8/90

الجمهورية الجزائرية الديمقراطية الشعبية
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

1 EDC

وزارة التعليم و البحث العلمي
Ministère de l'Enseignement et de la Recherche Scientifique

ECOLE NATIONALE POLYTECHNIQUE

DEPARTEMENT Electronique

PROJET DE FIN D'ETUDES

SUJET

COMMANDE D'UN ROBOT A 6 DEGRES DE LIBERTE PAR MICRO PC

Proposé par :
B. BOUSSEKSOU

Etudié par :
A.R.FRIDI

Dirigé par :
B. BOUSSEKSOU

PROMOTION JUIN 90

E.N.P. 10, Avenue Hacen Badi - El-Harrach - Alger

الجمهورية الجزائرية الديمقراطية الشعبية
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التعليم و البحث العلمي
Ministère de l'Enseignement et de la Recherche Scientifique

ECOLE NATIONALE POLYTECHNIQUE

DEPARTEMENT Electronique

PROJET DE FIN D'ETUDES

SUJET

COMMANDE D'UN ROBOT A 6 DEGRES DE LIBERTE PAR MICRO PC

Proposé par :

Etudié par :

Dirigé par :

PROMOTION

E.N.P. 10, Avenue Hacen Badi - El-Harrach - Alger

R E M E R C I E M E N T S

Je remercie de tout mon coeur tous ceux qui ont contribué de près ou de loin à ce modeste travail par leurs conseils ou par leurs prières .

S O M M A I R E

PREAMBULE

Un robot est un manipulateur reprogrammable, multifonctionnel, projeté pour déplacer des matériells, des pièces, des outils, des composants spécialisés selon des mouvements programmés et pour éxécuter une vaste gamme de taches. Il constitue en fait une partie d'une ligne de production automatique flexible.

Le robot industriel est né il y a quelques décennies, mais il n'est devenu une technologie intéressante du point de vue économique que grace au développement récent de la micro-électronique et en particulier des microprocesseurs qui ont réduit considérablement le cout de la partie électrique par rapport à la partie mécanique.

CHAPITRE 1

Introduction

I INTRODUCTION :

L'objet de notre travail est la commande d'un bras articulé à 6 degrés de liberté , se trouvant au laboratoire de robotique au departement de mécanique ,par un micro-ordinateur compatible PC .

Ce mini-robot est un exemple typique de robots industriels pouvant être commandés par un calculateur .

L'exposé de ce travail est divisé en cinq châpitres et un annexe .

Le châpitre II décrit les aspects mécaniques et électriques constituant le mini-robot .

Dans le chapitre III ,les notions de base sont exposées : introduction aux moteurs pas à pas (hybrides en particulier), leurs différents modes d'alimentation , ainsi que les relations liant angle/pas pour chacun des six moteurs équipant le mini-robot .

Une architecture de commande est arretée et une étude détaillée de tous les modules la constituant sont présentés dans le chapitre IV .

Le chapitre V a été consacré à la partie software devant gérer notre système de commande .

Avec le chapitre VI ,on conclut notre étude en exposant assez brièvement les possibilités d'exploitation futures de notre travail .

C H A P I T R E II

1- Structure mécanique

2- Structure électrique du mini-robot

STRUCTURE MECANIQUE :

La structure du mini-robot est projetée pour donner la souplesse maximale associéeà l'utilité pratique maximum .Elle reproduit parfaitement un bras humain (bras,avant bras,poignet main). Ce mini robot est décomposé en cinq parties fondamentales (voir fig 2-a) :

LA BASE :

