites physiques et intellectuelles pour repousser les frontières de ce qui lui est inconnu et se libérer des taches ingrates, fastidieuses et dangereuses pour s'investir dans des missions plus valorisantes (21).

Dans cette optique, l'analyse des champs d'application des robots (explorations sous marine, la conquête de l'espace, travail en milieu nucléaire, fonderie, assistance médicale..) leur rôle vis à vis de l'homme d'une part et de l'univers d'autre part, permet d'aboutir à leur classification en deux familles (20):

- **Robots de substitution** : ils sont destinés à remplacer partiellement ou totalement l'homme dans l'exécution de certaines taches.
- **Robots de coopération** : ils sont destinés à travailler en collaboration étroite avec un opérateur humain.

Le développement spectaculaire des robots a été rendu possible grâce aux développements qu'a connu parallèlement l'électronique, l'informatique et l'automatique qui permettent de commander des structures mécaniques complexes (chaînes ouvertes, fermées ou parallèles) à plusieurs degrés de liberté (ddl).

Une présentation succincte du robot AID-V5-EN est abordée au chapitre I, les différents composants des parties mécanique et électrique sont décrits et cela pour permettre au lecteur d'avoir une idée assez complète du robot considéré, car c'est un élément indissociable de notre travail.

Nous abordons ensuite, dans ce même chapitre, la présentation matérielle et logicielle de la carte **DSP** (digital signal processing) **Flex Motion**, une présentation

très détaillée des composants de la DSP elle même et du bornier **UMI-6 Flex** y sont présentés, nous introduirons par la suite le nouveau dispositif de commande du robot.

Nous verrons que la carte DSP présente des possibilités logicielles très intéressantes, les propriétés jugées nécessaires pour notre travail seront également détaillées.

Le but du chapitre II est de décrire du point de vue mécanique les différents modèles que nous pouvons construire pour rendre compte du comportement complexe des robots.

Le modèle complet d'un robot est décomposé en modèles élémentaires, relatifs à des aspects partiels du comportement global.

Les principaux modèles présentés sont : le modèle géométrique (direct et inverse), le modèle cinématique (direct et inverse), le modèle dynamique qui est introduit grâce à deux formulations célèbres, la formulation de NEWTON EULER et la formulation LAGRANGIENNE, et enfin le modèle de la commande. Chacun de ces modèles est brièvement décrit, certains problèmes posés à leur niveau sont résolus d'autres ne le sont pas, y sont cités.

Le chapitre I et II ne sont en fait qu'une base au travail effectué au chapitre III, dans lequel nous présentons notre contribution au sujet de la commande du robot AID-V5-EN.

Le chapitre III traite l'aspect de la programmation des robots industriels. Nous avons jugé utile d'introduire quelques notions sur les systèmes de programmation pour la robotique, afin d'aboutir à notre contribution à la programmation du robot au niveau effecteur, la tache et l'organisation de sa programmation sur PC est présentée sous forme de dessins et d'organigrammes.

« CHAPITRE I »

Présentation du robot AID et sa connexion avec le PC

I.1. GENERALITES SUR LES ROBOTS MANIPULATEURS :

I.1.1 INTRODUCTION :

Les récents développements de la robotique, tant sur le plan industriel que médical, laissent présager un brillant avenir à cette discipline. Le robot est avant tout un de ces nouveaux produits d'automatisation qui caractérisent cette dernière décennie. Le robot est formé de micro électronique et informatique mais également de mécanismes, de moteurs et de capteurs (3).

Les caractéristiques principales des robots sont (10):

- ***L'automaticité*** : est la possibilité pour le système d'accomplir une tache plus ou moins complexe, sans avoir recours à un opérateur humain.
- ***L'adaptativité*** : qui permet au système d'exécuter une tache précise dans un environnement variable, partiellement ou totalement inconnu.
- ***La polyvalence*** : permet au système d'atteindre des objectifs différents, tout en conservant la même structure. Pour un robot, ce changement d'objectif se traduit par la modification ou le remplacement du programme qui commande l'évolution du robot.

I.1.2. RETROSPECTIVE :

Ancêtres des robots, les premiers manipulateurs ont été développés et utilisés par les industries aérospatiales et nucléaires pour intervenir en milieu hostile. Ils datent d'une quarantaine d'années et sont en général du type « maître esclave » mécanique, commandés directement par un opérateur humain : leur rôle consiste tout simplement à transmettre le mouvement au bras esclave (10) (2).

Depuis lors, des progrès ont été accomplis pour arriver au concept actuel de robot.

En effet, il a fallu attendre jusqu'à 1962 pour que le premier robot industriel Unimate mis au point par deux ingénieurs américains ENGELBERGER et DEVOL voit le jour dans une usine de fabrication de voitures FORD. Un seul processeur pilotait simultanément les six axes.

Depuis ce jour, l'évolution de le robotique a suivi celle de l'automatique, l'informatique et l'électronique.

En 1977, les premières études sur les petits robots d'assemblage sont lancées, elles ont abouti aux premiers robots PUMA 250 et 500 commercialisés en 1980, six microprocesseurs assurent le pilotages des six axes du robot, et un septième assure la supervision de l'ensemble (21).

I.1.3. DEFINITION DES ROBOTS :

I.1.3.1. DEFINITION SELON LA NORME AFNOR (21):

L'Association Française de Normalisation définit le robot comme suit:
« Le robot industriel est un manipulateur commandé en position, reprogrammable, polyvalent, à plusieurs degrés de liberté (ddl), capable de manipuler des pièces, des outils au cours de mouvements variables pour l'exécution d'une variété de taches. Il prend souvent l'apparence d'un ou plusieurs bras se terminant par un poignet. Son unité de commande utilise notamment un dispositif de mémoire et éventuellement de perception et d'adaptation à l'environnement et aux circonstances. Ces machines polyvalentes sont généralement adaptées pour faire la même fonction cyclique ».

I.1.3.2. DEFINITION SELON LA NORME ISO (16):

L'Organisation Internationale de Normalisation, n'a pas définit le mot *robot* considérant qu'il s'agissait d'un mot du langage courant, mais *robot manipulateur industriel* comme suit :
« manipulateur à plusieurs degrés de liberté, à commande automatique, reprogrammable, multi-applications, mobile ou non, destiné à être utilisé dans les applications d'automatisation industrielle ».

I.1.4. FONCTIONS DE BASE DE LA ROBOTIQUE :

Les robots sont généralement destinés à effectuer trois types de tache :

I.1.4.1. TACHES DE MANUTENTION ET DE STOCKAGE (10):

Ce sont les premières et les plus importantes applications. La manutention et le stockage des pièces sont des opérations coûteuses en fabrication et en construction mécaniques. La manutention intervient dès la réception du matériel et des matières premières, elle est présente à chaque stade de la fabrication, elle assure enfin l'acheminement des produits finis jusqu'à leur livraison.

I.1.4.2. TACHE DE FABRICATION (10) (12):

Le robot est intégré à une chaîne de fabrication. Dans ce cas, les robots sont des supports d'outils et participent directement à l'élaboration des produits, ceci peut être :

- Un soudage : par point, à l'arc ou par laser.
- Une découpe : par fraise , par jet d'eau ou par laser.
- Une dépose d'adhésifs : colle ou mastic d'étanchéité.
- Un polissage : par bandes abrasives ou par brosses à lustrer.
- Une pulvérisation- métallisation.
- Etc.

Dans la plupart de ces applications, le robot doit effectuer une trajectoire dans l'espace avec une grande précision.

I.1.4.3. TACHE D'ASSEMBLAGE (10) (12):

Le principe de l'assemblage est de réaliser à partir de plusieurs pièces un ensemble complet et fonctionnel, ici le robot effectue les deux taches précédentes en même temps.

Cette tache nécessite plus particulièrement une grande précision de positionnement de l'effecteur. L'analyse des principales fonctions de la robotique met en relief le rôle essentiel du robot : celui d'exercer une action physique sur son environnement en divers points de l'espace.

Le sujet traité au cours de ce mémoire est un sujet expérimental avant tout, il est donc nécessaire de consacrer ce premier chapitre pour faire la présentation complète du dispositif étudié.

Le robot AID est présenté en premier lieu, puisque la commande lui est destinée, la carte DSP Flex Motion utilisée pour générer cette commande est présentée en deuxième partie de ce chapitre.

I.2. PRESENTATION DU ROBOT AID (1):

L'AID V5- EN (**A**ssistance **I**ndustrielle **D**auphinoise **V**ertical à **5** degrés de liberté **E**lectrique **N**umérique), ce robot est conçu par la firme « Assistance Industrielle Dauphinoise ».

Comme l'indique son nom ce robot est de type vertical à articulations rotoïdes et concourantes, il offre cinq degrés de liberté en version standard et six en option (axe 4), il se compose d'actionneurs électriques (moteurs à courant continu) qui se trouvent au niveau des liaisons et d'une commande numérique (voir figure I.1), ce robot a un champ d'action très important il est très proche des robots industriels.

I.2.1. PARTIE MECANIQUE :

I.2.1.1. GEOMETRIE DE L'AID :

(a) Rayon d'action :

- Le volume atteignable par l'extrémité de l'organe terminal est de l'ordre de $2m^3$.
- La surface maximale atteignable :
  - Dans un plan vertical : 2 m^2.
  - Dans un plan horizontal : 2,6 m^2.
- Rayon d'action du robot : 900 à 950 mm.
- Altitude maximale par rapport à la base du socle : 1320 mm.
- Altitude minimale : -200 mm.

(b) Débattement angulaire :

Numéro de l'axe	Débattement
1	270°
2	180°
3	270°
4	270°
5	180°
6	270°

Tableau I.1

I.2.1.2. CONCEPTION MECANIQUE :

Le socle et les segments sont réalisés en fonderie d'alliage léger, sauf le segment 3 qui est constitué d'un tube d'alliage léger. Les articulations des différents segments utilisent un module offrant sous un encombrement réduit la fonction articulation et la fonction réduction proprement dite.

(a) La fonction articulation :

Elle remplie par une butée à aiguilles et un roulement à billes, ce qui permet d'avoir un bon guidage sous un encombrement faible.
La butée à aiguilles assujettit le bras à se déplacer dans un plan.

Le roulement à bille détermine l'axe géométrique de rotation, de plus une rondelle élastique crée une précontrainte pour annuler les jeux.

(b) La fonction réduction :

Elle est assurée par un engrenage :

Pour les axes 1, 2 et 3 :

- La réduction est de 100.
- La liaison entre le moteur et l'arbre primaire du réducteur AID se fait par un système engrenage courroie crantée.
- L'écrou sert à régler les jeux de réducteur.
- Le codeur, les mini rupteurs, un top index et la came réglable qui les actionne sont intégrés au réducteur.

Pour les axes 4, 5 et 6 :

- La réduction est de 125.
- La liaison entre le moteur et l'arbre primaire du réducteur AID est assurée par un engrenage dont le rapport de réduction est de 50/51 pour les trois axes.
- L'écrou permet de régler le jeu de réducteur.
- Les mini rupteurs, un top d'index et la came réglable qui les actionne sont intégrés au réducteur.

(c) Pince :

Elle est standard de type bidigital à mors parallèles à commande électrique, sa course maximale ne dépasse pas 40 mm et la vitesse maximale d'ouverture ou de fermeture est de 40 mm/s, elle peut exercer une force de serrage de l'ordre de 30 N.

(d) Codeurs :

L'AID V5-EN est équipé de codeurs optiques de type incrémental, ils sont utilisés dans les asservissements de position, ils possèdent deux sorties principales chacune générant un certain nombre d'impulsions par tour qui détermine la résolution du capteur.

Les deux signaux de sortie sont déphasés d'un quart de pas en décalant les deux pistes de sorte que l'examen de la différence de phase entre ces deux signaux permet de déterminer le sens de rotation de l'arbre moteur. Un troisième signal existe et produit deux impulsions par tour (22).

Sur les axes 1, 2 et 3 il s'agit de codeurs MCB type GIK 40, 250 points à deux pistes en quadrature, ils sont montés sur les arbres primaires des réducteurs.

Sur les axes 4, 5 et 6 il s'agit de codeurs PORTESCAP type BY , 144 points par piste, ils sont intégrés aux moteurs.

(e) **Motricité :**

Les moteurs sont de type courant continu, ils sont caractérisés par le fait que le champ inducteur occupe une direction fixe, ce champ est créé par un aimant permanent.

Les moteurs utilisés sont donnés dans le tableau (I.2).

Numéros des axes	Nom du moteur	Type du moteur
1, 2, 3	MAXON	2260-815
4, 5	PORTESCAP	D21 216 E
6	PORTESCAP	L21 216 E
Pince	PORTESCAP	L21 213 E

Tableau I.2

(f) Performances:

a) Charge utile est de 1kg.
b) Masse totale est de 31 kg dont la masse fixe qui est égale à 8 kg.
c) Vitesse maximale approximative pour:
- Axes 1 : 1,2 rd/sec jusqu'à 1700 tr/mn.
- Axes 2 et 3 : 1 rd/sec jusqu'à 2600 tr/mn.
- Axes 4, 5 et 6 : 0,8 rd/sec jusqu'à 3300 tr/mn.

d) Efforts pouvant etre exercés par l'organe terminal sur un objet exterieur:
- Couple autour de l'axe de la pince: 3 mN.
- Force dans l'axe de la pince: 20N.
- Force normale au plan de la pince: 20N.

I.2.2. PARTIE ELECTRIQUE :

Le robot AID est constitué d'une partie commande numérique et d'une autre amplificateur de puissance.

I.2.2.1. PARTIE COMMANDE NUMERIQUE :

La commande numérique est de type CNC 7600R, elle est basée sur des ensembles électroniques modulaires enfichables et interchangeables, elle est équipée de:

a) Un ecran de visudisation CRT 9.
b) Un clavier de 37 touches.
c) Deux clés de verouillage.
d) Une cassette mémoire enfichable.
e) Un boitier de télécommande.
f) Une interface avec le lecteur de cassette audio.
g) Neuf cartes électroniques suivantes:

- ♣ Une carte unité centrale NPU: elle est constituée d'un microprocesseur central Z80-A, un processeur arithmétique AMD 9511 et un contrôleur d'interruption 8259.
- ♣ Une carte de contrôle de l'ecran.
- ♣ Une carte mémoire.
- ♣ Une carte entrées logiques.
- ♣ Une carte sorties logiques.
- ♣ Trois cartes d'axes: chacune gère deux axes et elle est constituée d'un microprocesseur Z80-A, une mémoire EPROM de contrôle des axes, une mémoire RAM, filtre et trigger d'entrée des impulsions codeurs, deux compteurs de position et deux convertisseurs digitaux analogiques pour commander les amplificateurs d'axes.
- ♣ Une carte d'interface constituée d'une interface cassette mémoire enfichable, une interface RS-232c, et des circuits adaptateurs de bus.

I.2.2.2. PARTIE PUISSANCE :

La partie puissance est constituée de:

a) Six cartes amplificateurs de puissance.
b) Deux condensateurs de filtrage des alimentations.
c) Transformateur d'alimentation générale.
d) Ventilateurs des amplificateurs de puissance.
e) Pont de diodes.
f) Interface pince.
g) Carte relais moteurs et pince.

Les cartes amplificateurs de puissance sont réparties en deux groupes :

- ♣ Pour les cartes des axes 1, 2 et 3 le courant nominal permanent est de 1,35 A, le courant crête est de 5,4 A et la tension de commande est égale à 30 V.

- Pour les cartes des axes 4, 5 et 6 le courant nominal permanent est de 0,45 A, le courant crête est de 1,8 A et la tension de commande est égale à 30 V.

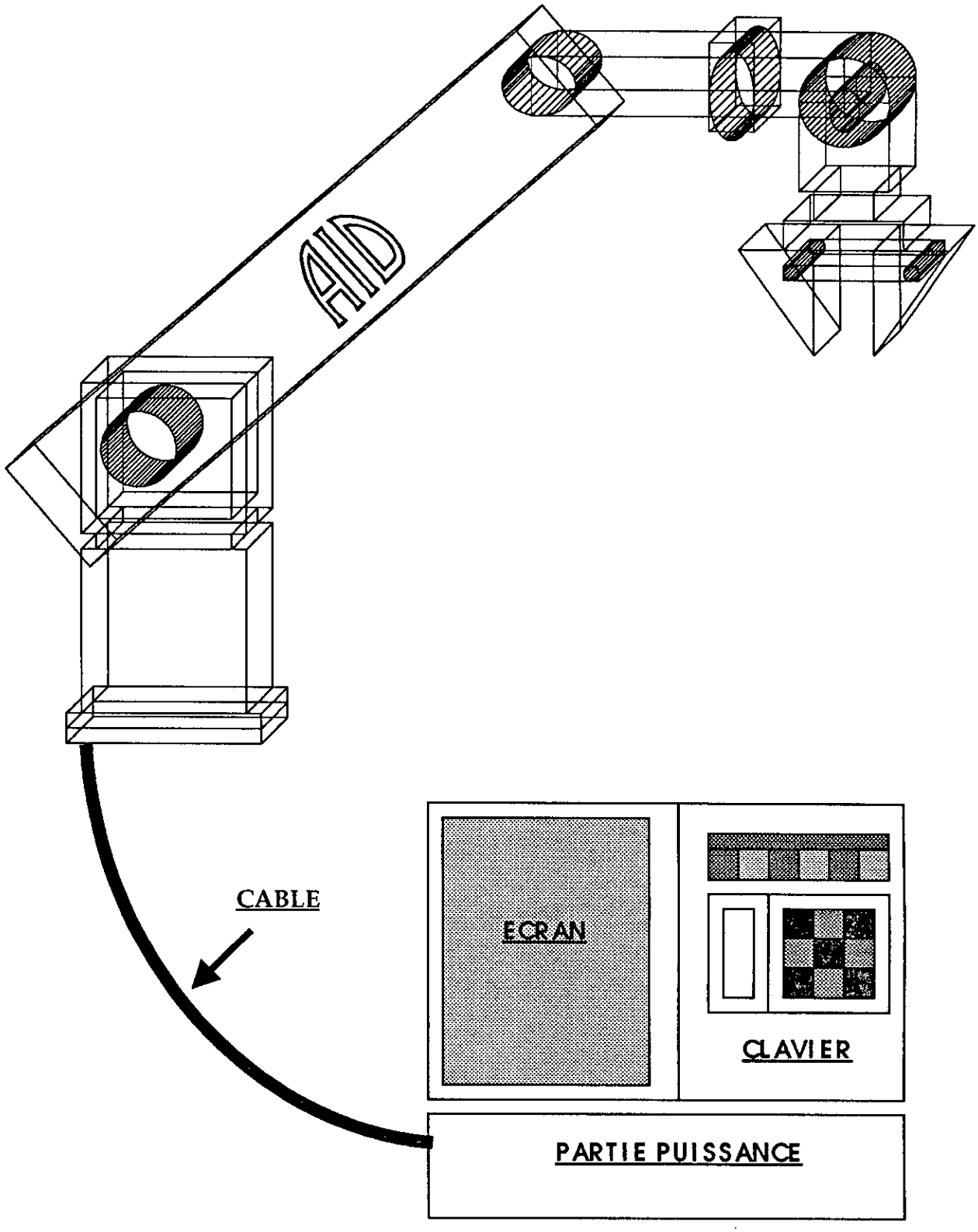


Figure I.1 : Le robot AID piloté par la commande numérique classique.

I.3. PRESENTATION DE LA CARTE DSP FLEX MOTION (8) (9):

La carte électronique DSP Flex Motion est une combinaison réussie entre les technologies avancées d'équipements hardware et software ce qui en fait un outil puissant et facile à employer dans toutes les applications de contrôle de mouvement.

Cette DSP a été conçue et réalisée par la firme américaine nuLogic en 1997, les concepteurs ont essayé de prévoir les besoins de mouvement du système (robot) et de mettre en place un modèle simple d'utilisation pour des performances maximales, ces quelques mots décrivent très clairement l'esprit de la Flex motion.

I.3.1. HARDWARE

L'aspect matériel de la Flex Motion se présente par la carte Flex Motion elle-même qui contient le processeur (figure I.2) et le bornier UMI-6 Flex qui permet la connexion entre le robot et le PC (figure I.3).