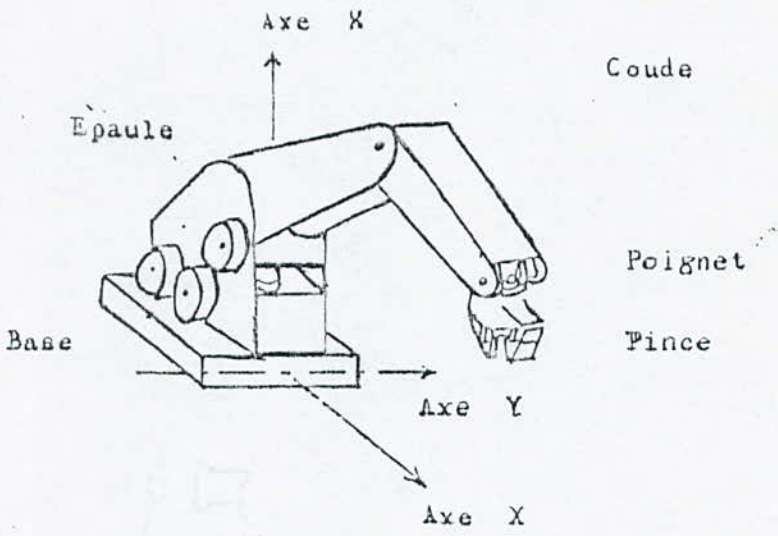
La base n'a pas seulement la fonction de soutenir le reste du bras,elle contient aussi les circuits de l'interface et le moteur qui commande la rotation.

L'EPAULE :

L'épaule qui tourne sur la base au moyen de l'engrenage principal porte cinq moteurs et leurs réducteurs qui s'accouplent avec les engrenages de l'avant bras.

LE BRAS :

Le bras tourne autour d'un axe horizontal de l'épaule.La partie inférieure porte les engrenages et les cables qui



ARCHITECTURE DU MANIPULATEUR VENETA

fig 2 - a

mettent en mouvement le coude,le poignet et la main.

L'AVANT BRAS :

L'avant bras tourne autour d'un axe horizontal sur le bras et porte les engrenages coniques du poignet.

LE POIGNET ET LA MAIN :

Les deux mouvements de la main et du poignet,la rotation autour de la main et la rotation de la main autour de l'axe horizontal (vers le haut et vers le bas) dépendent de la combinaison de deux mouvements indépendants.On obtient le mouvement de tanguage en mouvant les deux engrenages coniques dans des directions opposées et le mouvement vers le haut et vers le bas en mouvant le couple conique dans la même direction. On réalise la combinaison des deux mouvements en faisant tourner un des engrenages coniques plus que l'autre.
La main dotée de trois doigts ayant des extremités en caoutchouc peut simplememt s'ouvrir ou se refermer.

STRUCTURE ELECTRIQUE :

L'articulation des différentes parties citées plus haut,se fait au moyen de six moteurs pas à pas hibrides identiques,à

quatre phases chacun.Ces derniers de par leur fonctionnement en boucle ouverte,offrent l'avantage d'une électronique de commande assez simple.

Un apperçu sur la constitution de ce type moteur ainsi que sur les modes d'alimentation qu'ils offrent s'avère nécessaire pour pouvoir arreter l'architecture globale de commande.

C H A P I T R E III

1- Les moteurs pas à pas

2- Les moteurs pas à pas hybrides

3- Alimentation des moteurs pas à pas

4- Relation pas/angle de rotation

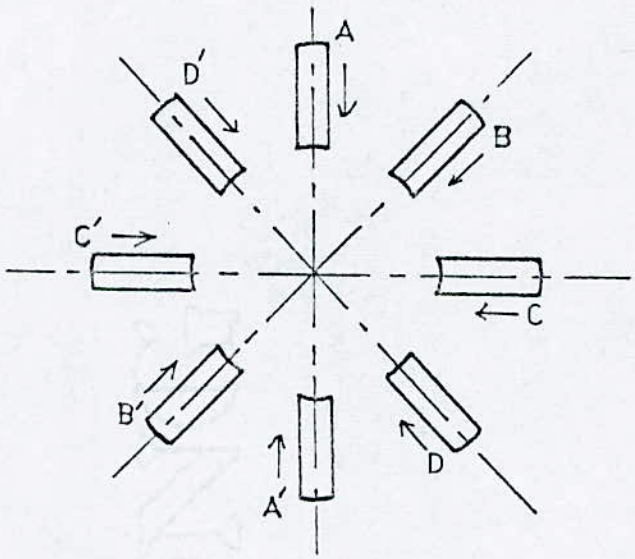
LES MOTEURS PAS A PAS :