Les performances sont le résultat d'utilisation du processeur Motorola 68331 et le circuit logique AD2111 DSP, et c'est la bus d'interface FIFO (First In First Out) et la commande puissante établie qui assurent des communications en grande vitesse lors de chargement des mouvements complexes.

- **Les composants de la carte :**

P1 : connecteur à 100 broches (vers UMI-6 Flex)
P2 : bus d'interface PC/ISA/PCI.
P3 : connecteur d'entrée sortie à 50 broches.
P4 : bus à 64 broches spécifie pour d'autres cartes nuLogic
P5 : port série à 10 broches.
P6 : connecteur de test à 10 broches.
P7 : connecteur pour source de tension extérieure à 10 broches.
SW1 : interrupteur du bus d'adresse PC/ISA
JP2 : bloque de configuration de source de tension (extérieure ou intérieure).
RP3, RP10 : résistances qui correspondent aux limites entrées sorties.
RP6, RP8, RP11, RP13, RP15 : résistances qui correspondent aux limites des signaux des encodeurs.
U12 : processeur 68331.
U23 : circuit logique AD2111 DSP.

- **Les composants du bornier UMI-6 Flex :**

	Axe 1	Axe 2	Axe 3	Axe 4	Axe 5	Axe 6
Bornier du signal de commande	**J18A**	**J19A**	**J20A**	**J21A**	**J22A**	**J23A**
Bornier des signaux d'encodeur	**J2**	**J4**	**J6**	**J8**	**J10**	**J12**
Bornier de zone d'index	**J3**	**J5**	**J7**	**J9**	**J11**	**J13**

Tableau I.3. : composants du bornier UMI-6 Flex.

J14 : bornier d'entrée analogique.

J15 : bornier de fin de course électrique.

J16, J24A, J25A : borniers non utilisés.

J26A : bornier d'entrée de tension (+5V, +12V).

JP2, JP3 : bloques de sélection des moteurs pas à pas et servomoteurs.

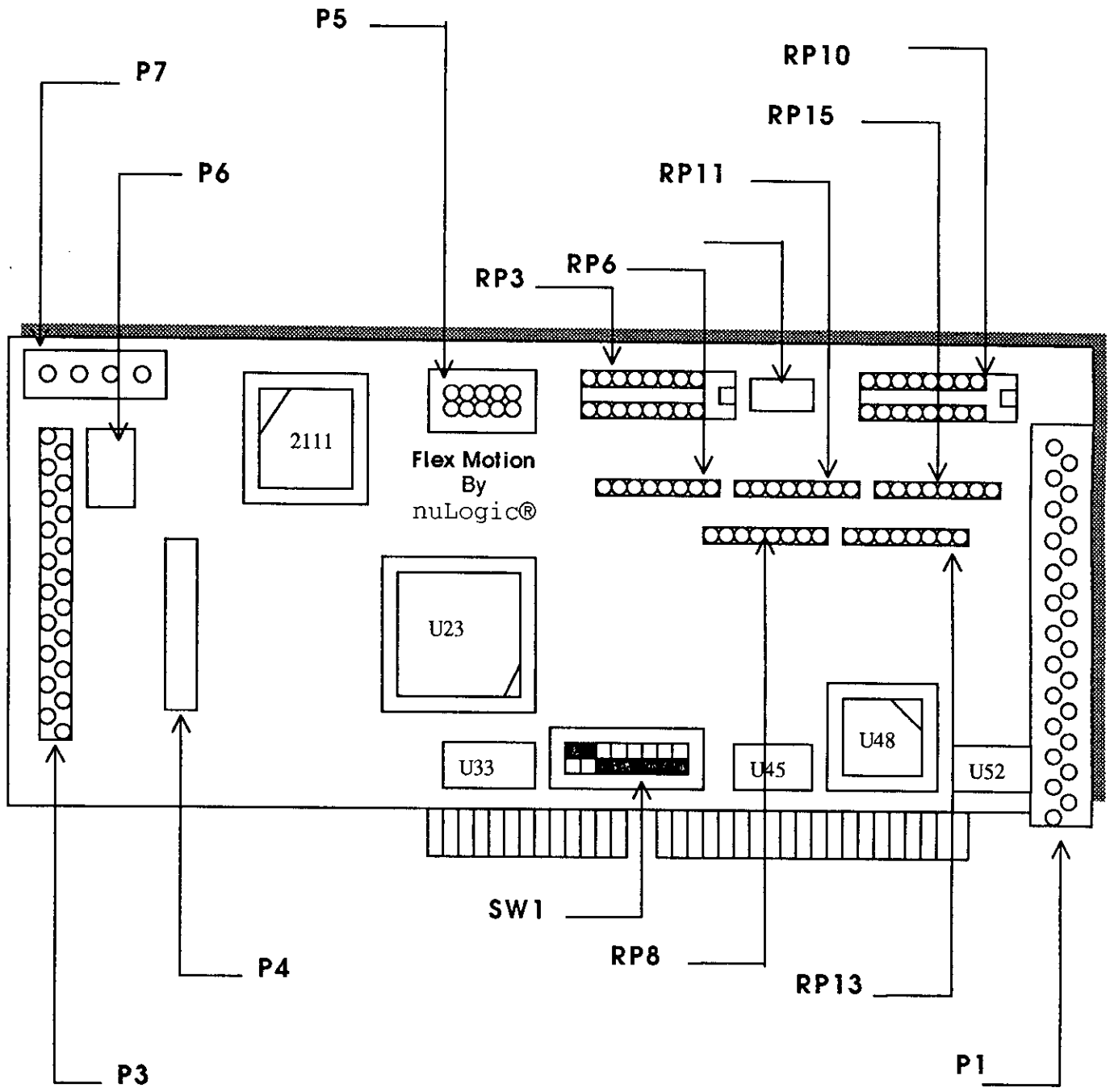


Figure I.2 : Les différents composants de la carte DSP **Flex Motion**.

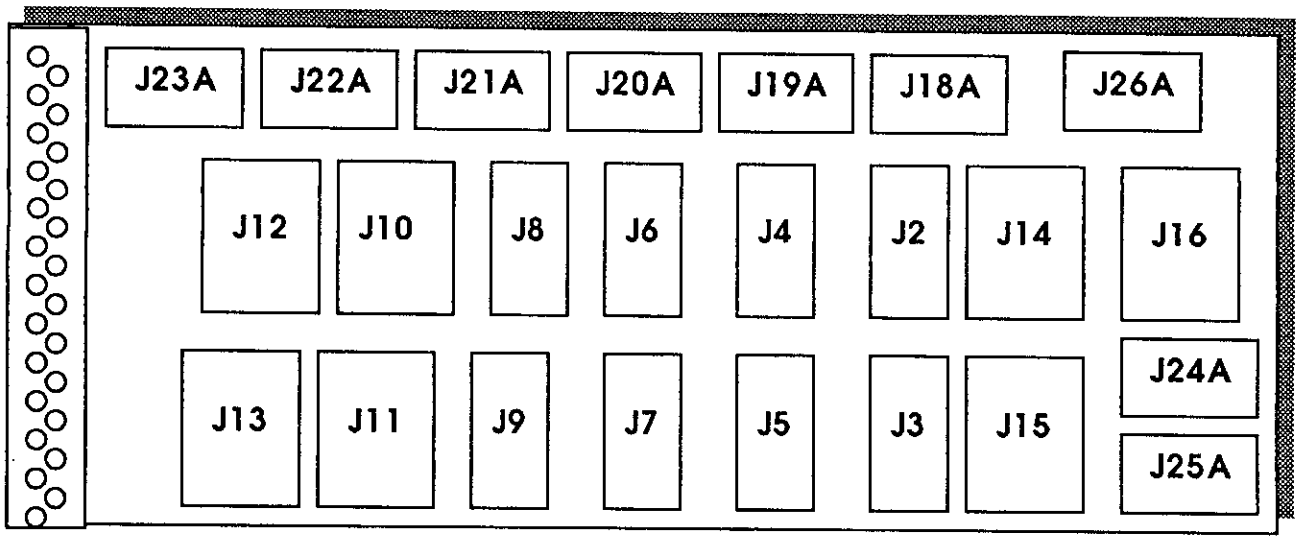


Figure I.3 : Schéma du bornier d'interface UMI 6- Flex.

I.3.2. SOFTWARE :

La programmation de la carte DSP Flex Motion peut éventuellement se faire par :

1. C ou C++.
2. VISUAL BASIC.
3. Quick Basic.
4. LabVIEW.
5. Lab Windows.

Le langage de programmation adopté est le Visual Basic version 4.0 qui est un langage visuel très puissant, et qui permet d'agir sur le programme en cours d'exécution et de changer les paramètres de contrôle en temps réel.

Lors de la programmation et la communication avec la carte DSP Flex motion, il est nécessaire d'identifier l'endroit de la carte ou sera envoyer la donnée ou la commande. Pour cela la notion de ressource prend toute son ampleur car toutes ces informations y serons stoker, cette notion peut être définie comme suit :

ϖ **Définition d'une ressource :**

« Une ressource correspond à une entité logicielle (software) ou physique (hardware) de la carte Flex Motion, ou bien la combinaison des deux qui sont très flexibles et hiérarchiques, les ressources physiques (encodeurs, DAC, ADC, ..) de bas niveau peuvent être assigner à des ressources logicielles (axe, vecteur espace,..) de haut niveau, il est donc toujours possible d'y accéder à tout instant ».

Voici à titre d'exemple quelques ressources toujours utilisées :

1. L'encodeur est une ressource ! !

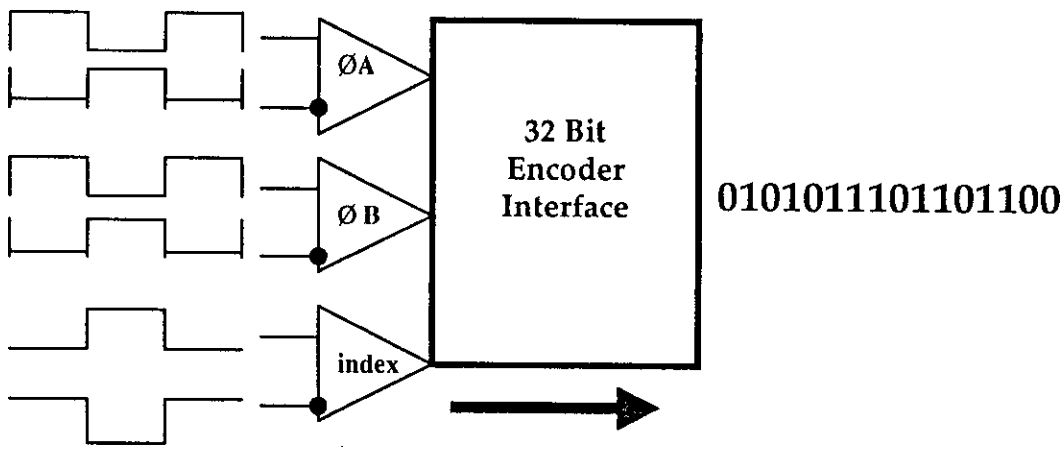


Figure I.4 : Encodeur à 32 bits

2. Le convertisseur numérique analogique est une ressource ! !

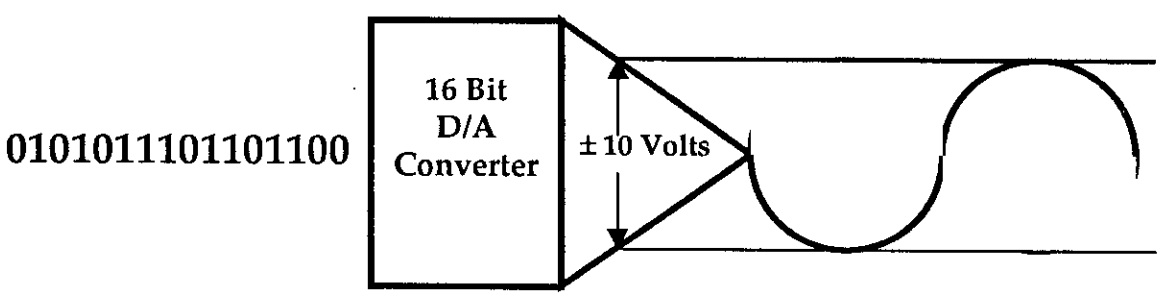


Figure I.5 : Convertisseur Numérique Analogique (DAC).

3. Un axe est une ressource I I:

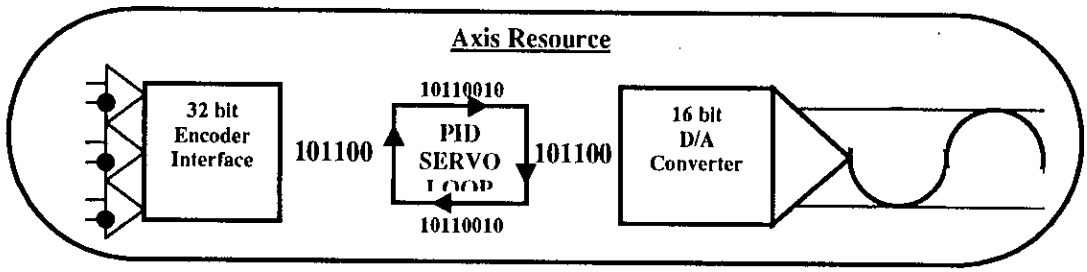


Figure I.6 : Axe ressource.

4. Le vecteur espace est une ressource I I

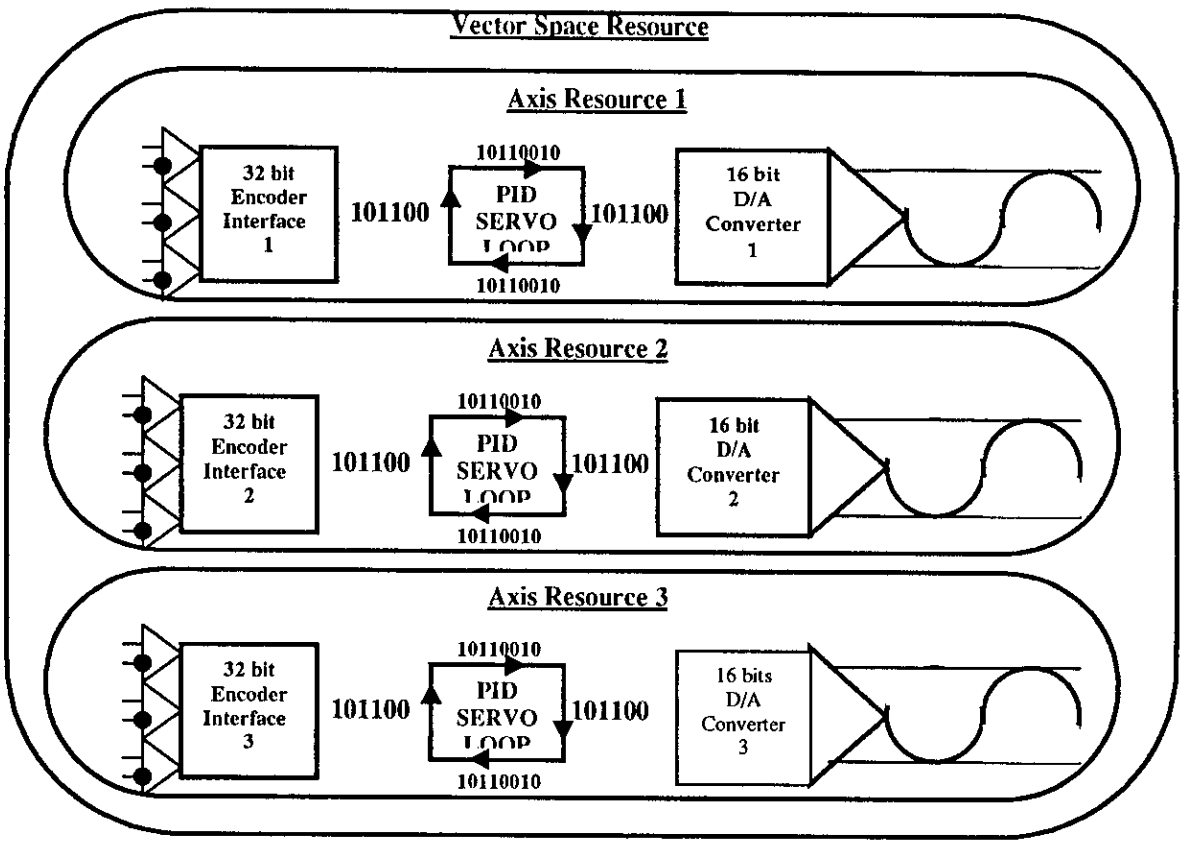


Figure I.7 : Vecteur Ressource.

5. Le convertisseur analogique numérique est une ressource I I

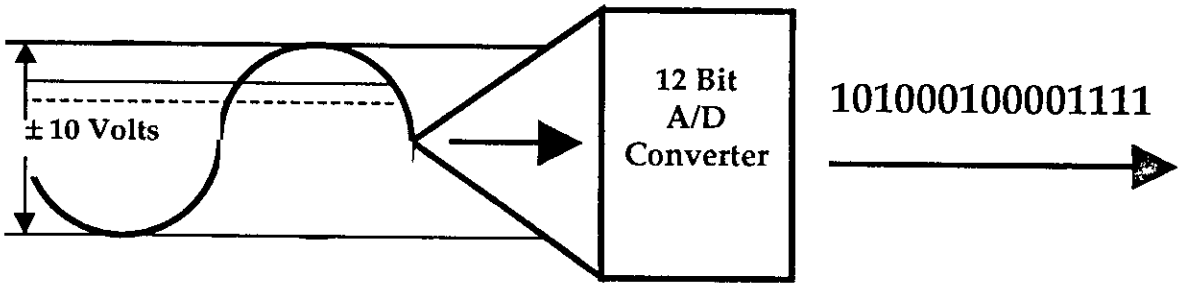


Figure I.8 : Convertisseur Analogique Numérique (ADC).

I.4. MISE EN ŒUVRE DE LA CONNEXION AID FLEX MOTION PC :

L'ancien système présenté à la figure I.1, permettait la commande du robot AID par un pupitre de commande numérique doté d'un clavier et d'un moniteur, la mise en oeuvre de la connexion de l'AID avec le PC à travers la carte Flex Motion nous a permis de conserver uniquement la partie puissance, le dispositif actuel se présente comme sur la figure (I.9).

I.4.1. LIAISONS PHYSIQUES :

Les liaisons physiques, c'est à dire l'identification des entrées sorties de la partie puissance et les branchements avec le bornier UMI6- Flex ont été effectuées dans (24).

Par ce dispositif le robot est fin prêt à recevoir les commandes directement à partir du PC, à travers une programmation adéquate, le chapitre III traite cette partie programmation.