Ce sont des moteurs qui permettent de convertir un signal électrique digital en mouvement angulaire discret d'amplitude constante, ce qui entraine un positionnement très précis.
D'autre part, de par sa nature, le moteur pas à pas est un moteur synchrone, de telle manière qu'il y ait cohérence entre le signal d'alimentation et la position du moteur. La figure (3-a) illustre de manière simplifiée cette idée pour un moteur pas à pas à huit pas/tour .
Il existe différents types de moteurs pas à pas :

- -A aimant permanent
- -A réluctance variable
- -Hybride
- -Hybride à aimant rotorique

LES MOTEURS PAS A PAS HYBRIDES :

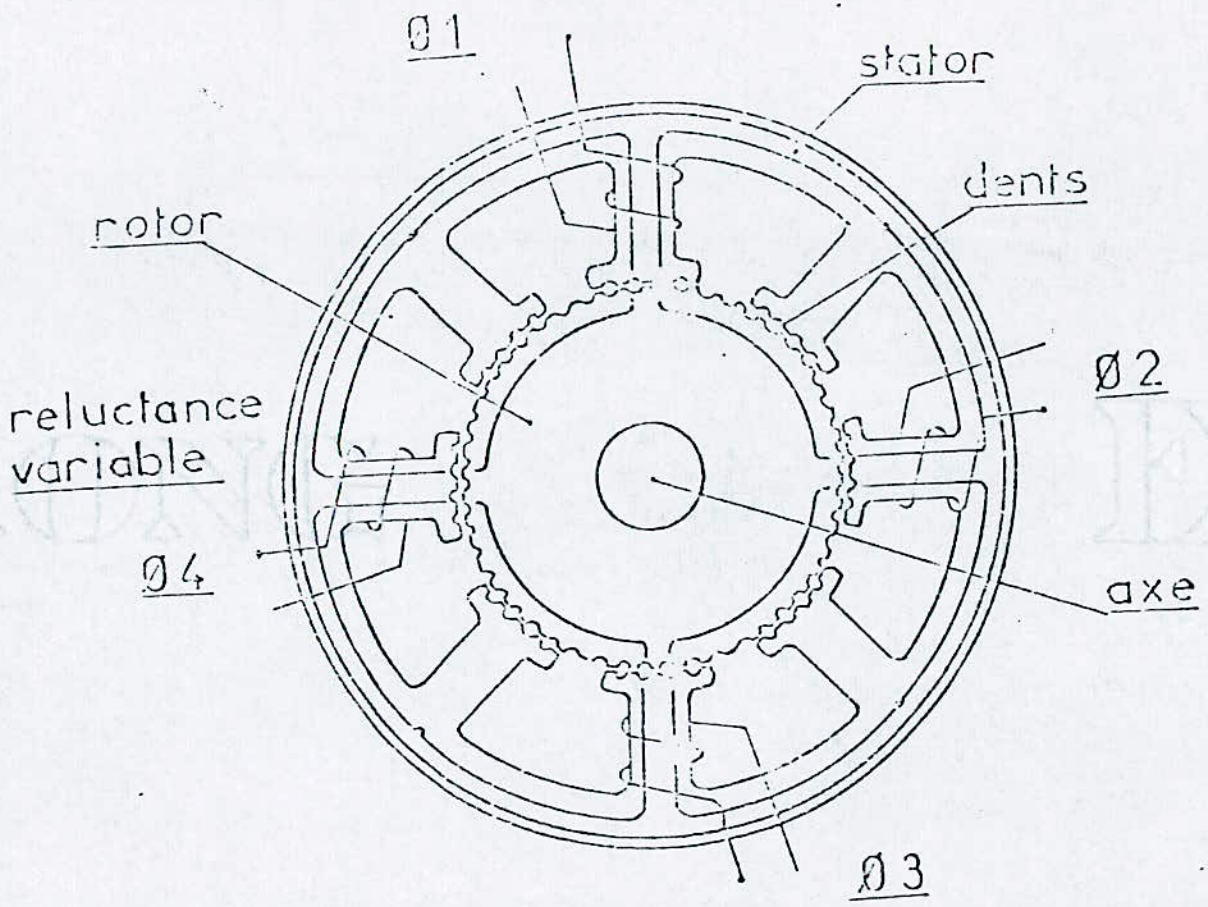
Ce type de moteur allie les avantages des moteurs à aimant permanent et ceux des moteurs à réluctance variable. Il a un couple proportionnel au courant et peut présenter un grand nombre de pas par tour. La structure d'un moteur pas à pas hybride présente deux stators identiques portant chacun huit épanouissements appelés "poles", munis de cinq dents espacées



	AA'	BB'	CC'	DD'
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

MARCHE DU MOTEUR

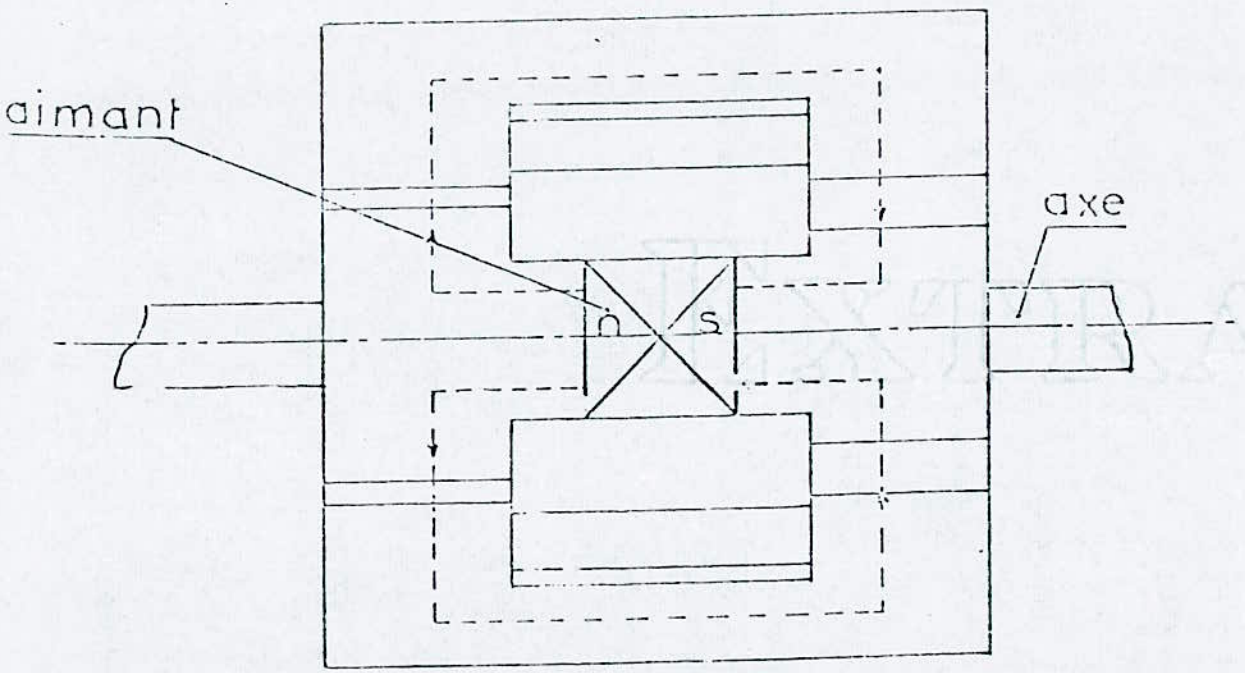
fig 3-a



coupe perpendiculaire à l'axe de rotation

fig 3-b

coupe suivant l'axe de rotation



coupe du moteur hybride

fig 3-c

de :

360/48 = 7.5 degres

Le rotor est également double et comporte deux roues de cinquante dents espacées régulièrement . Les roues sont décalées entre elles d'un demi pas dentaire.Un aimant interposé entre les élements du rotor magnétise l'un en nord et l'autre en sud.