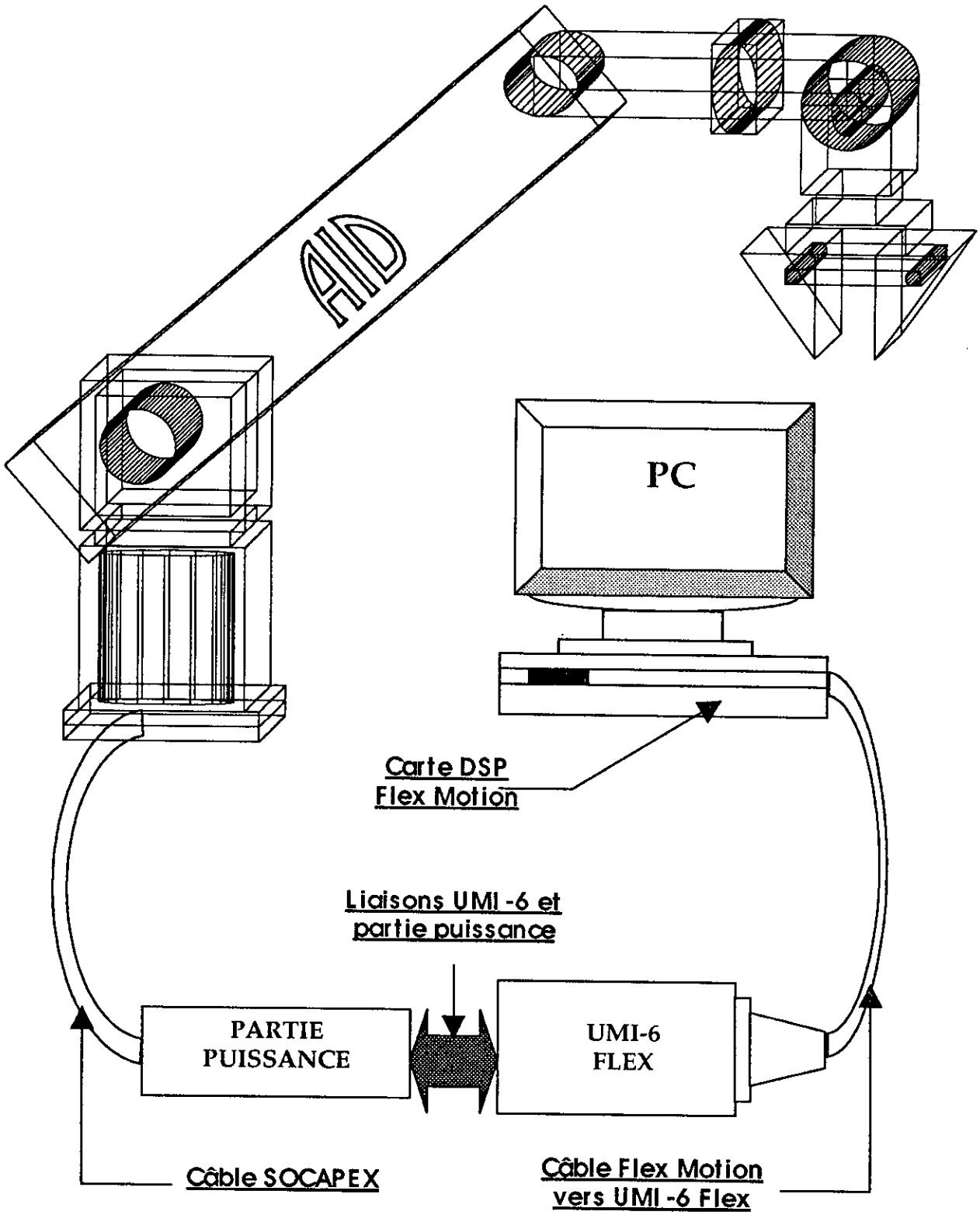


Figure I.9 : Nouveau dispositif de commande du robot AID via la DSP Flex Motion.

I.4.2. POSSIBILITES LOGICIELLES :

I.4.2.1. SYNTAXE DES COMMANDES FLEX MOTION :

La simplicité de programmation de la carte Flex Motion est à la hauteur de ses performances incroyables, cette programmation s'effectue à travers une multitude de commandes propre à la carte (annexe A) et qui obéissent toutes à un protocole ou convention d'écriture suivant :

flex_vvvv_nnnn_(_rtn)(BD_TYPEDEF board_id_or_addr, (BYTE device, DATA data1,data2,...) (BYTEretved IBYTE inpved I RETDATA ptr))

Tel que :

vvvv : est un verbe qui décrit l'action de la commande à exécuter, par exemple :

action de chargement	Flex_**load**...
action de lecture	Flex_**read**...
action d'activation	Flex_**enable**...
action de configuration	Flex_**config**...
action d'assignation	Flex_**assign**...
action d'effacement	Flex_**clear**...
action d'initialisation	Flex_**reset**...
action de sauvegarde	Flex_**save**...
action d'écriture	Flex_**write**...

nnnn : est un nom qui décrit la fonction ou ressource ou paramètre dont la commande doit affecter ou contrôler, par exemple :

lecture de la consigne	Flex_read_**target_pos**...
charger le gain proportionnel	Flex_load_**kp**...
activer ou désactiver les encodeurs	Flex_enable_**encs**...
Configurer les axes	Flex_config_**axis**...
attribution du mode opérationnel	Flex_set_**op_mode**...
charger une trajectoire existante	Flex_load_**helix**...

_rtn : Ce suffixe est un indicateur qui spécifie que la commande appliquée doit retourner ses valeurs à l'environnement du PC (affichage sur écran) ou bien à une case mémoire indiquée par un pointeur RETDATAptr (return data pointer).

_inputvect : Cet argument spécifie une adresse (input vector) et indique à la carte Flex Motion d'où provienne les valeurs nécessaires à l'exécution de la commande (consignes, gains, etc..), en général inputvect est égal en hexadécimal à FF.

I.4.2.2. PERIODE D'ECHANTILLONNAGE DE LA COMMANDE :

La commande Flex Motion suivante:

flex_enable_axes(boardID, BYTEdevice, BYTEpidrate, BYTEenabledaxes)

permet d'effectuer deux opérations à la fois, premièrement d'activer ou désactiver les axes destinés au mouvement et deuxièmement de fixer la période d'échantillonnage pour le calcul du signal de contrôle.

Il existe une correspondance entre le nombre d'axes activés et la période d'échantillonnage admise par la boucle de contrôle, ce choix s'effectue comme il est indiqué sur le tableau I.4.

Si la période d'échantillonnage BYTEpidrate ne correspond pas au nombre d'axes activés BYTEenabledaxes un message d'erreur est emmi pour le signaler.

Nombre d'axes activer	Période d'échantillonnage
1	62µs
2	125µs
3	188µs
4	250µs
5	312µs
6	375µs

Tableau I.4 : Choix des périodes d'échantillonnage en fonction du nombre d'axes.

I.4.2.3. Contrôle en SERVO ou STEPPER :

La commande Flex Motion :

flex_set_axis_mode (boardID, BYTEdevice, WORDaxismode)

permet de déterminer le mode de fonctionnement pour chaque axe.
Un axe peut être configurer en fonctionnement **SERVO** (commande en boucle fermée) ou **STEPPER** (commande de moteur pas à pas), certains axes peuvent fonctionner

uniquement en **SERVO** d'autres uniquement en **STEPPER**, le choix peut s'effectuer en assignant les bits correspondant à WORDaxismode au fonctionnement désiré pour chaque axe désigné par BYTEdevice.

Deux types d'algorithmes sont disponibles pour un fonctionnement en **SERVO** :

a) **PID** : correspond à l'algorithme standard du régulateur **P**roportionnel **I**ntégral **D**érivé, une commande en boucle fermée classique utilisant le signal d'erreur en association avec les différent gains statiques (Kp, Ki, Kd) pour l'obtention d'une bonne stabilité du dispositif.

b) **PIVFF** : (**P**roportionl **I**ntegral **F**eed**F**orward) cet algorithme résulte d'une adaptation du régulateur **PID**, il a suffit d'annuler l'action dérivée et de rajouter un signal d'anticipation de vitesse au PI, ce type de régulateur est très performant lors de mouvement lents.

Trois types d'algorithmes sont disponibles pour un fonctionnement en **STEPPER** :

a) **OPEN** : la commande de moteur pas à pas s'effectue en boucle ouverte, il suffit d'attaquer les DAC par une tension de commande est obtenir la position associée à cette commande appliquée.

b) **NORM** et **HOLD** : la commande du moteur pas à pas se fait en boucle fermée.

I.4.2.4. PARAMETRES DE CONTROLE :

Nous proposons une brève présentation des différents gains de contrôle.

ϖ **Le gain proportionnel Kp :**

flex_load_kp(boardID, BYTEdevice, WORDkp, BYTEinpvect)

BYTEdevice : axe considéré.
WORDkp : valeur numérique du gain proportionnel.
BYTEinpvect : adresse de la valeur Kp.

Cette commande charge la valeur du gain proportionnel nécessaire au filtre digital qui est en fait le régulateur associé à l'axe sélectionné par BYTEdevice.

A chaque instant d'échantillonnage l'erreur de position est calculée et multipliée par le gain Kp pour produire la tension de commande sur le DAC à l'axe BYTEdevice.
Pour calculer la contribution de l'action proportionnelle sur la sortie du DAC à 16 bit, la tension de contrôle est calculée par l'expression suivante :

$$Vout(proportional) = (20volts/2^{16}) * Kp * position\ error \quad \ldots\ldots\ldots (I.1)$$

ϖ **Le gain intégral Ki :**

flex_load_ki(boardID, BYTEdevice, WORDki, BYTEinpvect)

BYTEdevice : axe considéré.
WORDki : valeur numérique du gain intégral Ki.
BYTEinpvect : adresse de la valeur Ki.

Le gain intégral assure une erreur statique de position nulle dans la boucle de régulation.

Une valeur non nulle de Ki, tend à augmenter l'erreur de position lors d'une accélération ou lors d'une décélération des actionneurs.
Cet effet peut être adouci par l'emploi du paramètre ILIMIT, de trop grandes valeurs de Ki souvent cause dans la boucle de régulation une instabilité, pour ces raisons il serait plus judicieux de choisir Ki égal à zéro jusqu'à ce que le système opère stablement, et alors une petite valeur de Ki peut être ajoutée pour minimiser l'erreur de position statique.

A chaque instant d'échantillonnage l'erreur est calculée est rajoutée à l'accumulation des erreurs précédentes pour effectuer une intégration numérique classique.
Pour calculer la contribution de l'action intégrale sur la sortie du DAC à 16 bit, la tension de contrôle est calculée par l'expression suivante :

$$Vout\ (integral) = (20volts/2^{16}) * Ki * Limit(integralsum, ILIMIT). \quad .....(I.2)$$

ϖ **La limite d'intégration ILIMIT :**

flex_load_ilim(boardID, BYTEdevice, WORDilim, BYTEinpvect)

BYTEdevice : axe considéré.
WORDilim : valeur numérique de la limite d'intégration ILIMIT.
BYTEinpvect : adresse de la valeur ILIMIT.

Peut être utilisée pour limiter les forces excessives et pour minimiser l'effet adverse que la compensation intégrale peut provoquer lors d'une accélération ou une décélération des actionneurs, ILIMIT ne peut avoir d'effet lorsque le gain Ki est nul.
Dans (I.2) ILIMIT est comparée à la somme d'intégration **integralsum** et la plus petite valeur des deux est multipliée par Ki.

ϖ **Le gain dérivé Kd :**

flex_load_kd(boardID, BYTEdevice, WORDilim, BYTEinpvect)

BYTEdevice : axe considéré.
WORDilim : valeur numérique du gain de dérivation Kd.
BYTEinpvect : adresse de la valeur Kd.

Le gain dérivé contribue à restituer la force proportionnelle à la variation de l'erreur de position, cette force agit comme un facteur d'amortissement visqueux.

A chaque période d'échantillonnage l'erreur de position est calculée, ainsi la contribution de l 'action dérivée sur la sortie des DAC est calculée par l'expression suivante :

$$Vout\ (derivative) = (20volts/2^{16}) * Kd * (pos\_err(t) - pos\_err(t0)) \quad .....(I.3)$$

Tel que : t-t0=Td

ϖ **Constante Td :**

flex_load_td(boardID, BYTEdevice, WORDtd, BYTEinpvect)

BYTEdevice : axe considéré.
WORDilim : valeur numérique de la constante Td.
BYTEinpvect : adresse de la valeur Kd.

Td est différente de la période d'échantillonnage de la commande (62µs / axe). Cette valeur détermine la fréquence de calcul de la variation de l'erreur de position, elle est calculée par l'expression suivante :

$$Derivative\ Sample\ Period = (Td+1) * Sample\ Period. \quad .....(I.4)$$

Tout ces paramètres peuvent également être charger en même temps par la commande :

Flex_load_loop_params (boardID, BYTEdevice, pidval, BYTE inpvect)

Tel que pidval est un vecteur contenant la totalité des paramètres cités ci avant.

La commande envoyée sur la sortie des DAC à pour expression :

Vout = Vout(proportional) + Vout (integral) + Vout (derivative)

I.4.2.5. PARAMETRES DES ACTIONNEURS :

La DSP Flex Motion nous permet également de charger les paramètres des six actionneurs de la manière suivante :

ϖ **Chargement du profile de vitesse désiré :**

Les commandes permettant le chargement du profile de vitesse désiré sont :

Vitesse de l'actionneur :

Flex_load_vel (boardID, BYTEdevice, DWORD vel, BYTEinpvect)

Accélération de l'actionneur :

Flex_load_accel (boardID, BYTEdevice, DWORD accel, BYTEinpvect)

Décélération de l'actionneur :

Flex_load_decel (boardID, BYTEdevice, DWORD decel, BYTEinpvect)

ϖ **Lissage du profile de vitesse :**

Le chargement du coefficient de lissage s'effectue par la commande :

Flex_load_scurve_time (boardID, BYTEdevice, DWORD scurveval,.... BYTEinpvect)

Tel que :

BYTEdevice : axe considéré.

DWORD : valeur numérique de la vitesse, accélération ou décélération ou du coefficient de lissage.

BYTEinpvect : adresse de la valeur considérée.

I.4.2.6. MOUVEMENT DES AXES :

Grâce aux commandes :

Flex_start (boardID, BYTEdevice, WORD axismap)
Flex_stop (boardID, BYTEdevice, WORD axismap)

BYTEdevice : axe considéré (en général c'est le vecteur control qui est considéré).

WORD axismap : *mapping des axes* destinés au mouvement ou à l'arrêt.

Le mouvement des 6 axes est synchronisé au démarrage (**flex_start**), il est également possible de les arrêter en même temps (**flex_stop**).si le signal de commande de l'un des axes n'arrive pas à celui ci, le mouvement total ne s'effectue pas, et un signal d'erreur est émis.

I.4.2.7. LECTURE DES DONNEES :

La DSP Flex Motion permet aussi l'accès aux données numériques pendant l'évolution du robot dans son aire de travail.

L'accès intéressant pour notre application se situe au niveau des encodeurs (lecture de la position en temps réel de l'axe considéré) et les DAC (lecture de l'entrée de commande aux actionneurs pendant le mouvement d'un axe considéré).
Ces lectures sont rendues possibles grâce aux commandes Flex Motion suivantes :

flex_read_encoder_rtn (boardID, BYTEdevice, encodercounts)

flex_read_dac_rtn (boardID, BYTEdevice, dacvalue)

BYTEdevice : axe considéré.
encodercounts : valeur numérique de l'encodeur l'axe considéré.
Dacvalue : valeur numérique du DAC de l'axe considéré.

I.4.2.8. MODES OPERATIONNELS :

La DSP offre également cinq modes opérationnels (annexe A : tableau A.6), nous citons les deux modes nécessaires à notre application :

- **Mode opérationnel relatif** : permet aux actionneurs de bouger par rapport à la valeur du compteur de position associé à chacun d'entre eux, si par exemple le compteur affiche 100, et que la consigne est égale à 150 l'actionneur effectuera une distance de 50, si par contre la consigne est égale à 100 l'actionneur restera immobile.
- **Mode opérationnel absolu** : permet aux actionneurs de bouger en rajoutant à la valeur du compteur de position associé à chacun d'entre eux, la consigne de position désirée si par exemple le compteur affiche 100, et que la consigne est égale à 150 l'actionneur effectuera une distance de 250.

Les possibilités logicielles de la carte DSP Flex Motion sont très nombreuses, il nous est impossible de les présentées toutes en détail, la totalité des commande Flex Motion sont présentés en annexe A.

I.5. CONCLUSION :

Au terme du chapitre I, nous aboutissons au conclusions suivantes :

L'instrumentation du robot AID permet dans un premier temps de faire de la régulation de position, dans un second temps faire de la programmation niveau effecteur et cela grâce à son association à la carte DSP Flex Motion.

En effet, la carte DSP est destinée au robot AID (ou tout autre robot du même type), ceci est très facilement vérifiable puisque la majorité des commandes Flex Motion sont adressées dans leur syntaxe au robot lui même (c'est à dire aux axes).

Cette carte, prend en charge plusieurs aspect de la commande puisque l'algorithme PID est disponible au niveau de la carte de ce fait la génération de la commande est l'adaptabilité de celle ci vis à vis de la saturation est également prise en charge par la DSP.

Ces avantages certains permettent à l'utilisateur de se situer à un niveau supérieur de la programmation du robot, il passe donc de la programmation du *niveau actionneur* à la programmation du *niveau effecteur*, le robot sera ainsi exploiter pour des taches plus intéressantes.

« Chapitre II »

Modélisation du robot AID

II.1. INTRODUCTION :

On peut scinder les robots en trois grandes classes : les bras articulés ou manipulateurs, les télémanipulateurs, et enfin les robots mobiles.

Industriellement, le bras articulé est le plus, le seul pourrait on ajouter, répandu. Le télémanipulateur répond aux besoins de quelques industriels ou laboratoires travaillant en milieux difficiles ou hostiles pour l'homme. quant aux robots mobiles sont pour l'essentiel du domaine des laboratoires (6).

Nous nous intéressons plus particulièrement au produit industriel, c'est à dire au robot manipulateur.

La description de n'importe quelle nature soit elle, se fait dans la majeur partie des cas, non pas au niveau du robot (articulations) mais au niveau du porteur. La pièce extrémité porte un effecteur ou outil qui peut être simple (ventouse aspirante, pince à deux mors) ou complexe (poignet compliant , main articulée) (6) (20).

Le chapitre II est consacré à la modélisation des robots, et plus particulièrement à la modélisation du robot AID pour une utilisation ultérieure.

II.2. MODELISATION :

Toute maîtrise de l'action imposée à l'effecteur exige la connaissance des positions à tout instant de tous les corps intermédiaires (*modèle géométrique*), de la vitesse des déplacements relatifs entre eux (*modèle cinématique*), les liaisons entre corps intermédiaires sont motorisés séparément. L'action de l'ensemble des moteurs induit une configuration du bras articulé et un effort en effecteur (*modèle dynamique*) (4).

La maîtrise de tout les modèles décrit, ou de tous les comportements est essentielle pour définir les taches précises que doit effectuer l'organe terminal (*synthèse de la commande*).

Le robot est composé de plusieurs corps connectés par des liaisons à un seul degré de liberté (ddl) de translation ou de rotation, cette structure mécanique constitue une chaîne continue ouverte. Les chaînes continues ouvertes sont simples sur le plan conception et largement exploitées, et ont donné naissance à plusieurs génération de robot pratiquement chez tous les constructeurs.

Le robot est composé de n articulations avec n+1 corps notés L0,...,Ln tel que L0 est affecté au bâti fixe au sol et Ln est affecté à l'organe terminal qui est destiné à effectuer une tache bien précise à travers le mouvement de l'organe terminal, notre intérêt principal est donc d'étudier le mouvement du corps n. afin de

représenter la position est l'orientation de l'organe terminal, nous affectons un repère $R_j(O_j - x_j y_j z_j)$ à chaque corps du bras manipulateur.

La méthode est basée sur les règles générales suivantes (23):

(a) Les articulations sont supposées rigides.

(b) Les mouvements articulaires sont des rotations ou des translations axiales, les liaisons à plusieurs ddl sont décomposées en autant de liaisons à un seul ddl par adjonction de corps fictifs.

(c) Les repères R_j sont fixent dans le mouvement de la liaison j $(j=\overline{1,n})$.

(d) La coordonnée généralisée q_j $(j=\overline{1,n})$ caractérise le mouvement au niveau de la liaison j.

a) Espace d'état : (20) (23)

L'espace d'état donne des informations sur la configuration du robot et caractérise les positions et les orientations relatives des différents corps constitutifs.

Dans le cas des robots en chaîne ouverte, les variables aux liaisons sont utilisées pour spécifier cet espace donc la dimension est égale au nombre de liaisons.

Toute configuration géométrique peut être repérée par le vecteur des variables des positions axiales dit aussi vecteur des coordonnées généralisées suivant :

$q=[q_1,q_2,.....,q_n]^T$ de dimension (n×1) .

b) Espace opérationnel : (20) (23)

Cet espace concerne la position et l'orientation de l'organe terminal dans un repère fixe lié au sol qui peuvent être caractérisé par la vecteur suivant :

$x=[x_1,x_2,.....,x_n]^T$ de dimension (m×1).

II.2.1. MODELISATION GEOMETRIQUE :

Il est impératif de maîtriser le passage rapide de référentiels à d'autres référentiels, compte tenu des liaisons existantes entre corps intermédiaires. Les méthodes sont très nombreuses, (coordonnées généralisées, angles d'EULER (2), paramètres de DENAVIT HARTENBERGER (2) (3), paramètres de DENAVIT HARTENBERGER modifiés (23))

II.2.1.1. MODELISATION GEOMETRIQUE DIRECTE :

Le modèle géométrique direct est traduit par un ensemble d'équations d'où l'on peut déduire la position et l'orientation de l'organe terminal en fonction des variables articulaires.