Les stators portent deux enroulements communs répartis chacun sur la moitié des pôles (fig 3-b et 3-c) .

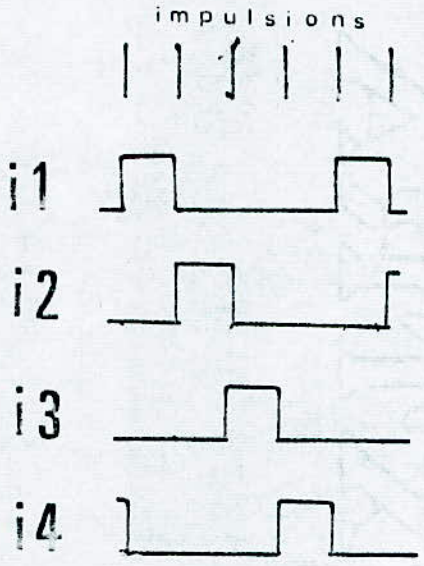
ALIMENTATION DES MOTEURS PAS A PAS :

Pour qu'un moteur pas à pas puisse tourner d'un pas ou d'une fraction de pas,il faut lui envoyer successivement un courant I parcourant les enroulements du stator afin de créer un champ tournant que suivra le rotor.

Ainsi de nombreuses séquences d'alimentation peuvent etre envisagées.On peut distinguer principalement cinq modes de fonctionnement pouvant etre appliqués quelque soit le nombre de phases du moteur:

MODE 1 :

On alimente une phase à la fois par le courant I du bobinage



Chronogramme

mode 1

Tab des seq

	i1	i2	i3	i4
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

. fig 3d

C'est dans ce mode que l'on défini l'angle de pas θp . La fig 3-d montre la séquence d'alimentation ainsi que la forme théorique des courants pour une alimentation unipolaire.

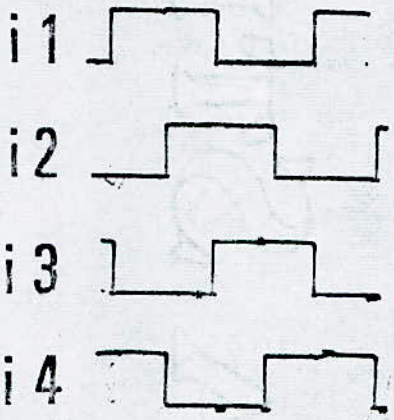
MODE 2 :

Dans ce mode nous avons un couple plus important , le rotor venant se positionner entre deux phases.Le couple sera $\sqrt{2}$ fois plus grand que celui obtenu dans le mode précedent.

Ainsi deux phases du moteur sont alimentées simultanément par des courants In . La fig 3-e montre la forme du courant ainsi que la table des séquences.

MODE 3 :

On s'apperçoit avec ces modes de fonctionnement que l'on peut obtenir deux positions stables,une lorsqu'une seule phase est alimentée,l'autre lorsque deux phases sont alimentées.Ainsi la combinaison des deux modes déjà cités permet un fonctionnement en pas et en demi pas.Il en résulte que le nombre de pas et le nombre d'impulsions de commande doivent être double pour réaliser un tour complet . Cette commande en double précision est de réalisation simple mais