Cette position et orientation de l'organe terminal découlent de la connaissance des variables articulaires, elles sont décrites sous la forme matricielle suivante (23):

$$x = k(q) \quad \text{.....(II.1)}$$

L'usage des coordonnées cartésiennes pour la position, et des cosinus directeurs pour l'orientation.

Le modèle géométrique direct donne une solution unique et son calcul est toujours possible, il ne présente pas de grand intérêt en phase d'exploitation, son rôle se réduit à la délimitation du volume de travail, calcul des vitesses et des accélérations maximales en bout de chaîne et identification des paramètres géométriques.

ϖ Notation de DENAVIT HARTENBERGER : (2)

La transformation de DH permet de décrire de façon systématique la relation géométrique entre une paire de corps adjacents appartenant à une structure en chaîne ouverte, elle est basée sur une représentation matricielle (4x4) de la position et l'orientation du corps considéré (Figure I.1).

Le repère R_j est défini de la manière suivante :

- Les axes z_j portent le mouvement au niveau des liaisons.
- L'axe x_j est selon la perpendiculaire commune aux axes z_j et z_{j+1}, s'ils sont parallèles des considérations de symétrie ou de simplification de l'interprétation des résultats peuvent orienter le choix de la perpendiculaire à retenir.

La position relative des deux repères peut être complètement définie grâce aux quatre paramètres suivants :

- a_j : la longueur de la normale commune des axes z_j et z_{j+1} distance constante dans le cas d'une rotation.
- d_j : la distance entre les axes z_{j-1} et z_j le long de x_{j-1}, l'origine O_{i-1} et le point H_j.

- α_j : angle fixe entre les axes z_{j-1} et z_j .
- $\theta_j = q_j$: l'angle entre l'axe et la normale commune.

La matrice de passage définissant respectivement la position du repère i par rapport au repère i-1 est donnée par (II.2):

$$A_j^{j-1} = \begin{bmatrix} \cos\theta_j & -\sin\theta_j \cos\alpha_j & \sin\theta_j \sin\alpha_j & a_j \cos\theta_j \\ \sin\theta_j & \cos\theta_j \cos\alpha_j & -\cos\theta_j \sin\alpha_j & a_j \sin\theta_j \\ 0 & \sin\alpha_j & \cos\alpha_j & d_j \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{.....(II.2)}$$

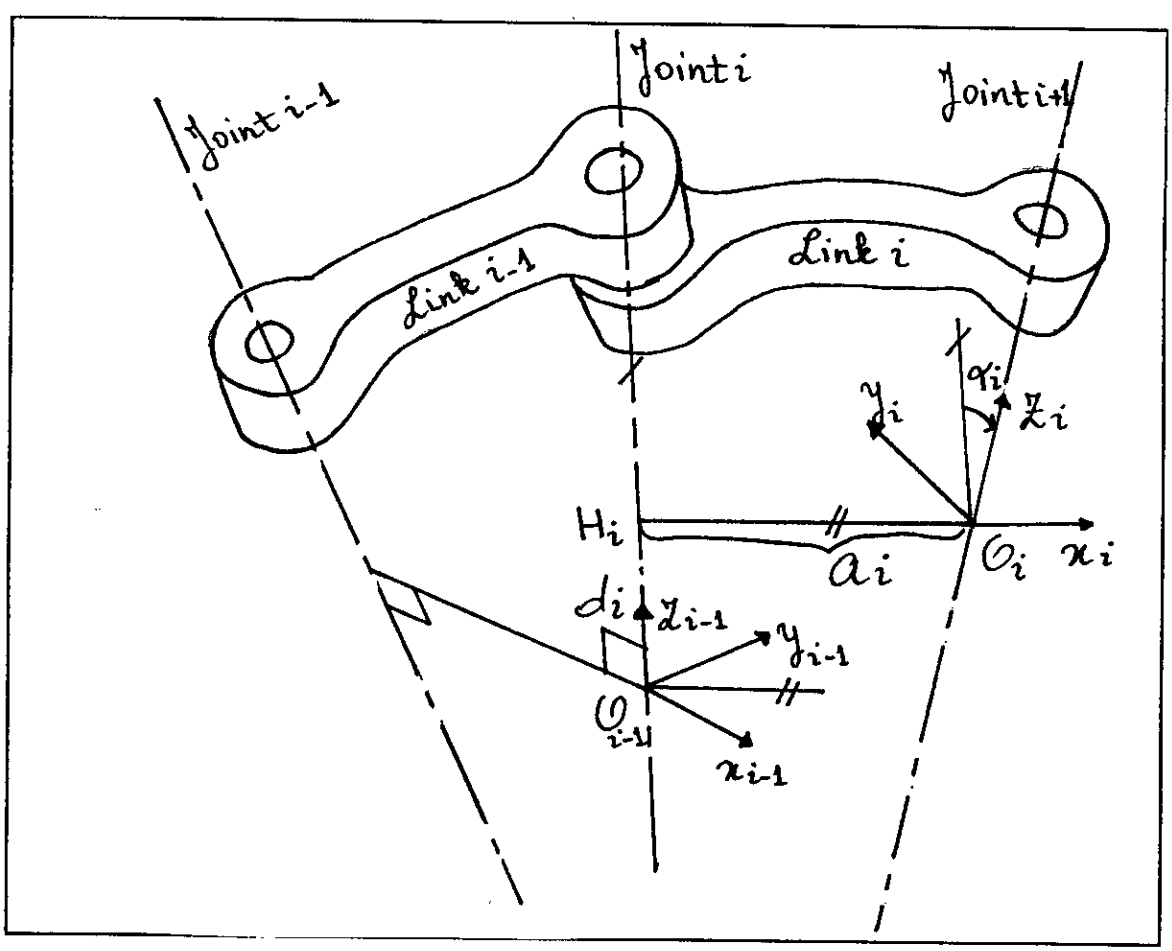


Figure II.1 : Notation de DENAVIT HARTENBERGER.

II.2.1.2. MODELISATION GEOMETRIQUE INVERSE :

Il donne la configuration géométrique du robot correspondant aux contraintes de positionnement et d'orientation imposées à l'effecteur, il permet donc le calcul des variables articulaires (cordonnées généralisées) en fonction des cordonnées opérationnelles.

La solution du modèle géométrique inverse n'est pas unique, mais afin de dégager la solution la plus adaptée, il suffit d'imposer des contraintes supplémentaires.

Les limites physiques de débattement peuvent exclure certaines solutions.

Sur le plan pratique, le modèle géométrique inverse présente un intérêt indéniable pour le bras manipulateur, il permet de transcrire les besoins de positionnement et d'orientation de l'effecteur face à une tache donnée dans le système (espace des variables articulaires utilisées par la commande, en d'autres termes le nombre de degrés de liberté introduit par la commande), tout robot doté de modèle géométrique inverse est un robot versatile.

La détermination des variables articulaires $q^T = [q_1, q_2, \ldots\ldots, q_n]$ en fonction de la position désirée n'est pas chose facile, il existe néanmoins trois classes de méthodes pour la détermination du modèle géométrique inverse :

Méthode géométrique : cette méthode se base sur la structure géométrique du robot, le but est de déterminer les expressions des coordonnées généralisées en fonction des coordonnées opérationnelles.

Cette méthode est très efficace lorsqu'il s'agit de robot à structure simple (robot plan à deux degrés de liberté par exemple).

Méthode analytique : ce type de méthode se base sur le modèle géométrique direct, on peut citer la méthode de Paul (23) qui procède par isolement successifs des différentes variables articulaires en commençant par q_1 en exploitant totalement les matrices de passage, et peut être appliquée à tout les types de robots.

Méthode numérique : lorsque le problème du modèle géométrique inverse est irrésolvable par les deux méthodes précédentes, on a souvent recours aux méthodes numériques qui prennent en charge la résolution d'un système d'équations non linéaires.

Dans ce cas le modèle géométrique n'est évidement pas obtenu sous forme d'expressions explicites, mais sous forme de valeurs numériques qui sont traitées par des calculs itératifs qui posent des problèmes de convergence et de temps de calcul.

II.2.2. MODELISATION CINEMATIQUE :

II.2.2.1. MODELISATION CINEMATIQUE DIRECTE :

Le modèle cinématique direct découle directement du modèle géométrique direct, il exprime les vitesses de rotation de translation de l'effecteur en fonctions des vitesses aux liaisons q et des variables articulaires ainsi que les paramètres géométriques.
En dérivant le modèle géométrique direct (II.1), il en résulte le modèle cinématique suivant (3):

$$\begin{cases} V = J(q)\dot{q} \\ J_{ij} = \dfrac{\partial k_i(q)}{\partial q_j} \end{cases} \qquad i = \overline{1,m} \text{ et } j = \overline{1,n} \quad \text{.....(II.3)}$$

Tel que $J(q)$ est la matrice Jacobienne de dimension (mxn) qui est obtenu de la différentiation du modèle géométrique, et J_{ij} sont les composantes de la matrice $J(q)$.

II.2.2.2. MODELISATION CINEMATIQUE INVERSE :

Comme pour le modèle géométrique inverse l'objectif reste le même quant à la résolution du modèle cinématique inverse mais en terme de vitesse de l'effecteur.
Ce modèle permet donc le calcul des vitesses articulaires en fonction des exigences de la tache en terme de vitesse de translation et de rotation de l'organe terminal.
Pour un bras manipulateur à 6 ddl, le Jacobien est de dimension 6x6, si cette matrice est non singulière aux différentes configurations du bras manipulateur, la matrice inverse notée J^{-1} existe, la solution s'écrit donc sous la forme suivante :

$$\dot{q} = J^{-1}(q)\dot{p} \quad \text{.....(II.4)}$$

Sachant que le Jacobien $J(q)$ change avec la variation de la configuration du bras manipulateur, cette matrice peut être singulière par moment, l'équation (II.4) n'est donc plus valable, (2) propose donc une solution optimale pour résoudre le problème cinématique inverse, cette solution est obtenue par la méthode de la matrice pseudo inverse.

Cette se propose de déterminer à partir du vecteur $\dot{q}$ qui satisfait l'équation (II.4) pour $\dot{p}$ et $J(q)$ donnés le modèle cinématique inverse, tout en minimisant une fonction coût donné par l'expression suivante :

$$G(\dot{q}) = \dot{q}^T W \dot{q} \text{ .....(II.5)}$$

Tel que W est une matrice de pondération constante définie positive de dimension (nxn), en effectuant des manipulations mathématiques la solution se présente comme suit :

$$\dot{q} = W^{-1} J^T \left(J W^{-1} J^T\right)^{-1} \dot{p} \text{ .....(II.6)}$$

Si la matrice de pondération W est égale à la matrice identité l'équation (II.6) s'écrit donc :

$$\dot{q} = J^{\#} \dot{p} \text{ .....(II.7)}$$

Avec $J^{\#} = J^T \left(JJ^T\right)^{-1}$ appelé la matrice pseudo inverse de la matrice Jacobienne $J(q)$.

Le modèle cinématique inverse présente un grand intérêt lors de l'exploitation surtout quand le robot travaille en cours de trajectoire avec des contraintes de vitesse en déplacement ou en orientation de l'effecteur (par exemple un cordon de soudage uniforme nécessite une vitesse constante de la torche à souder).

II.2.3. MODELISATION DYNAMIQUE :

L'analyse du comportement dynamique du robot est traduit par un modèle dit dynamique donc qui permet de calculer les torseurs aux niveaux des liaisons en fonction des variables cinématiques $(q_i, \dot{q}_i, \ddot{q}_i)$ et des paramètres géométriques, de n groupes de paramètres inertiels du robot et des efforts extérieurs ainsi que le torseur exercé par l'environnement sur l'organe terminal (une charge massique ou une pression).

Au cours d'une application les grandeurs cinématiques articulaires ainsi que le torseur Ft sont sujets à variations, la commande seul des couples portes par les axes sont nécessaires.

L'équation dynamique fondamentale d'un robot à n ddl est sous la forme générale suivante (3) (23):

$$M(\theta)\ddot{\theta}+N(\theta,\dot{\theta})+G(\theta)+H(\dot{\theta})=T \quad \text{.....(II.8)}$$

tel que :

$\theta(t)$: vecteur positions angulaires de dimension nx1.

$T(t)$: vecteur des torseurs appliqués de dimension nx1.

$M(\theta)$: matrice d'inertie définie positive de dimension nxn.

$N(\theta,\dot{\theta})$: vecteur des torseurs dus aux forces de Corriolis et centrifuges de dimension nx1.

$G(\theta)$: vecteur des torseurs dus à la gravitation de dimension nx1.

$H(\dot{\theta})$: vecteur des torseurs dus aux frottements visqueux de dimension nx1.

Les éléments $M(\theta)$, $N(\theta,\dot{\theta})$, $G(\theta)$ et $H(\dot{\theta})$ sont fortement non linéaires en $\theta,\dot{\theta}$.

Deux grandes approches existent. Celle de NEWTON EULER (2) qui décrit le comportement dynamique d'un système en terme de forces et de moments. Celle de Lagrange (2) (3) (23) qui le décrit en terme de travail et d'énergie.

II.2.3.1. FORMULATION LAGRANGIENNE :

Le comportement dynamique du système est exprimé sous forme d'énergie et de travail utilisant les coordonnées généralisées, toutes les forces et les contraintes sont automatiquement éliminés, l'équation qui résulte directement de cette formulation est sous une forme compacte, cette formulation est très simple pour la formulation du modèle dynamique direct.

La déduction du modèle dynamique direct passe par le calcul du Lagrangien donné par :

$$L(q,\dot{q})=Uc-Up \quad \text{.....(II.9)}$$

tel que :

Uc : représente l'énergie cinétique totale du système.

Up : représente l'énergie potentielle totale du système.

Le modèle dynamique direct est calculé suivant :

$$T_i=\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right)-\frac{\partial L}{\partial q_i} \quad \text{.....(II.10)}$$

Tel que T_i est la force généralisée qui correspond à la coordonnée généralisée q_i.

L'expression de l'énergie cinétique est donnée par (II.11) et l'énergie potentielle par (II.12).

$$Uc = \frac{1}{2}\dot{q}^T H\dot{q} \quad \text{.....(II.11)}$$

$$Up = \sum_{i=1}^{n} mig^T r_{0,ci} \quad \text{.....(II.12)}$$

II.2.3.2. FORMULATION NEWTON EULER :

La formulation de NEWTON EULER sont des équations sous forme récurrente (II.13) ce qui rend la formulation plus adaptée pour construire le modèle dynamique inverse.

$$\begin{cases} f_{i-1,i} - f_{i,i+1} + m_i g - m_i \dot{v}_{ci} = 0 \\ N_{i-1,i} - N_{i,i+1} + r_{i,ci} \times f_{i,i+1} - r_{i-1,ci} \times f_{i-1,i} - I_i \dot{\omega}_i - \omega_i \times (I_i \omega_i) = 0 \end{cases} \quad i = \overline{1,n} \text{ .....(II.13)}$$

Les équations de la dynamique sont coûteuses en temps de calcul (équations différentielles du second ordre) et délicates au niveau de la précision. L'idéal est de calculer le équations dynamiques inverses en temps réel et sans erreurs.

De nombreuses méthodes récursives sont proposées pour réduire le temps de calcul, modèle récurrent de modélisation dynamique des bras manipulateurs (21).

II.2.4. COMMANDE DES ROBOTS ELECTRIQUES : (3) (4) (17)

Tout les robots industriels sont à l'heure de la motorisation électrique, les actionneurs les plus fréquents utilisent des moteurs électriques à courant continu à aimant permanent, situés au niveau de chaque articulation.

L'articulation motorisée comporte un capteur de position souvent des capteurs optiques, et parfois en plus un capteur de vitesse qui est une génératrice tachymétrique disposés sur le moteur.

Le schéma classique d'une boucle d'asservissement de position d'un axe est donné par la figure II.2.

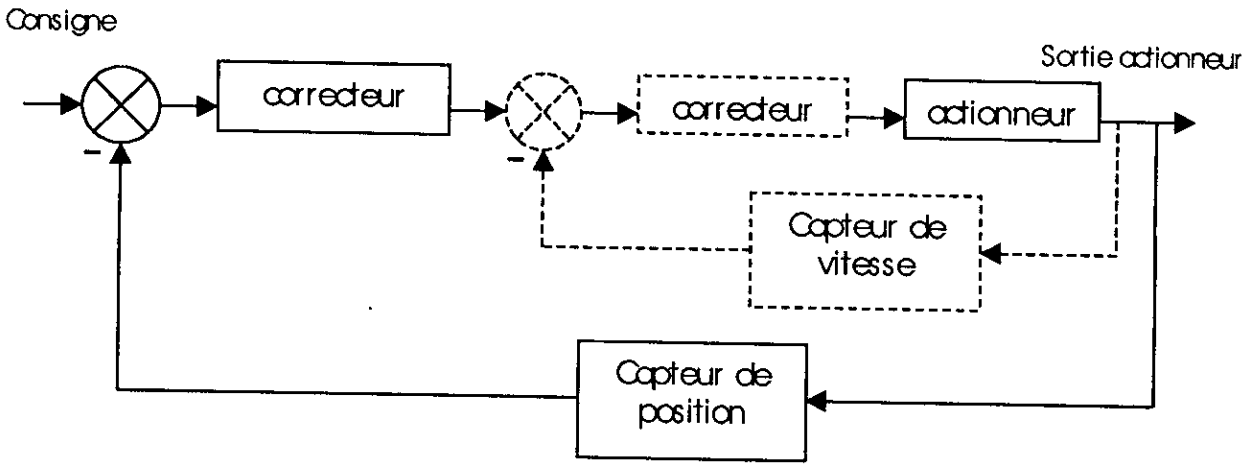


Figure II.2 : Schéma d'asservissement d'un axe de robot.

Les correcteurs sont en général des contrôleurs PID, qui n'exigent pas la connaissance d'un quelconque modèle, il sont localement stables et assurent une régulation correcte.

ϖ **Génération du mouvement point à point :**

Le mode dit *point à point* transfert le robot manipulateur d'une configuration géométrique à une autre sans se préoccuper de la trajectoire suivie par l'effecteur.

Ce mode permet tout simplement de définir une succession de situations spatiales notées j de l'effecteur généralement assez éloignées les unes des autres.

Cette situation j est caractérisée par la coordonnée opérationnelle $x(j)$ de dimension 6×1, et qui constitue un « point » de l'espace de travail du robot manipulateur d'ou la nomenclature « point à point », deux situations sont envisageable dans ce mode de commande :

ϖ Lorsque la vitesse au point j est nulle c'est à dire $\dot{x}(j)=0$, ce point est dit *point d'arrêt.*

ϖ Lorsque la vitesse au point j est non nulle c'est à dire $\dot{x}(j)\neq 0$, ce point est dit *point de passage.*

Ce mode de commande est simple mais très efficace, il est disponible sur tous les robots industriels, convient bien pour les taches de transfert de pièces. Vu du coté effecteur il s'agit de donner la position désirée qui se traduit par les coordonnées opérationnelles $x(j)$, et déterminer les coordonnées généralisées $q(j)=[q_1(j),q_2(j),q_3(j),q_4(j),q_5(j),q_6(j)]^T$ à partir du modèle géométrique inverse suivant l'expression (II.14).

$$q(j) = MGI[x(j)] \quad \text{.....(II.14)}$$

tel que MGI est le modèle géométrique inverse du robot.

Le déplacement imposé par la consigne sera réalisé suivant une certaine loi de vitesse (appelée également profil de vitesse) de l'actionneur pour chaque asservissement de chaque articulation.

On suppose que le contrôleur PID confère à l'asservissement un comportement quasi idéal, en d'autres termes la sortie en vitesse de l'actionneur est pratiquement égale à la consigne délivrée par le générateur de consigne de vitesse c'est à dire le PID.

Dans ces conditions il est donc possible de déterminer les profiles de vitesse en respectant les contraintes de limitation de vitesse et accélération de chaque actionneur suivantes :

$$\begin{cases} |\dot{q}_i| \le \overline{V_i} \\ |\ddot{q}_i| \le \overline{a_i} \end{cases} \quad \text{.....(II.15)}$$

tel que :

$\overline{V_i}$: vitesse maximale de l'actionneur i.

$\overline{a_i}$: accélération maximale de l'actionneur i.

Pour un actionneur électrique la limitation en vitesse provient de la limitation en vitesse du moteur compte tenu du rapport de réduction de la transmission.

Le profil de vitesse est le même pour toutes les articulations (actionneurs) d'un robot manipulateur, pour cette raison nous considérons un seul déplacement articulaire.

Selon la course articulaire à effectuer le profil de vitesse aux capacités maximales de l'actionneur définies par (II.15), affecte la formes trapézoïdale de la figure (II.3).

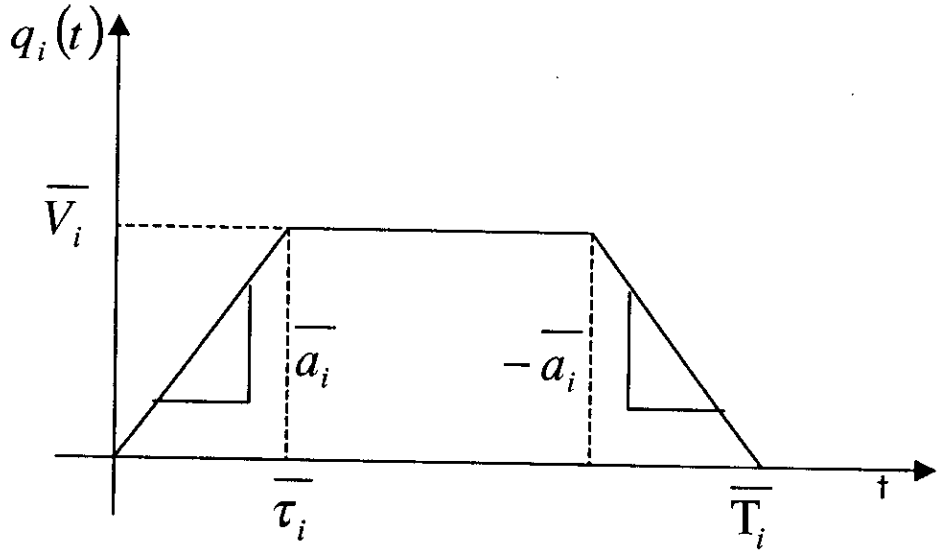


Figure II.3 : profile de vitesse trapézoïdal de l'actionneur i.

La durée d'accélération $\overline{\tau_i}$ est égale à la durée de décélération, le temps de transfert est le plus petit possible, il est noté $\overline{T_i}$.

$$\Delta q_i = \int_0^{\overline{T_i}} \dot{q}(t)\,dt \quad \text{.....(II.16)}$$

l'expression (II.16) représente l'aire de la courbe $\dot{q}(t)$ (figure II.3), il est donc facile d'établir l'expression du temps de transfert :

$$\overline{T_i} = \frac{\Delta q_i}{\overline{V_i}} + \frac{\overline{V_i}}{\overline{a_i}} \quad \text{.....(II.17)}$$

Si l'on souhaite effectuer le transfert en un temps plus long, il faut alors réduire le palier de vitesse qui deviendra V_i et l'accélération qui sera alors notée a_i et le temps de transfert T_i.

ϖ **Lissage du profil de vitesse :**

Le profil de vitesse en trapèze de la figure (II.3) présente des discontinuités de l'accélération articulaire, celle ci peut être responsable d'à coût dans le mouvement, pour éviter cela certains générateurs de consigne de vitesse assure la continuité des accélérations. Les points anguleux sont remplacés par des « raccordements » paraboliques, pour lesquels la dérivée de l'accélération est une constante $\overline{c_i}$.

Le profil de vitesse $q_i(t)$ et le profil d'accélération $q_i(t)$ obtenus sont présentés sur la figure (II.4).

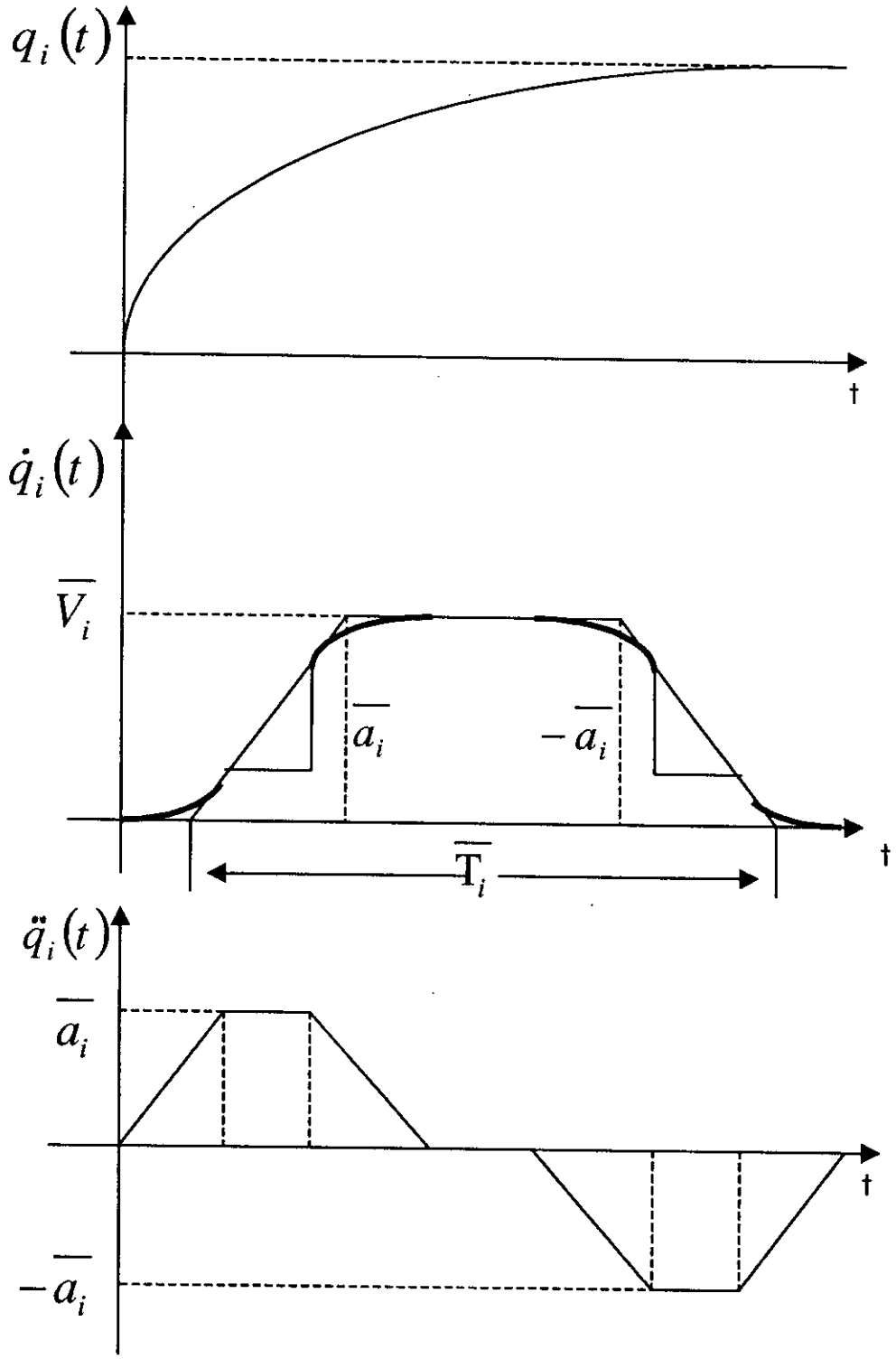


Figure II.4 : Profile de vitesse lissé et accélération de l'actionneur i.

II.3. MODELISATION DU ROBOT AID :

II.3.1. MODELE GEOMETRIQUE DIRECT :

En se basant sur la notation de DENAVIT HARTENBERGER citée plus haut, et en se basant sur le calcul matriciel présenté en annexe B, les matrices de passages du robot AID sont données par :

$$H_1^0(q_1)=\begin{bmatrix} C1 & S1 & 0 & 0 \\ -S1 & C1 & 0 & 0 \\ 0 & 0 & 1 & d1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{.....(II.18.a)}$$

$$H_2^1(q_2)=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C2 & S2 & 0 \\ 0 & -S2 & C2 & d2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{.....(II.18.b)}$$

$$H_3^2(q_3)=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C3 & S3 & 0 \\ 0 & -S3 & C3 & d3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{.....(II.18.c)}$$

$$H_4^3(q_4)=\begin{bmatrix} C4 & S4 & 0 & 0 \\ -S4 & C4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{.....(II.18.d)}$$

$$H_5^4(q_5)=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C5 & S5 & 0 \\ 0 & -S5 & C5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{.....(II.18.e)}$$

$$H_6^5(q_6)=\begin{bmatrix} C6 & S6 & 0 & 0 \\ -S6 & C6 & 0 & 0 \\ 0 & 0 & 1 & d4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{.....(II.18.f)}$$

tel que : d1=0,4 m, d2=0,3 m,d3=0,4 m,d4=0,21 m.

Nous allégeons l'écriture des matrices en optant pour la notation suivante :
$Ci=\cos q_i \quad et \quad Si=\sin q_i$

La position et l'orientation du point terminal du robot est déterminé par la multiplication successive des matrices II.18, le produit matriciel nous donne une matrice 4x4 définie par l'expression II.19.

$$H_6^0(q_1,q_2,q_3,q_4,q_5,q_6)=\begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ h_{31} & h_{32} & h_{33} & h_{34} \\ h_{41} & h_{42} & h_{43} & h_{44} \end{bmatrix} \quad \text{.....(II.19)}$$

$$H_6^0(q_1,q_2,q_3,q_4,q_5,q_6)=\begin{bmatrix} R & & P \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Les expressions des composantes $h_{ij} \quad (i=\overline{1,4} \quad j=\overline{1,4})$ sont données en annexe C.

R est la matrice des cosinus directeurs, elle donne l'orientation du repère de la dernière articulation par rapport au repère de base.

$$R=\begin{bmatrix} n_x & t_x & b_x \\ n_y & t_y & b_y \\ n_z & t_z & b_z \end{bmatrix} \quad \text{.....(II.20)}$$

les composantes de cette matrice sont calculées en annexe B.

P est le vecteur position de l'organe terminal par rapport au repère de base.

$$P=\begin{bmatrix} p_x & p_y & p_z \end{bmatrix}^T \quad \text{.....(II.21)}$$

II.3.1.2. MODELE GEOMETRIQUE INVERSE :

Le modèle géométrique inverse donne les coordonnées généralisées en fonction des coordonnées opérationnelles de la manière suivante :
La position du point remarquable est donné par :

$$\begin{cases} p_x = x - b_x \times d4 \\ p_y = y - b_y \times d4 \\ p_z = z - b_z \times d4 \end{cases} \quad \text{.....(II.22.a)}$$

Les expressions des variables articulaires sont :

$$\begin{cases} q_1 = A\tan 2\left(\dfrac{p_y}{p_x}\right) \\ q_2 = A\tan 2\left(\dfrac{A_7}{A_8}\right) \\ q_3 = A\tan 2\left(\dfrac{A_9}{A_{10}}\right) \\ q_4 = A\tan 2\left(\dfrac{A_{11}}{A_{12}}\right) \\ q_5 = A\tan 2\left(\dfrac{A_{13}}{A_{14}}\right) \\ q_6 = A\tan 2\left(\dfrac{A_{15}}{A_{16}}\right) \end{cases} \quad \text{.....(II.22.b)}$$

Tel que les coefficients $A_k \ \left(k = \overline{1,16}\right)$ sont donnés en annexe C.

Ainsi par les équations (II.22.a), (II.22.b) le modèle géométrique inverse est totalement déterminé.

II.4. CONCLUSION :

Nous avons présenté au cours de ce chapitre les différents modèles de comportement d'un robot. Le modèle général est l'assemblage de tous ces sous ensembles, traduit par les modèles géométrique, cinématique et dynamique. Pour chacun de ceux ci nous avons brièvement décrit les problèmes actuels ainsi que les solutions proposées dans la littérature spécialisée.

L'objectif initial de ce travail a été souligné depuis le début du mémoire, pour cela, uniquement le modèle géométrique du robot AID est considéré, nous n'éprouvons pas le besoin de développer les autres modèles puisque c'est l'objectif recherché qui détermine les outils théoriques nécessaires à sa réalisation, le modèle géométrique est très suffisant pour l'application convoitée.

« CHAPITRE III »

Commande du robot AID

III.1. INTRODUCTION :

La machine à commande numérique, les systèmes de manipulation à distance répondaient à des besoins de flexibilisation de la production, de travaux de hautes dextérité sans présence humaine directe.

Ces besoins vont en s'amplifiant et ces machines semblent déjà avoir fait leur temps...

Le langage souvent révèle ces domaines d'avenir qui répondent à la fois à un besoin concret et à l'imaginaire qui résolvait fictivement les problèmes posés : effecteur, poignet compliant, bras articulé, système de vision... robot.

La robotique investit désormais l'industrie, et bientôt la vie quotidienne. Ecarté le spectaculaire, l'impression prévaut cependant qu'il y a une distance considérable entre le possible dans la production concrète de notre décennie et le souhaitable jugé en critère de productivité, sécurité, ergonomie (étude quantitative et qualitative du travail dans l'entreprise, visant à améliorer les conditions de travail et à accroître la productivité).

Le robot est un système qui s'acquitte d'une action. Ce système décide de l'action, gère en permanence l'ensemble des sous systèmes interagissants entre eux.

Les sous systèmes essentiels sont l'homme, la mécanique, le contrôleur, la mémoire, les capteurs. Ils sont eux mêmes composés de parties... le robot est avant tout un système très complexe (6).

III.2. GENERALITES SUR LES SYSTEMES DE PROGRAMMATION POUR LA ROBOTIQUE :

Un robot de manipulation peut par définition exécuter une grande variété de taches sans modifications matérielles majeures. Une telle versatilité repose sur deux propriétés essentielles des robots :

1) ils possèdent des structures mécaniques générales capables de supporter des outils terminaux variés (pince à deux mors parallèles, torche à souder, ventouse aspirante, ...).
2) ils ont la faculté de s'adapter à certaines variations de l'environnement, à condition que ces variations soient mesurables par des capteurs et qu'elles soient prévues au niveau des modèles fournis aux systèmes de commande.

III.2.1. LES METHODES DE PROGRAMMATION : (5) (18) (20)

La génération du programme peut s'effectuer de deux manières différentes selon qu'elle fait ou non intervenir le robot :

III.2.1.1. COMMANDE PAR APPRENTISSAGE :

La commande par apprentissage est dite programmation à posteriori, ce mode de programmation consiste à conduire le robot ou une réplique exacte conformément à la tache à exécuter et à mémoriser l'évolution correspondante des signaux émis par les différents capteurs postés aux niveaux des articulations.
Ce mode de programmation est particulièrement destiné à l'exécution d'une tache à caractère répétitif, il permettra au robot de naviguer de façon autonome, grâce aux enregistrements précédents servant de consignes à un ensemble d'asservissement qui les recopie exactement.
L'apprentissage peut se faire de 3 manières :

1. L'organe terminal du robot est conduit directement par la main de l'opérateur entraînant les différents capteurs suivant le geste ou la série de gestes effectués qu'il aura ensuite à répéter.
2. Le principal inconvénient de la méthode cité en 1 réside dans la gêne qu'éprouve l'opérateur à mouvoir une structure souvent très lourde et rigide surtout dans le cas d'un mouvement complexe ou qui demande une précision accrue, une alternative à cela celle de l'apprentissage maître esclave analogue à celle d'un télémanipulateur.
3. La dernière méthode consiste à piloter le robot directement du PC, cette stratégie d'apprentissage est beaucoup moins contraignante que les deux autres il n y a pas de grands efforts physiques à fournir et n'exige pas de grands moyens matériels.

III.2.1.2. LOGICIEL DE COMMANDE :

Contrairement au mode de programmation précèdent le logiciel de commande ne fait pas intervenir le robot, il est élaboré en tenant compte des points suivants :

1. Les objectifs des taches à effectuer.
2. Les contraintes imposées par le robot et par l'univers de travail.
3. Les capacités du langage de programmation.

III.3.2. NIVEAUX DE PROGRAMMATION DES ROBOTS : (20)

On distingue habituellement quatre niveaux de programmation conceptuels suivants :

III.3.2.1. PROGRAMMATION NIVEAU ACTIONNEUR :

La tache est décrite en terme d'évolution des variables d'action caractérisant le fonctionnement des actionneurs animant les différents degrés de liberté du robot.

III.3.2.2. PROGRAMMATION NIVEAU EFFECTEUR :

A ce niveau la description d'une tache consiste en un ensemble d'instructions définissant les déplacements et les opérations que doit effectuer l'effecteur (organe terminal), par exemple une pince de préhension, un outil de vissage ou de soudage, etc...

Nous devons donc disposer impérativement du modèle géométrique inverse du robot, ainsi que des dispositifs permettant l'échange avec l'univers.

De nombreux langages de programmation niveau effecteur ont été développés pour programmer des robots, parmi ces langages on peut citer :

WAVE, AL, PAL, LM, EMILY, AML, RAIL, VAL, (20), MAL (13), PASRO (5), PILOT (15) la plupart de ces langages ont déjà été utilisés pour programmer une grande variété de taches de manipulation.

En fait ces langages ne sont que des extensions de langages traditionnels comme le BASIC, PASCAL ou ALGOL le C, le FORTRAN etc..

Ils induent des instructions algorithmiques classiques combinés avec des constructions particulières adaptées à la robotique, par exemple : modélisation de l'univers de travail du robot, spécification de ses mouvements, gestion des capteurs ou encore la gestion du système de commande.

III.3.2.3. PROGRAMMATION NIVEAU OBJET :

La tache à ce niveau est décrite en terme des opérations sur les objets, par exemple l'insertion d'un goujon dans un trou, collage de deux pièces, agrafage de deux pièces, etc...

La tache est décrite d'instructions qui définissent les opérations à exécuter indépendamment de l'effecteur du robot et des outillages spécialisés.

III.3.2.4. PROGRAMMATION NIVEAU OBJECTIF :

La tache à ce niveau est décrite en terme de l'objectif à atteindre par exemple l'assemblage de deux pièces.

Le logiciel doit être capable d'engendrer un programme au niveau objet qui sera lui-même transformé par la suite en une série de micro taches exécutées par l'effecteur du robot.

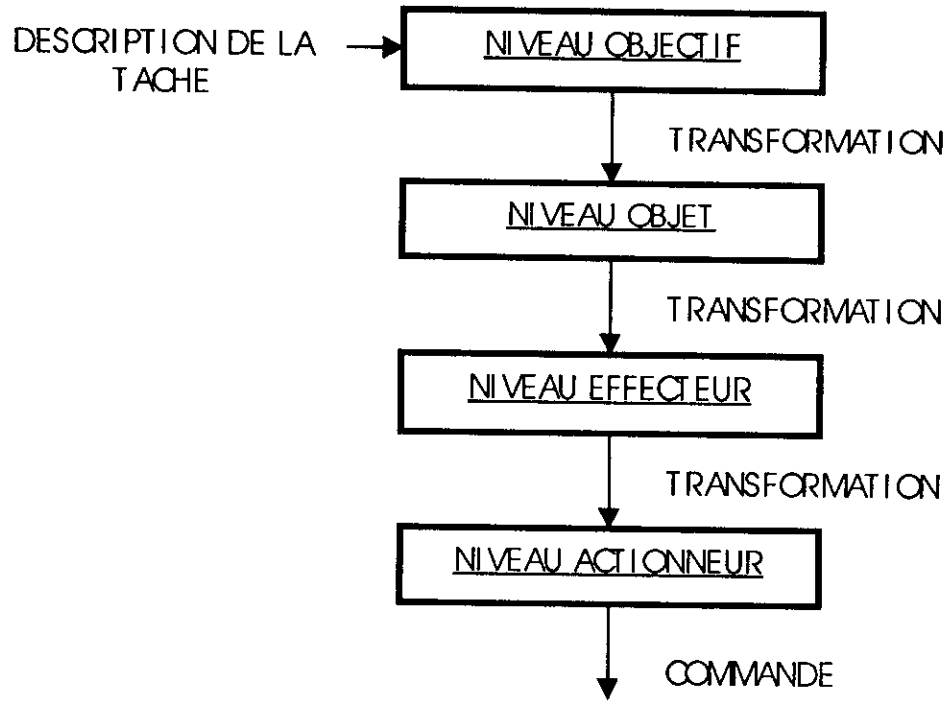


Figure III.1 : Schéma du processus de description des taches (20).