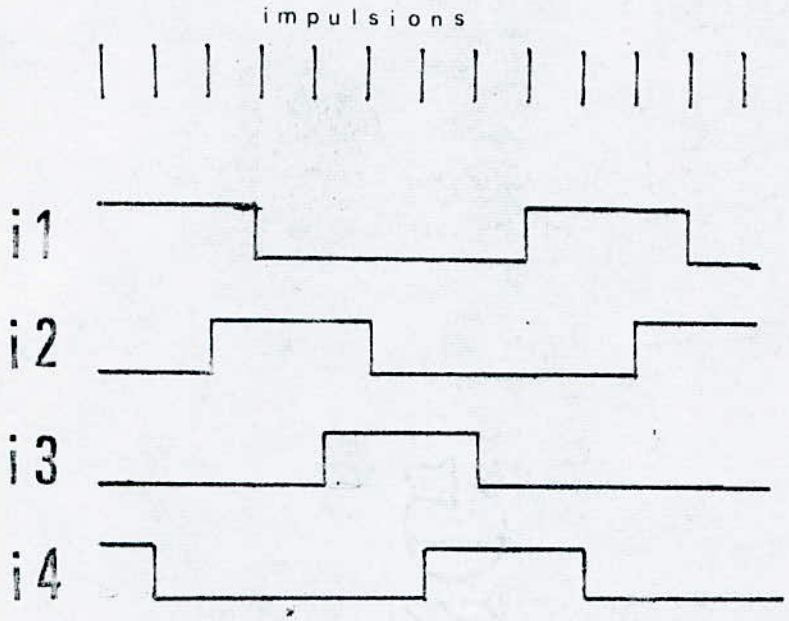


Chronogramme ▲ mode 2

Tab des seq ▼

	i1	i2	i3	i4
1	1	1	0	0
2	0	1	1	0
3	0	0	1	1
4	1	0	0	1

fig 3e

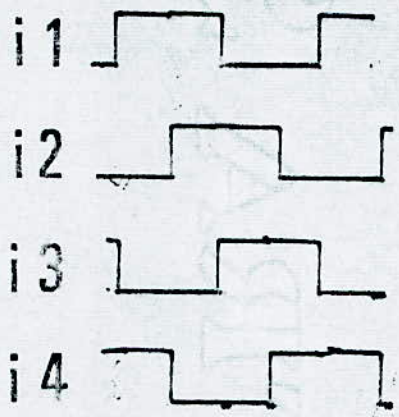


Chronogramme ▲
Tab des seq ▼

mode 3

	i1	i2	i3	i4
1	1	0	0	0
2	1	1	0	0
3	0	1	0	0
4	0	1	1	0
5	0	0	1	0
6	0	0	1	1
7	0	0	0	1
8	0	0	0	1

fin 3f



Chronogramme ▲ mode 2

Tab des seq ▼

	i1	i2	i3	i4
1	1	1	0	0
2	0	1	1	0
3	0	0	1	1
4	1	0	0	1

fig 3e

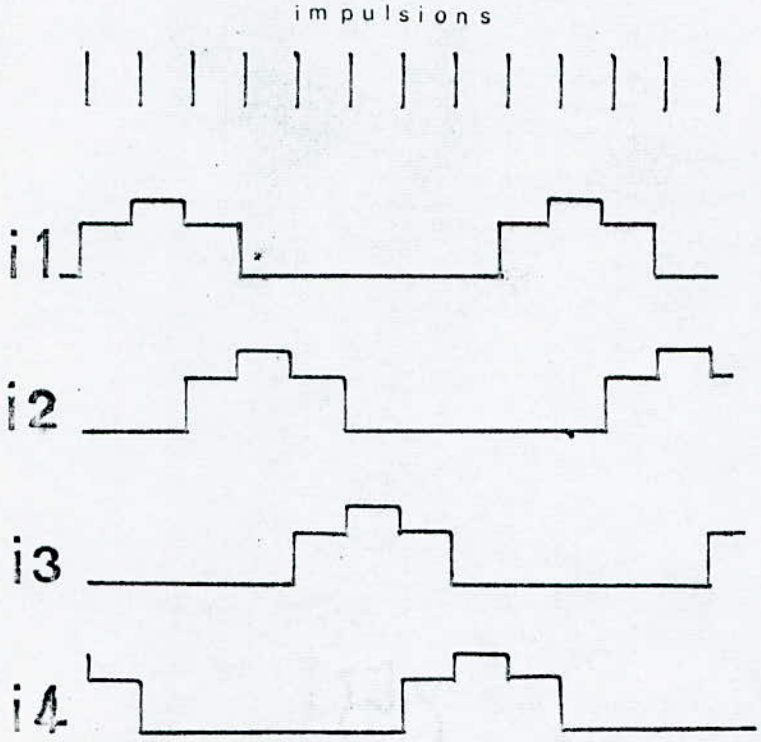
présente toutefois l'inconvénient de ne pas présenter le même couple moteur pour tous les pas .La figure 3-f montre la table des séquences ainsi que la forme des courants .C'est le mode qu'on a choisi pour notre application .