III.3.3. BASES DE LA PROGRAMMATION DES ROBOTS :

La programmation destinée à la robotique obéit à certaines règles de bases qui permettent de ne pas s'éloigner du but recherché, la programmation niveau effecteur étant notre but, nous définirons au cours de cette section toutes les bases se rapportant à ce type de programmation, et par la même occasion adaptées ces bases au outils expérimentaux dont nous disposons point de vue matériel et logiciel (consulter les chapitres précédents).

III.3.3.1. MODELISATION DE L'UNIVERS DE TRAVAIL DU ROBOT : (5) (7) (18)

Le langage de manipulation repose essentiellement sur un mode de représentation de l'univers à base de repères cartésiens. Le principe consiste à modéliser l'outil terminal du robot (modèle géométrique inverse) et les objets de l'environnement (les différents postes de travail) par un ensemble de repères cartésiens judicieusement choisis (figure III.2).

Il est tout aussi nécessaire de prévenir les butées, éviter que le robot n'entre en collision avec lui même et avec son environnement et enfin déterminer les points atteignables par l'organe terminal (effecteur).

Le raisonnement se fait donc en terme de position et d'orientation de ces repères, indépendamment de la forme des objet manipulés et de la structure

méconique du robot. Les types de données intervenant dans cette phase de la programmation sont les vecteurs positions, et les transformations géométriques, avec tout ce qui engendre comme calcul matriciel ou somme, soustraction et produit vectoriel.

Afin d'éviter toute sorte d'ambiguïté nous signalons que le terme *position du robot* désigne non pas les différentes positions des axes mais la position de l'organe terminal, qui est fixé à l'extrémité du bras.

Il est parfois très utile de pouvoir définir une position du robot par rapport à une autre. Ceci s'obtient par une simple addition de deux vecteurs. La figure III.3 montre un vecteur de base auquel on ajoute un vecteur relatif. Le vecteur résultant définit une position relative du robot (par rapport au vecteur de base), la relation qui décrit cette transformation vectoriel est:

Vecteur résultant = vecteur de base + vecteur relatif

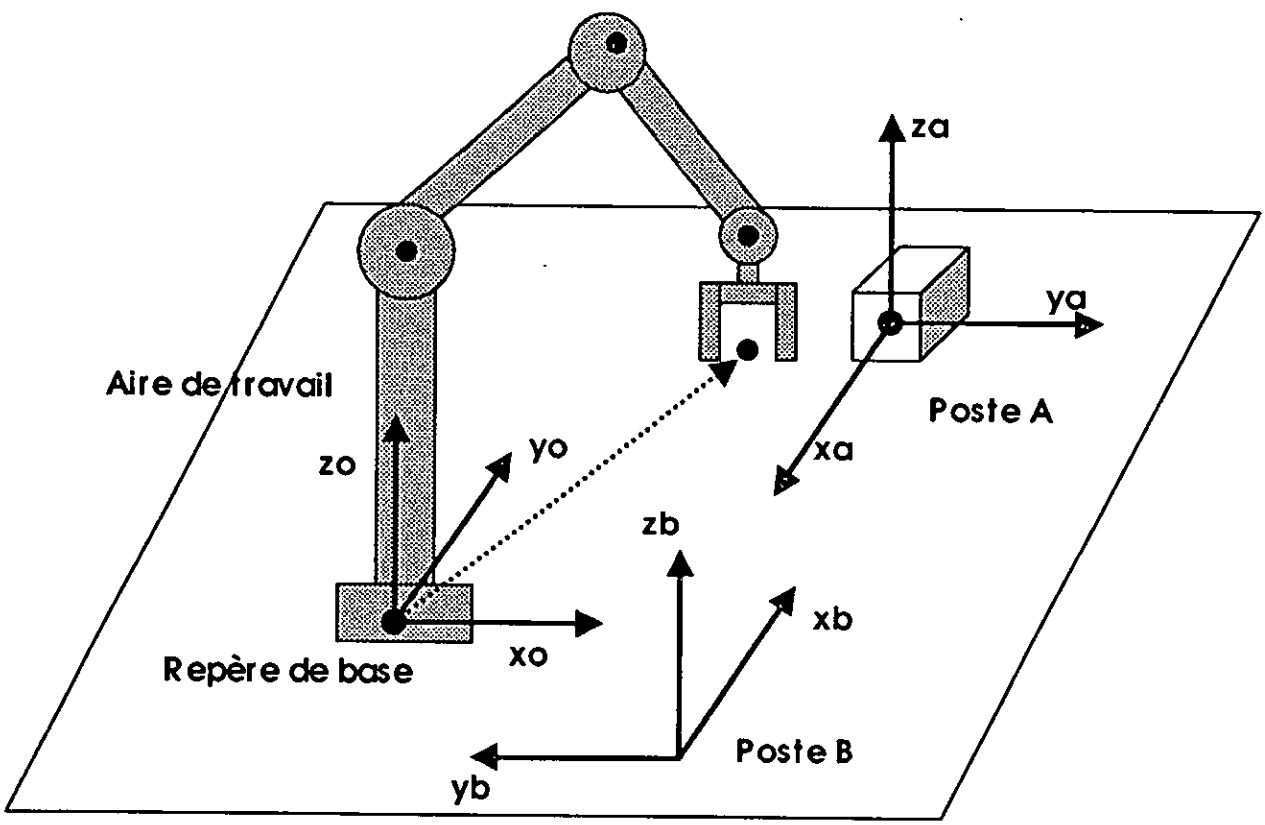


Figure III.2 : définition d'une position relativement au repère de base (5).

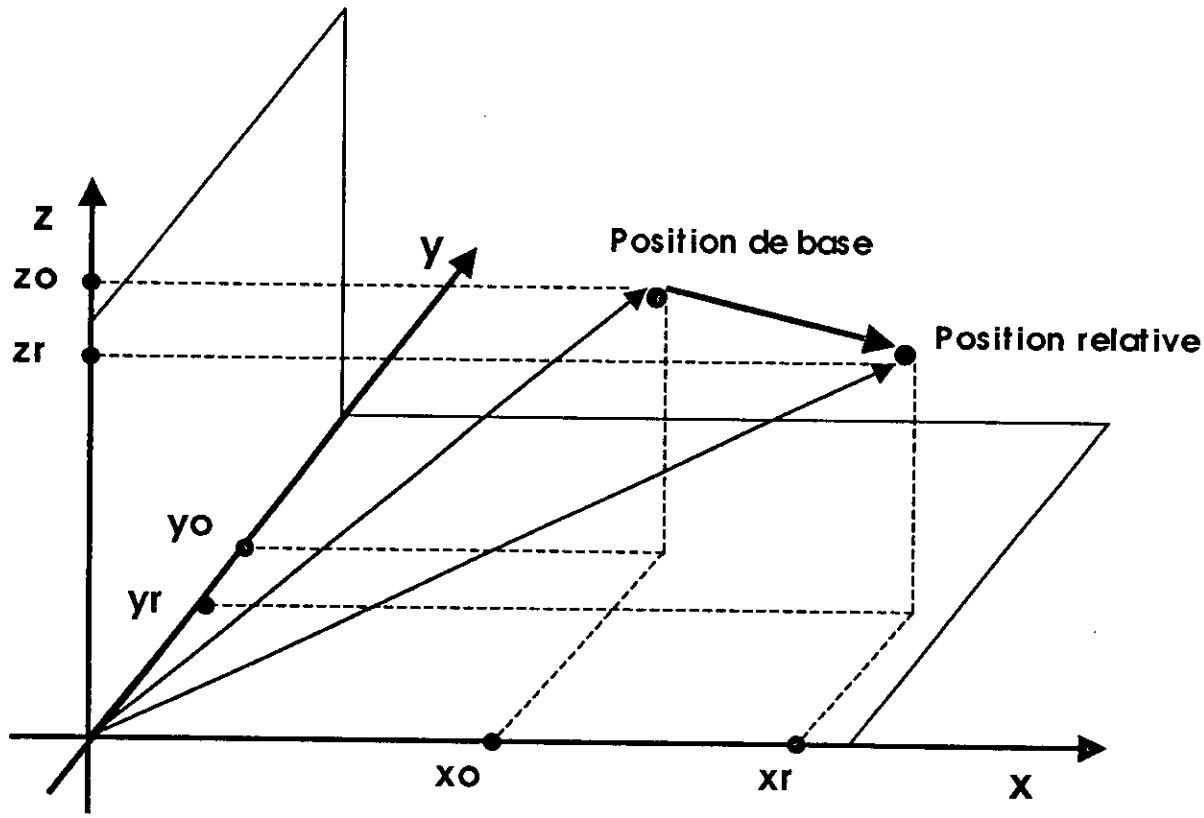


Figure III.3 : définition d'une position relative du robot par addition d'un vecteur (5).

III.3.3.2. ENVIRONNEMENT INFORMATIQUE DU LANGAGE DE MANIPULATION :

Le langage VISUAL BASIC version 4.0 sert comme interface de programmation entre le PC et la carte DSP flex Motion.

Il offre 5 principaux avantages au programmeur :

1. Une syntaxe clairement définie.
2. Une structure de donnée cohérente et très flexible.
3. Aspect interactif qui permet d'agir sur les programmes en temps réel.
4. Compilateur très rapide et puissant.
5. Aspect visuel permet de grande possibilités de visualisation graphiques et animation en temps réel.

Le VISUAL BASIC 4.0 offre des possibilités de programmation sans limites, pour plus d'informations consulter (14) (11) (19) ainsi que le CD d'accompagnement.

III.4. APPLICATION AU ROBOT AID :

Afin de matérialiser tous les concepts théoriques développés à ce stade du mémoire, et afin d'exploiter les performances du robot AID en combinaison avec celles

de la carte DSP Flex Motion, nous proposons dans cette partie réalisation de faire un programme de commande niveau effecteur.

III.4.1. DESCRIPTION DU CAHIER DES CHARGES :

Il s'agit de déplacer des pièces d'un poste A à un poste B, dans un mouvement cyclique, puisque les robots industriels sont généralement destinés à effectuer des taches longues et répétitives.

III.4.2. ORGANISATION LOGICIELLE DU PROGRAMME DE COMMANDE :

Le programme de commande du robot AID adopte un principe de structure hiérarchisée, dans laquelle chaque sous programme doit être écrit de façon à ce qu'une modification dans le corps d'un sous programme donné, ou procédure n'affecte pas le reste du programme.

Une procédure ou une fonction se présente comme un petit programme utilisé dans le contexte d'un autre, peuvent être appelées à tout moment du programme principal.

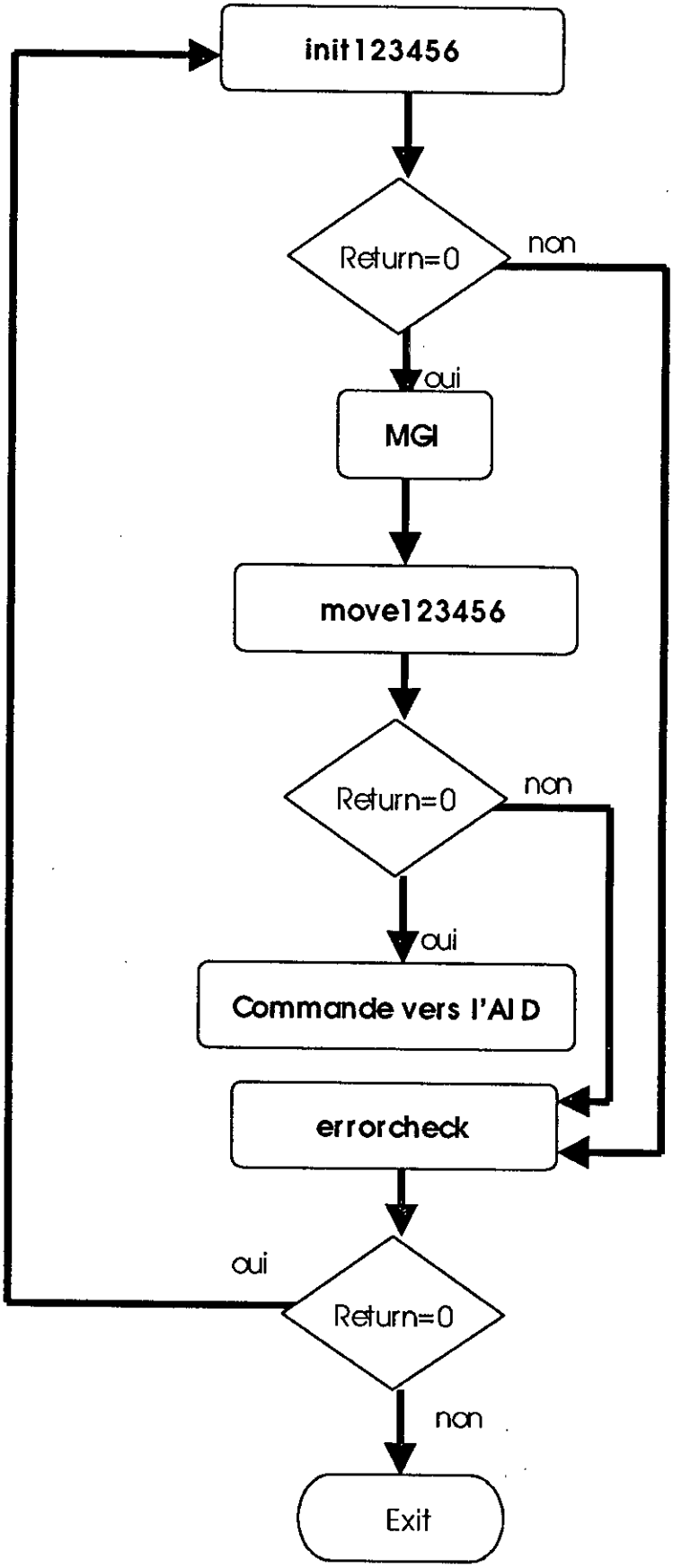


Figure III.4 : Organigramme du programme de commande du robot AID.

Le programme principal de commande du robot AID se compose de 3 procédures essentielles :

1) **Procédure d'initialisation** : représentée par le schéma de la figure III.5.a.
2) **Procédure de mouvement** : représentée par le schéma de la figure III.5.b.
3) **Procédure de gestion des erreurs modales** : représentée par le schéma de la figure III.5.c.

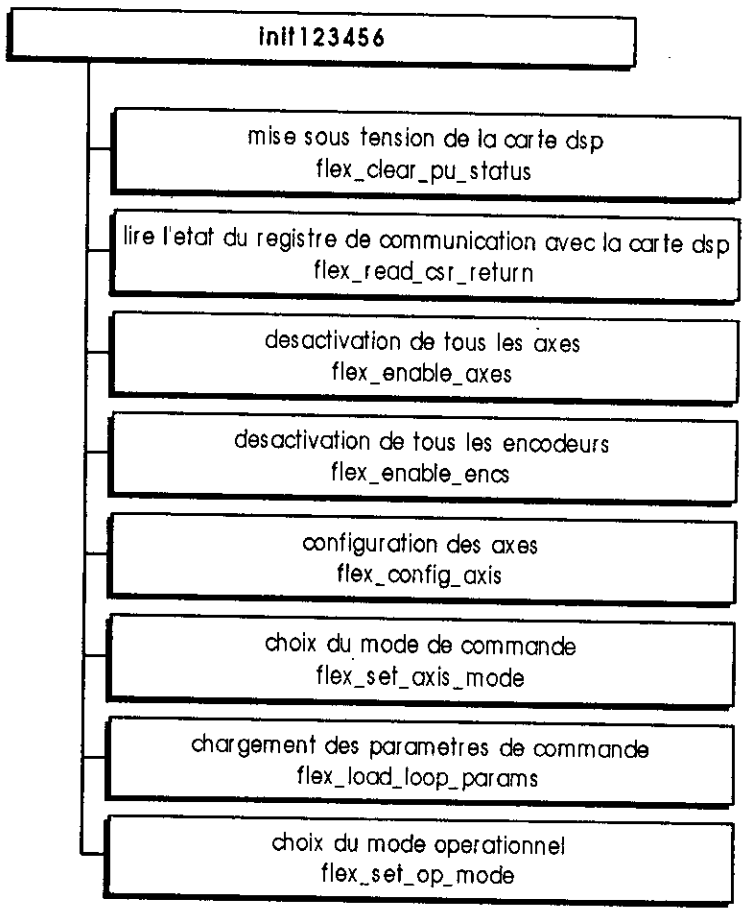


Figure III.5.a: procédure d'initialisation.

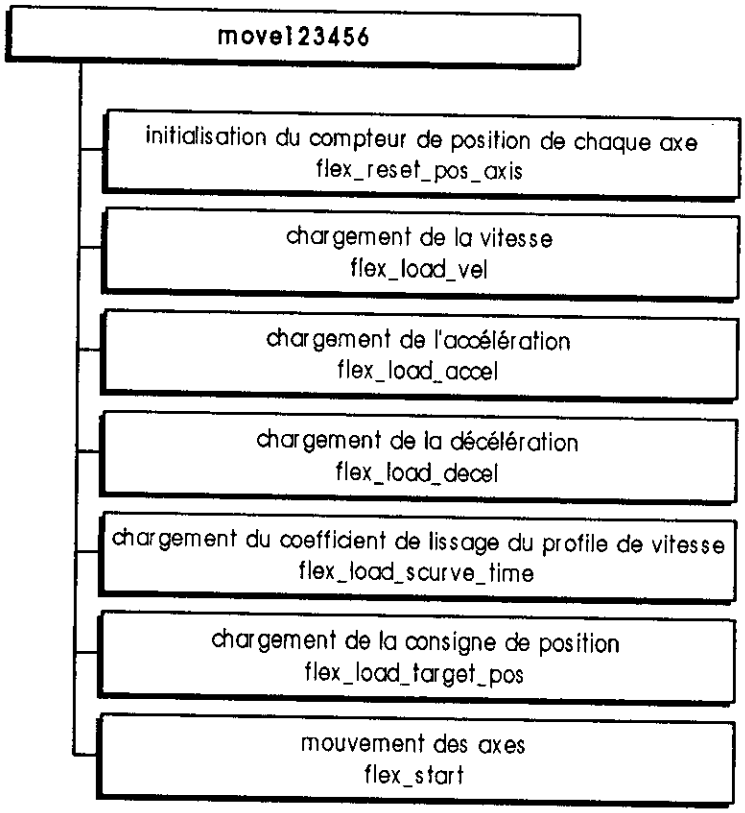


Figure III.5.b : Procédure de mouvement.

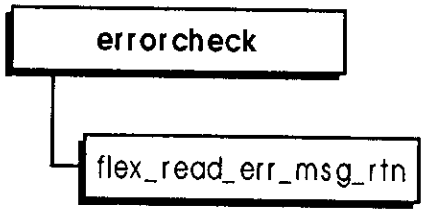


Figure III.5.c : Procédure de gestion des erreurs modèles.