MODE 4 :

Ce mode de fonctionnement est réservé lorsqu'il s'agit d'un moteur de type hybride,ou lorsqu'on désire un fonctionnement régulier en demi pas . Lorsqu'un seul courant existe dans le moteur,celui-ci est fixé à la valeur $I\sqrt{2}$ on obtient ainsi une précision de positionnement double et les même performances en couple du mode 2 (fig 3-g) .

MODE 5 :

Ce mode est généralement appelé " MINISTEPPING ",cette technique est semblable à une alimentation en courant sinusoidale synchrone . Elle nécéssite cependant des circuits de commande trés élaborés .



Chronogramme ▲

Tab des seq ▼

mode 4

	i1	i2	i3	i4
1	1	0	0	1
2	1^+	0	0	0
3	1	1	0	0
4	0	1^+	0	0
5	0	1	1	0
6	0	0	1^+	0
7	0	0	1	1
8	0	0	0	1^+

$1^+ = i\sqrt{2}$

fig 3 g

RELATION ENTRE LE PAS DU MOTEUR ET L'ACCROISSEMENT ANGULAIRE :

On indique ci-dessous les calculs qui permettent de connaitre la relation entre chaque pas du moteur et l'accroissement angulaire de chaque joint .

BASE :

*ANGLE DE PAS * RAPPORT 1 * RAPPORT 2*

$$7.5 * \frac{20 \text{ DENTS}}{72 \text{ DENTS}} * \frac{12 \text{ DENTS}}{108 \text{ DENTS}}$$

$= 0.2314$ degree/pas

$= 4.32152$ pas/degree

EPAULE :

$$7.5 * \frac{14 \text{ DENTS}}{72 \text{ DENTS}} * \frac{12 \text{ DENTS}}{108 \text{ DENTS}}$$

$= 0.162$ degre/pas

$= 6.17284$ pas/degre

COUDE :

Identique à l'épaule .

POIGNET :

Identique à la base .

MAIN :

$$7.5 * \frac{20 \text{ DENTS}}{72 \text{ DENTS}} * \frac{12 \text{ DENTS}}{108 \text{ DENTS}} = 0.2314 \text{ °/pas}$$

$$\frac{\pi * d * 0.2314}{360 * 2} = \frac{0.0524}{2} \text{ mm}$$

$$= 0.0262 \text{ mm de mouvement de la main par pas du moteur}$$

Avec d = 26 mm (*diamètre de la poulie*) .

Le mouvement total de la main de la position ouverte à fermée est donc de 20 mm .

L'angle réalisé par chaque doigt = 50 degrés soit :

$$\frac{50}{20} * 0.0262 \text{ mm} = 0.0655 \text{ degrés/pas}$$

$$= 15.2672 \text{ pas/degrés}$$

C H A P I T R E IV

1- *Architecture générale du système*

2- *Carte d'interface parallèle*

* *Le PPI 8255*

* *Adressage de la carte*

* *Adressage des moteurs*

3- *Carte base de commande*

ELECTRONIQUE DE COMMANDE :