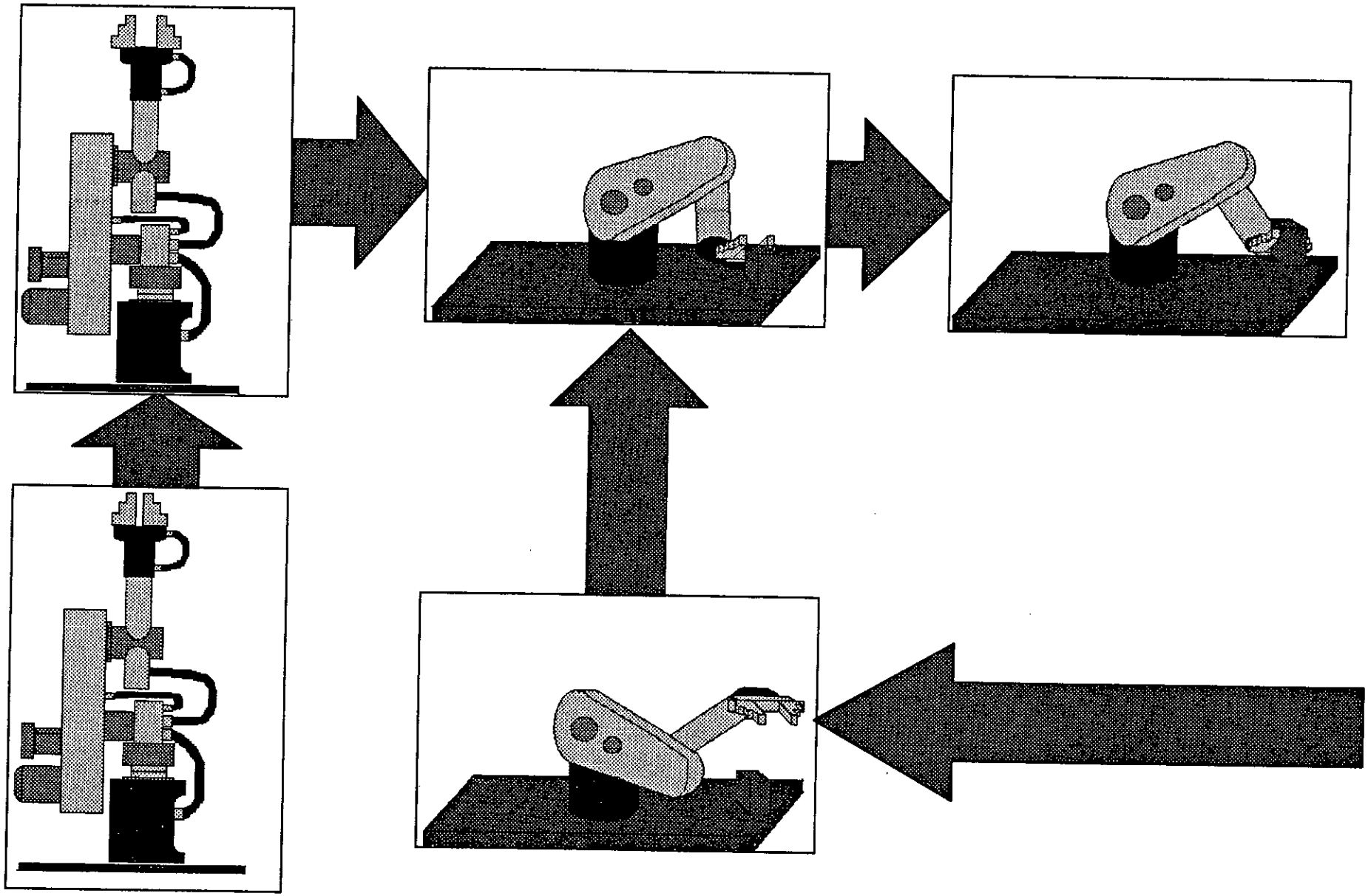
III.5. CONCLUSION :

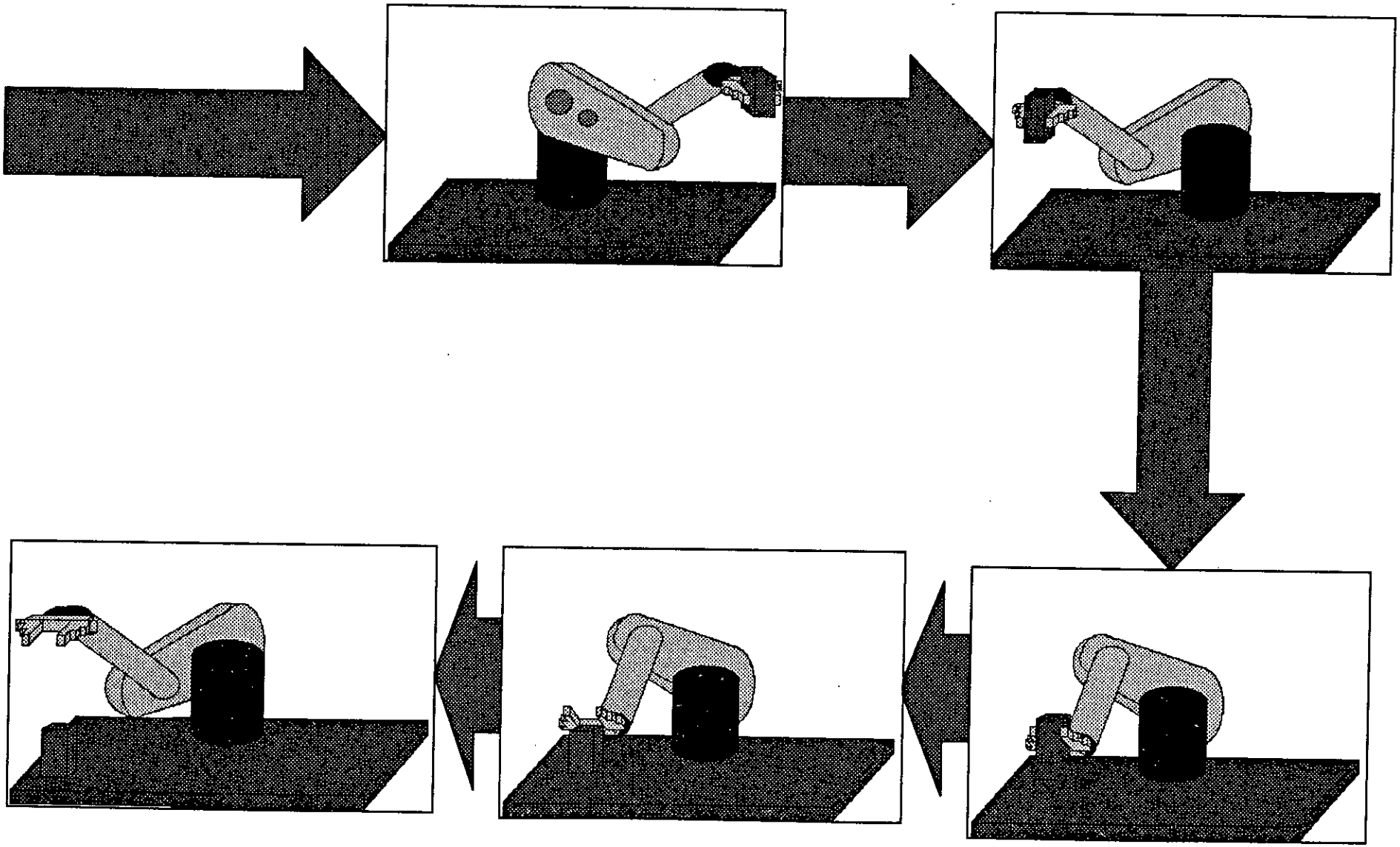
L'informatique constitue un élément essentiel à la conduite des robots industriels. Elle permet en particulier d'obtenir une meilleure maîtrise des mécanismes d'action et de perception, en offrant la possibilité d'accéder plus efficacement aux fonctions de base d'un robot.

La structure informatique utilisée est basée sur le concept « robot programmable », car elle s'appuie sur la définition d'un langage permettant de décrire symboliquement l'enchaînement des actions qui sont nécessaires à l'exécution d'une tache de manipulation ordonnée par l'utilisateur.

la construction d'un programme de commande de robot est un travail délicat qui nécessite une bonne maîtrise de l'univers physique.

Le programmeur doit en effet déterminer quantitativement les actions à exécuter ainsi que les données sensorielles à prendre en compte. Il doit également prévoir les incidents (collisions en particulier), et spécifier les interactions qui lieront, au cours de l'exécution du programme, certaines actions du robot à des données externes (données transmises par des capteurs , signaux produit par d'autres robot ou d'autres appareils..).





CONCLUSION GENERALE

CONCLUSION GENERALE :

Au terme de ce travail, nous exposons les problèmes que nous avons rencontré lors de sa réalisation en laboratoire.

Effectivement, comme pour toute réalisation nous nous sommes heurté à quelques problèmes d'ordre pratique, ces problèmes nous ont stoppés pendant plusieurs semaines.

Le problème en question se posait au niveau des axes du robot, l'exécution du programme de commande introduit sur le PC ne faisait réagir que le premier axe, alors que les cinq autres restaient immobiles, aucun message d'erreur ne nous parvenait de la carte DSP Flex Motion.

En procédant par élimination, nous avons pu déterminer l'origine de ce problème qui était en fait d'ordre logiciel, il a fallu donc réinstaller une réactualisation des drivers de la carte DSP Flex Motion, ces drivers nous sont parvenus il y a exactement deux semaines.

Nous avons pu par la suite faire la programmation puisque la carte fonctionne actuellement de façon normale.

Nous avons pu constater grâce à ce travail les difficultés de la commande par la programmation, même si la carte DSP Flex Motion possède des performances très élevées, nous sommes incités en permanence à réfléchir sur des problèmes d'ordre mécanique, informatique et automatique.

La robotique est donc le cadre idéal d'application des sciences de l'ingénieur, puisqu'elle fait intervenir plusieurs domaines à la fois.

Une vaste palette de perspectives s'offrent quant à la poursuite du travail effectué jusqu'à lors. Nous pouvons par exemple citer la commande du robot par apprentissage, doter le dispositif actuel d'instrumentation nouvelle en d'autres termes rajouter des capteurs de vision (camera), des capteurs de distance (capteurs à ultrasons), des capteurs tachymétrique pour que le robot soit autonome d'avantage.

Faire de la programmation niveau objet et objectif, introduire d'autres algorithmes de commande (commande adaptative par exemple) etc...

Etat de la recherche en robotique aux Etats Unis et au japon :

Aujourd'hui, il n'est pas d'argumentation dans le monde des robots industriels sans référence aux Etats Unis et au Japon qui investissent beaucoup dans ce domaine.

En effet, en 1990 on estime à 250 000 robots installés aux Etats Unis. Les décisions impulsant le domaine de la robotique en découlent.

En premier lieu, il y a nécessité d'accélérer les transfert de savoir entre les laboratoires éminents et l'industrie, actuellement toutes les grandes université américaines possèdent des unités de recherche en matière de robotique.

Au japon, c'est l'industrie qui entreprend la part essentielle de la recherche, sur le plan financier en particulier. En conséquent l'arrivée des robots à leur poste de travail dans les usines est beaucoup plus rapide.

ANNEXES

GROUPES DE COMMANDES DE LA CARTE DSP FLEX MOTION

La DSP Flex Motion possède une grande variété de commandes, elles peuvent être reparties en 15 groupes suivants (9):

Groupe de commandes 1 : « Maintenance de la carte DSP Flex Motion ».

Ce groupe comporte des commandes agissant directement sur l'état de la carte DSP Flex Motion.

Syntaxe de la commande	Action de la commande
Flex_verifyaddr	Vérification de l'adresse de la carte
Flex_clear_pu_status	Mise à zéro du bit indicateur de la mise sous tension de la carte
Flex_reset_bd	Initidisation de la carte
Flex_save_defaults	Sauvegarder toutes les configurations et paramétrages effectués par défaut
Flex_self_test_rtn	Test la carte
Flex_load_fpga	Charger un nouveau programme FPGA
Flex_load_DSP	Charger nouveau programme dans la DSP
Flex_extract_board_type	Déterminer l'adresse de la carte
Flex_read_addr_arrays	Lecture de l'adresse de la carte
Flex_write_addr_arrays	Ecriture de l'adresse de la carte

Tableau A. 1

Groupe de commande 2 : « Communication de bas niveau »

Syntaxe de la commande	Action de la commande
Flex_read_csr_rtn	Lire le registre de communication du PC vers la carte
Flex_communicate	Communication avec la carte DSP
Flex_flush_rdb	Vide le RDB (Return Data Buffer) par une lecture répétitive de celui ci
Flex_read_err_msg_rtn	Lecture des messages d'erreurs

Tableau A. 2

Le tableau A.2 comprend entre autre la commande permettant la lecture des messages d'erreurs provenant du programme de gestion des erreurs (**flex_read_err_msg**), ces erreurs proviennent pour plusieurs raisons parmi elles :

- Utilisation erronée d'une des commandes de la Flex Motion.
- Commande non valide pour un mode utilisé.
- Donnée non existante ou non déclarée ou chargée.
- Spécification erronée des bits opérationnels ou mauvaise assignation, etc..

Toutes les erreurs ne sont pas lues en même temps, le buffer étant de type LIFO (Last In First Out), l'erreur la plus récente est immédiatement disponible.

Groupe de commande 3 : « Organiser la configuration d'interface »

Les commandes disponibles au niveau de ce groupe servent principalement à la configuration, le masquage et la lecture des interruptions.

Syntaxe de la commande	Action de la commande
Flex_config_irq	configurer le nombre d'interruptions
Flex_set_irq_mask	Masquage des interruptions
Flex_read_irq_status	Lecture de l'état de la carte en interruption

Tableau A. 3

Groupe de commande 4 : « Configuration de ressource »

Ce groupe de commande comporte toutes les commandes permettant la configuration des axes pendant l'étape d'initialisation plus communément appelée « **axis resource mapping** ».

Ces commandes permettent entre autre de :

- Définir les procédures de retour (feedback) et de sortie (output) pour chaque axe (**flex_config_axis**).

Chaque axe peut lui voir assigner deux procédures de retour et deux procédures de sortie (dans le cas d'une régulation de position et de vitesse).

Les procédures de retour sont les encodeurs (6) et les convertisseurs analogiques numériques ADC (8), les procédures de sortie sont les convertisseurs numériques analogiques DAC (6) ainsi que les générateurs d'impulsion.

- Attribuer le mode de contrôle pour chaque axe (**flex_set_axis_mode**), il en existe deux , le mode SERVO (mode automatique en boucle fermée) et le mode STEPPER (moteur pas à pas en boucle ouverte ou en boucle fermée).
- Choisir dans un premier temps les axes destinés au mouvement, et dans un second temps attribuer la période d'échantillonnage à chaque axe à raison de 62µs par axe (**flex_enable_axes**).
- Activer ou désactiver les six encodeurs disponibles au niveau de la carte Flex Motion (**flex_enable_encs**).
- Configurer les vecteurs espace, qui peuvent contenir entre 2 et 3 axes destinés à effectuer un mouvement coordonné dans l'espace (**flex_config_vect_spc**).
- Configurer les axes ressources avec une vitesse limitée (couple limité) en limitant l'intervalle de sortie des DAC (**flex_load_torque_offset**).

Etc.

Syntaxe de la commande	Action de la commande
Flex_config_axis	Configuration des axes du robot
Flex_set_axis_mode	Assignation du mode de commande
Flex_enable_axes	Activation ou désactivation des axes pour le mouvement
Flex_enable_encs	Activation ou désactivation des encodeurs
Flex_config_mc_criteria	Configuration du critère de mouvement mc (move complete)

Flex_config_vect_spc	Configuration des vecteurs espace
Flex_load_torque_lim	Chargement du couple limite en limitant la tension à la sortie des DAC
Flex_load_torque_offset	Chargement des limitation de tension de sortie au niveau des DAC.
Flex_load_vel_tc_rs	Chargement de la constante de temps du filtre digital à pole unique passe bas.
Flex_load_counts_rev	Chargement de la valeur de retour par unite de mesure des encodeurs (encodercounts).
Flex_load_steps_rev	Chargement de la valeur de retour par unite de mesure des DAC (dacvalue).
Flex_config_step_mode_pol	Configure le mode et la polarite.
Flex_config_axis_array	Configuration des axes

Tableau A. 4

Groupe de commande 5 : « Configuration de paramètres de contrôle »

La DSP permet un fonctionnement en servo (PID, PIVFF), donc il est évident de charger les différentes valeurs des paramètres de contrôle, le groupe de commande 5 permet de charger ces valeurs individuellement, ou bien les charger tous en même temps, il existe huit paramètres de contrôle :

1. Gain proportionnel **Kp**.
2. Gain intégral **Ki**.
3. Limite d'intégration **ILIMIT**.
4. Gain dérivé **Kd**.
5. Période d'échantillonnage de l'action dérivée **Td**.
6. Gain de retour en vitesse **Kv**.
7. Gain d'anticipation en vitesse **Vff** (velocity feedforward gain).
8. Gain d'anticipation en accélération **Aff** (acceleration feedforward gain).

Syntaxe de la commande	Action de la commande
Flex_load_loop_params	Chargement de tout les paramètres de contrôle
Flex_load_kp	Chargement du gain proportionnel
Flex_load_ki	Chargement du gain intégral
Flex_load_ilim	Chargement de la limite d'intégration
Flex_load_kd	Chargement du gain dérivé
Flex_load_td	Chargement de la période d'échantillonnage de l'action dérivée
Flex_load_kv	Chargement du gain en vitesse
Flex_load_aff	Chargement du gain d'anticipation en accélération
Flex_load_vff	Chargement du gain d'anticipation en vitesse
Flex_load_notch_params	Chargement d'un filtre

Tableau A. 5

Groupe de commande 6 : « Contrôle de trajectoire »

Comprend toutes les commandes permettant :

- L'initialisation du compteur de position de tous les axes (**flex_reset_pos**).
- Attribution du mode opérationnel des axes (**flex_set_op_mode**), il en existe cinq :
  1. Mode de position absolue.
  2. Mode de position relative
  3. Mode de vitesse.
  4. Mode relatif à la position mesurée
  5. Mode de module de position.

- Charger le profile de vitesse désiré pour chaque actionneur, en d'autres termes charger la vitesse de rotation, l'accélération et la décélération pour chaque actionneur.

- Charger les paramètres d'une trajectoire déjà près programmée sur la DSP on en dénombre les trois suivantes :

1. Arc circulaire s'effectue dans un plan à 2 dimensions (**flex_load_2d_arc**).
2. Arc sphérique s'effectue dans un plan à 3 dimensions (**flex_load_3d_arc**).
3. Arc hélicoïdal s'effectue dans un plan à 3 dimensions (**flex_load_helix**).

Syntaxe de la commande	Action de la commande
Flex_reset_pos	Initialisation du compteur de position pour chaque axe
Flex_set_op_mode	Assignation du mode opérationnel choisi
Flex_load_target_pos	Chargement de la consigne de position
Flex_load_vel	Chargement de la vitesse de rotation des actionneurs
Flex_load_rpm	Chargement de la valeur de la vitesse de rotation des actionneurs en révolution par minute
Flex_load_accel	Chargement de la valeur de l'accélération des actionneurs
Flex_load_decel	Chargement de la valeur de la décélération des actionneurs
Flex_load_accel_rpsps	Chargement de la valeur de l'accélération des actionneurs en révolution par seconde par seconde (rev/s^2)
Flex_load_decel_rpsps	Chargement de la valeur de la décélération des actionneurs en révolution par seconde par seconde (rev/s^2)
Flex_load_scurve_time	Chargement de la valeur du coefficient de lissage du profile de vitesse
Flex_load_follow_err	Chargement de l'erreur admise pendant la régulation par mesure de sécurité
Flex_load_blend_fact	Charger le facteur de temporisation
Flex_load_pos_modulus	Chargement de la position lorsque le mode modulus est choisi

Flex_load_2d_arc	Chargement des paramètres d'une trajectoire circulaire
Flex_load_3d_arc	Chargement des paramètres d'une trajectoire sphérique
Flex_load_helix	Chargement des paramètres d'une trajectoire en hélice
Flex__set_point_mode	Assignation du mode point à point à un axe ou un vecteur espace
Flex_load_base_vel	Chargement de la vitesse de base pour un moteur pas à pas
Flex_load_accel_fact	Chargement du facteur accélération pour un moteur pas à pas

Tableau A. 6

Groupe de commande 7 : « début et fin de mouvement »

Comprend les commandes du début (**flex_start**) et fin de mouvement (**flex_stop**), leur principal avantage est de pouvoir bouger ou arrêter les six axes en même temps et cela en attaquant directement le vecteur contrôle des axes.

Syntaxe de la commande	Action de la commande
Flex_start	Mouvement synchronisé des axes ou des vecteurs espace
Flex_blend	Arrêt d'un mouvement pendant une durée du facteur de temporisation charger préalablement
Flex_stop	Arrêt synchronisé des axes ou des vecteurs espace
Flex_halt Flex_kill	Indique quel axe ou vecteur espace peut s'arrêter individuellement ou simultanément

Tableau A. 7

Groupe de commande 8 :

Syntaxe de la commande	Action de la commande
Flex_config_gear_master	Configuration d'un axe ou encodeur comme maitre, pour un fonctionnement nemitre esclave
Flex_load_gear_ratio	Charge le rapport d'effort entre l'axe esclave te le maitre, ce rapport peut etre relatif ou absolu
Flex_enable_gear	Activation ou desactivation des axes esclaves

Tableau A. 8

Groupe de commande 9 : « Restitution ou lecture des données de trajectoire »

Comprend toutes les commandes permettant la lecture ou la restitution des valeurs numériques de la position ou vitesse des axes, ainsi que la position des encodeurs et les DAC, et cela avant pendant et après le mouvement du robot, ces valeurs peuvent être interprétées sans l'aide du manuel de commandes de la Flex Motion.

Syntaxe de la commande	Action de la commande
Flex_read_pos Flex_read_pos_rtn	Lecture de la position actuelle des axes
Flex_read_vel Flex_read_vel_rtn	Lecture de la vitesse
Flex_read_rpm_rtn	Renvoi le lecture de la vitesse
Flex_read_encoder Flex_read_encoder_rtn	Lecture des encodeurs
Flex_read_raw_vel Flex_read_raw_vel_rtn	Lecture de la vitesse non filtrée
Flex_read_follow_err Flex_read_follow_err_rtn	Lecture de l'erreur
Flex_read_target_pos	Lecture de la consigne en position

Flex_read_target_pos_rtn	
Flex_acq_traj_data	Acquisition des valeurs de trajectoire
Flex_read_traj_data Flex_read_traj_data_rtn	Lecture des valeurs de trajectoire
Flex_read_dac Flex_read_dac_rtn	Lecture des DAC
Flex_read_steps_gen Flex_read_steps_gen_rtn	Lecture des impulsions générées par un moteur pas à pas

Tableau A. 9

Groupe de commande 10 : « Restitution ou lecture de l'état de trajectoire »

Comprend toutes les commandes permettant la lecture ou la restitution des états des axes (actionneurs), et cela avant pendant et après le mouvement du robot, ces valeurs peuvent être interprétées avec l'aide du manuel de commande de la Flex Motion.

Syntaxe des commandes	Action de la commande
Flex_read_axis_status Flex_read_axis_status_rtn	Lecture de l'état de l'axe
Flex_read_rs_status Flex_read_rs_status_rtn	Lecture de l'état de marche ou d'arrêt des axes ou des vecteurs espace
Flex_read_motoff_status Flex_read_motoff_status_rtn	Lecture de l'état de marche ou d'arrêt des moteurs ou des vecteurs espace
Flex_read_mcs_rtn	Lecture de l'état du mouvement complet des axes ou des vecteurs espace
Flex_read_blend_status Flex_read_blend_status_rtn	Lecture de l'état de la temporisation appliquée aux axes ou aux vecteurs espace
Flex_read_array_status Flex_read_array_status_rtn	

Tableau A. 10

Groupe de commande 11 : « contrôle analogique et digital des entrées/sorties »

Syntaxe de la commande	Action de la commande
Flex_set_port_dir	Assignation de direction désirée aux ports 1 et 2 en entrée ou sortie
Flex_set_port_pol	Polarité des ports
Flex_config_io_pin	Configurer les pins entrées/sorties
Flex_read_port Flex_read_port_rtn	Lecture de l'état logique des ports entrées/sorties
Flex_config_clocks	Configurer le compteur/ timer ou PWM
Flex_load_pwm_duty	Charger le période de l'horloge PWM
Flex_load_timer_bp	Charger la période du timer
Flex_read_cap_timer Flex_read_cap_timer_rtn	Lecture du timer
Flex_enable_adcs	Activer ou désactiver les ADC
Flex_read_adc Flex_read_adc_rtn	Lecture des ADC
Flex_load_dac	Charger des valeurs aux DAC

Tableau A. 11

Groupe de commande 12 :

Syntaxe de la commande	Action de la commande
Flex_set_home_pol	Polarité de l'entrée *home* comme étant inversée ou non inversée
Flex_enable_homes	Activation ou désactivation des entrées *home*
Flex_set_lim_pol	Limitation de la polarité
Flex_enable_lims	Activation ou désactivation des entrées
Flex_load_sw_lim_pos	Chargement de la limite logicielle de la

	position de chaque axe
Flex_enable_sw_lim_pos	Chargement d'une décélération lorsque l'axe arrive aux limites de la position imposée
Flex_enable_sw_lims	Activation ou désactivation de la procédure de lecture très précise de la position des axes ou encodeurs
Flex_load_pos_bp	Chargement de la position du breakpoint
Flex_load_bp_modulus	Chargement du breakpoint en mode de module de position
Flex_load_at_bp	Chargement du temps d'anticipation du breakpoint
Flex_enable_bp	Activation ou désactivation du breakpoint chargé préalablement

Tableau A. 12

Groupe de commande 13 :

Syntaxe de la commandes	Action de la commande
Flex_read_home_status Flex_ read_home_status_rtn	Lecture de l'état de toutes les entrées *home*
Flex_read_lim_status Flex_ read_lim_status_rtn	Lecture des états de toutes les entrées limitées
Flex_read_cap_pos Flex_ read_cap_pos_rtn	Lecture des données position à partir de l'axe ou de l'encodeur indiqué
Flex_read_hs_cap_status Flex_ read_hs_cap_status_rtn	Lecture de l'état des données position à partir de l'axe ou de l'encodeur indiqué
Flex_read_pos_bp_status Flex_read_pos_bp_status_rtn	Lecture de l'état de la position *breakpoint* Pour tout les axes ou les encodeurs
Flex_read_at_bp_status Flex_read_at_bp_status_rtn	Lecture de l'état du breakpoint
Flex_read_vel_status Flex_read_vel_status_rtn	Lecture de l'état de la vitesse pour tous les axes

Tableau A. 13

Groupe de commande 14 : « Initialisation du mouvement des axes »

Pour une parfaite identification de la position du robot dans l'espace le programmateur à besoin de se référer à une configuration géométrique initiale, les commandes du groupe 14 permettent une initialisation du robot en détectant les fins de course de chaque axe (**flex_find_home** et **flex_find_index**).

Syntaxe des commandes	Action de la commande
Flex_find_home	Recherche de la direction de la trajectoire qui peut être suivi par un axe et la localisation d'un éventuel switch d'initialisation
Flex_find_index	Recherche pendant la procédure d'initialisation l'index de l'encodeur associé à chaque axe

Tableau A. 14

Groupe de commande 15 : « Opérations sur les variables»

Ce groupe comprend toutes les commandes de manipulation des variables, addition, soustraction, multiplication et division ainsi que les opérations d'affectation et de lecture des variables, etc..

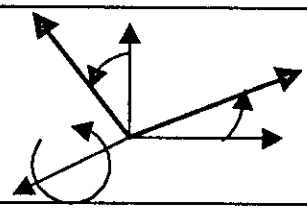
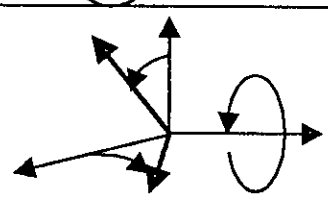
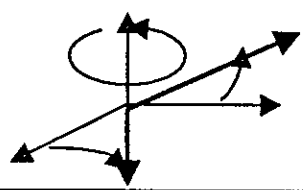
Syntaxe des commandes	Action de la commande
Flex_begin_store	Initialise le début de la sauvegarde d'un programme
Flex_end_store	Fin de la sauvegarde d'un programme
Flex_run_prog	Exécution d'un programme
Flex_pause_prog	Suspension de l'exécution du programme
Flex_stop_prog	Arrêt de l'exécution d'un programme
Flex_insert_label	Marquer une position dans le programme pour des besoins de saut conditionels

Flex_jump_label	Saut conditionnel à la position indiquée préalablement
Flex_set_status_momo	Contrôle du registre d'état
Flex_load_delay	Charger un retard dans un programme en milisecondes
Flex_wait Flex_wait_or	Attendre avec condition
Flex_add_var	Addition de deux variables
Flex_sub_vars	Soustraction de deux variables
Flex_mult_vars	Multiplication de deux variables
Flex_div_vars	Division de deux variables
Flex_load_var	Charger une variable
Flex_read_var	Lecture d'une variable
Flex_and_vars	And logique entre deux variables
Flex_or_vars	Or logique entre deux variables
Flex_xor_vars	Xor logique entre deux variables
Flex_not_var	Negation logique entre deux variables
Flex_save_object	Sauvegarde d'un objet
Flex_delete_object	Effacer un objet
Flex_free_object	Changer les valeurs d'un objet
Flex_load_description Flex_read_description_rtn	Chargement d'une description
Flex_read_registry_rtn	Lecture d'un registre

Tableau A. 15

MATRICES DE PASSAGE ENTRE REPERES VOISINS ET LES ANGLES D'EULER

ϖ Matrices de passage entre repères voisins : (20)

Repères	Matrices de passage	
	Passage de (i-1) vers (i)	Passage de (i) vers (i-1)
	$M_i^{i-1}=\begin{bmatrix} 1 & 0 & 0 \\ 0 & C_i & -S_i \\ 0 & S_i & C_i \end{bmatrix}$	$M_{i-1}^{i}=\begin{bmatrix} 1 & 0 & 0 \\ 0 & C_i & S_i \\ 0 & -S_i & C_i \end{bmatrix}$
	$M_i^{i-1}=\begin{bmatrix} C_i & 0 & S_i \\ 0 & 1 & 0 \\ -S_i & 0 & C_i \end{bmatrix}$	$M_{i-1}^{i}=\begin{bmatrix} C_i & 0 & -S_i \\ 0 & 1 & 0 \\ S_i & 0 & C_i \end{bmatrix}$
	$M_i^{i-1}=\begin{bmatrix} C_i & -S_i & 0 \\ S_i & C_i & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$M_i^{i-1}=\begin{bmatrix} C_i & S_i & 0 \\ -S_i & C_i & 0 \\ 0 & 0 & 1 \end{bmatrix}$

ϖ Angles d'EULER : (2)

Les angles d'EULER sont des variables articulaires indépendantes qui déterminent l'orientation du repère considéré au repère de base uniquement.

Nous définissons la matrice de rotation R qui correspond aux trois rotations consécutives associées aux angles d'EULER.

En se basant sur le calcul de passage entre repères voisins, il est simple de déterminer les différentes matrices suivantes :

Passage du repère (O - x y z) au repère (O - x' y' z')

$$x = R_z(\phi)x' \qquad avec: \quad R_z(\phi)=\begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \text{.....(C.1)}$$

Passage du repère (O - x' y' z') au repère (O - x'' y'' z'')

$$x' = R_{x'}(\theta)x'' \qquad avec: \quad R_{x'}(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \text{.....(C.2)}$$

Passage du repère (O - x'' y'' z'') au repère (O - xb yb zb)

$$x'' = R_{z''}(\psi)x^b \qquad avec: \quad R_{z''}(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{.....(C.3)}$$

les coordonnées du point xb du repère O – xb yb zb par rapport au repère de base O – x y z sont données par l'expression suivante :

$$x = R(\phi,\theta,\psi)x^b \qquad avec: \qquad R(\phi,\theta,\psi) = R_z(\phi)R_{x'}(\theta)R_{z''}(\psi) = \begin{bmatrix} n_x & t_x & b_x \\ n_y & t_y & b_y \\ n_z & t_z & b_z \end{bmatrix} \text{...(C.4)}$$

tel que :

$$\begin{cases} n_x = \cos\phi\cos\psi - \sin\phi\cos\theta\sin\psi \\ t_x = -\cos\phi\sin\psi - \sin\phi\cos\theta\cos\psi \\ b_x = \sin\phi\sin\theta \\ n_y = \sin\phi\cos\psi + \cos\phi\cos\theta\sin\psi \\ t_y = -\sin\phi\sin\psi + \cos\phi\cos\theta\cos\psi \text{ .....(C.5)} \\ b_y = -\cos\phi\sin\theta \\ n_z = \sin\theta\sin\psi \\ t_z = \sin\theta\cos\psi \\ b_z = \cos\theta \end{cases}$$

Les angles d'EULER sont très largement utilisés en robotique.

EXPRESSIONS ANALYTIQUES DU MODELE GEOMETRIQUE DIRECT ET INVERSE DU ROBOT AID

C.1. Modèle géométrique direct :

Les expression des composante h_{ij} $\left(i=\overline{1,4} \quad j=\overline{1,4}\right)$ sont données par :

$$
\begin{cases}
h_{11} = C6(C1C4 - S1C23S4) - S6(S1S4 + C1C23S4) \\
h_{12} = C6(C1S4 + C1S1C23) + S6(-S1S4 + C1C23C4) \\
h_{13} = S1S23C6 + C1S23S6 \\
h_{14} = d1S1S2C6 + d3C1S2S6 \\
h_{21} = -C5S6(C1C4 - S1C23S4) - C5C6(S1S4 + C1C23S4) + S23S4S5 \\
h_{22} = -C5S6(C1S4 + C1S1C23) - C5S6(-S1S4 + C1C23C4) - S23C4S5 \\
h_{23} = -S1S23C5S6 + C1S23C5C6 + C23S5 \\
h_{24} = -d1S1S2C5S6 + d3C1S2C6 + S5(d3C2 + d1 + d2) + d4S5 \\
h_{31} = S5S6(C1C4 - S1C23S4) + S5C6(S1C4 + C1C23S4) + S23S4C5 \\
h_{32} = S5S6(C1S4 + S1C23C4) - S5C6(C1C23C4 - S1S4) - S23C4C5 \\
h_{33} = S1S23S5S6 - C1S23S5C6 + C23C5 \\
h_{34} = d1S1S2S5S6 - d3C1S2S5C6 + C5(d3C2 + d1 + d2) + d4C5 \\
h_{41} = 0 \\
h_{42} = 0 \\
h_{43} = 0 \\
h_{44} = 1
\end{cases}
$$

C.2. Modèle géométrique inverse :

Les coefficients A_k $\left(k = \overline{1,16}\right)$ sont donnés par les expressions suivantes :

$$A_1 = -2d3(p_z - 1)$$
$$A_2 = p_x \cos q_1 + p_y \sin q_1$$
$$A_3 = -2d3A_2$$
$$A_4 = d4^2 - d3^2 - (p_z - d1)^2 - A_2^2$$
$$A_5 = A_1^2 + A_3^2$$
$$A_6 = \sqrt{A_5 - A_4^2}$$
$$A_7 = A_1 A_4 + A_3 A_6$$
$$A_8 = A_3 A_4 - A_1 A_6$$
$$A_9 = -(p_z - d1)\sin q_2 - A_2 \cos q_2 + d3$$
$$A_{10} = -A_2 \sin q_2 + (p_z - d1)\cos q_2$$
$$A_{11} = b_x \sin q_1 - b_y \cos q_1$$
$$A_{12} = -\cos(q_2 + q_3)\left[b_x \cos q_1 + b_y \sin q_1\right] - b_z \sin(q_2 + q_3)$$
$$A_{13} = -\cos q_4\left[\cos(q_2 + q_3)\left[b_x \cos q_1 + b_y \sin q_1\right] + b_z \sin(q_2 + q_3) + \sin q_4\left[b_x \cos q_1 + b_y \sin q_1\right]\right]$$
$$A_{14} = -\sin(q_2 + q_3)\left[b_x \cos q_1 + b_y \sin q_1\right] + b_z \cos(q_2 + q_3)$$
$$A_{15} = -\cos q_4\left[n_x \sin q_1 - n_y \cos q_1\right] - \sin q_4\left[\cos(q_2 + q_3)\left[n_x \cos q_1 + n_y \sin q_1\right] + n_z \sin(q_2 + q_3)\right]$$
$$A_{16} = -\cos q_4\left[t_x \sin q_1 - t_y \cos q_1\right] - \sin q_4\left[\cos(q_2 + q_3)\left[t_x \cos q_1 + t_y \sin q_1\right] + t_z \sin(q_2 + q_3)\right]$$

REFERENCES BIBLIOGRAPHIQUES

REFERENCES BIBLIOGRAPHIQUES

(1) – Documentation de l'AID.

(2) – Haruhiko ASADA et Jean-Jacques E. SLOTINE
« ROBOT ANALYSIS AND CONTROL »
A Wiley- Interscience Publication, 1985.

(3) – Haruhiko ASADA et Kamel YOUCEF-TOUMI
« DIRECT DRIVE ROBOTS : Theory and Practice »
The MIT Press, 1987.

(4)- BARRACO et CUNY
« CONCEPTION MECANIQUE, CINEMATIQUE ET DYNAMIQUE DES ROBOTS »
Revue Française de Méconique, 1987.

(5) – C. BLUME, W. JAKOB ET J. FAVARO
« PASRO ET PASRO/C : le Pascal et le langage C appliqués à la robotique »
Editions MASSON, 1988.

(6) – DELBOS
« TENDANCES ACTUELLES ET INTERROGATIONS EN ROBOTIQUE »
Revue Française de Méconique, 1987.

(7) – DOMBRE et FOURNIER
« CONCEPTION ASSISTEE PAR ORDINATEUR EN ROBOTIQUE »
Revue Française de Méconique, 1987.

(8) – « FLEX MOTION User Manuel »
nuLogic®, 1997.

(9) – « FLEX MOTION Command User Manuel »
nuLogic®, 1997.

(10) – Jean- Noël FOULC et Pierre LOPEZ
« initiation à la robotique : définition et aspects théoriques »
Le Nouvel Automatisme- septembre 1980.

(11) – FRANKE
« MA FORMATION AVISUAL BASIC 4.0 »
EDITIONS Micro Application, 1995.

(12) – J-F. GERMAIN
« COMMENT ABORDER UN PROJET ROBOTIQUE »
Revue Française de Mécanique, 1987.

(13) – Giuseppina GINI, Maria GINI, Renato GINI et Dario GIUSE
« A MULTI TASK SYSTEM FOR ROBOT PROGRAMMING»
Institut Polytechnique de Milan, 1979.

(14) – HALVORSON
« FORMATION A VISUAL BASIC 4.0 »
The Microsoft Press

(15) – C. ISIK et A. MEYSTEL
« KNOWLEDGE BASED PILOT FOR INTELLIGENT MOBILE AUTONOMOUS ROBOT »
ROBOTICS AND AUTOMATION, IEEE 1986.

(16) – KOPLEWICZ
« POINT SUR LA NORMALISATION NATIONALE ET INTERNATIONALE SUR LES ROBOTS INDUSTRIELS »
Revue Française de Mécanique, 1987.

(17) – J.-P. LALLEMAND et S. ZEGHLOUL
« ROBOTIQUE ASPECTS FONDAMENTAUX : Modélisation mécanique, CAO robotique, Commande »
Editions MASSON 1994.

(18) – J-C. LATOMBE
« SYSTEMES DE PROGRAMMATION POUR LA ROBOTIQUE »
Revue Française de Méconique, 1987.

(19) – MC KINNEY
« VISUAL BASIC SANS LIMITES »
The Microsoft Press.

(20) – Hacéne MELOUAH
« CONTRIBUTION A L'ETUDE DE LA PROGRAMMATION ET DE LA COMMANDE D'UN ROBOT MANIPULATEUR »
Thèse de Docteur Ingénieur en Automatique – université de Franche-Comté – 1981.

(21) – « ROB' 95 »
Actes des Journées d'Etudes sur la Robotique et son Environnement.
Ecole Militaire Polytechnique du 16 au 18 septembre 1995.

(22) – C-W. de SILVA
« CONTROL SENSORS AND ACTUATORS »
PRENTICE HALL, 1989.

(23) – « THEORY OF ROBOT CONTROL »
Edited for : The European Summer School of the Laboratory of Automatic Control of Grenoble, session 6 : september 7-11, 1992, Grenoble, France.

(24) – « Mise en oeuvre de la connexion AID Flex Motion PC »
Mini Projet EMP, 1998.