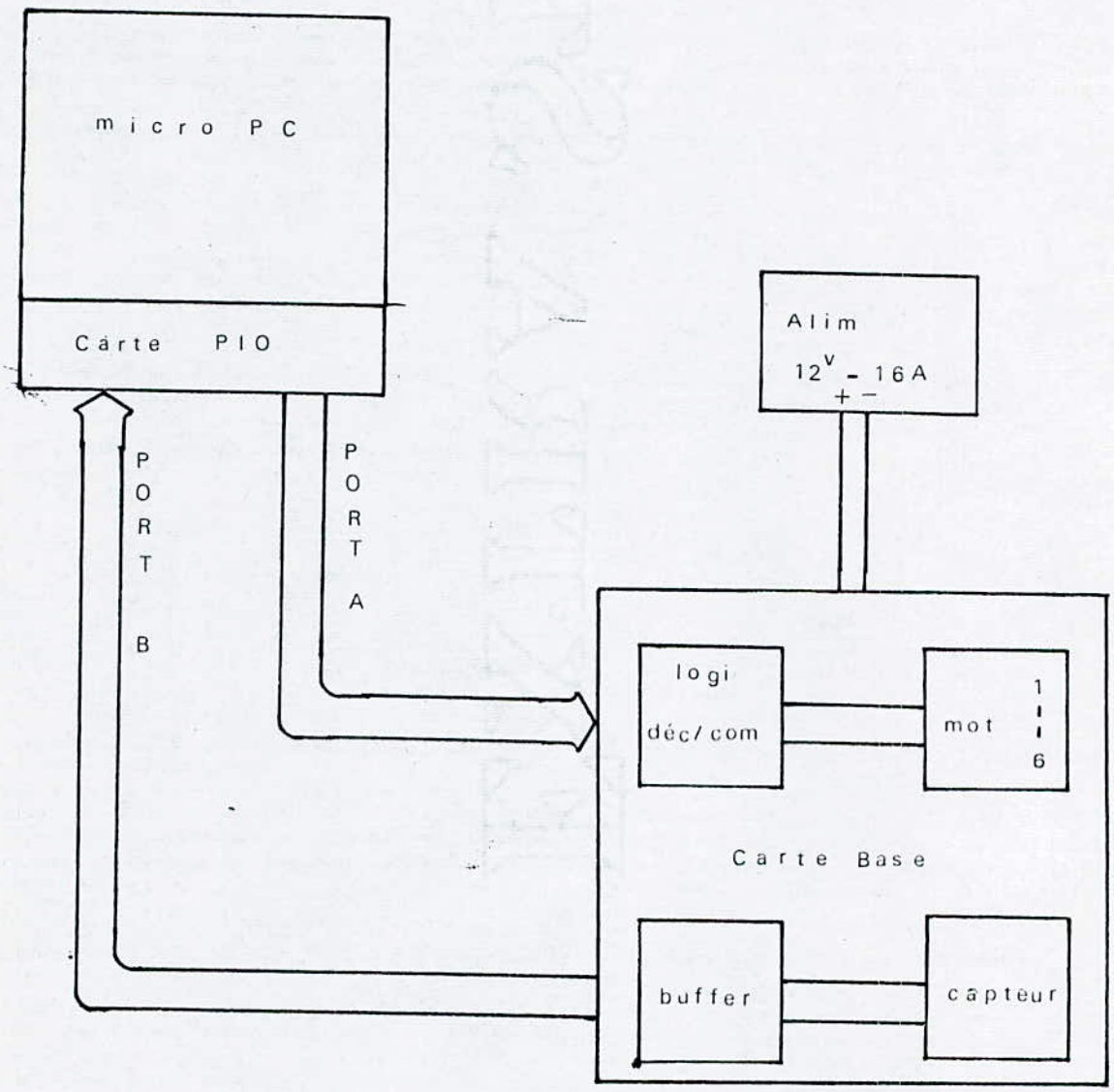
ARCHITECTURE GENERALE DU SYSTEME :

L'architecture générale du système se scinde en deux parties distinctes (fig 4-a) :

1- Une carte de base accompagnant le bras, devant d'une part interpreter tous les signaux de commande venant de l'ordinateur et acheminer ceux-ci aux moteurs appropriés et d'autre part fournir à l'ordinateur les informations relatives à l'état des capteurs de position associés à chacun d'eux .

2- Une carte d'interface organisée autour du PPI 8255 d'INTEL , elle a pour rôle d'interfacer la carte base avec le micro-ordinateur PC.

On analyse maintenant ,en détail,toutes les parties éléctroniques en se réferant aux schémas correspondants .



Architecture Gle du système

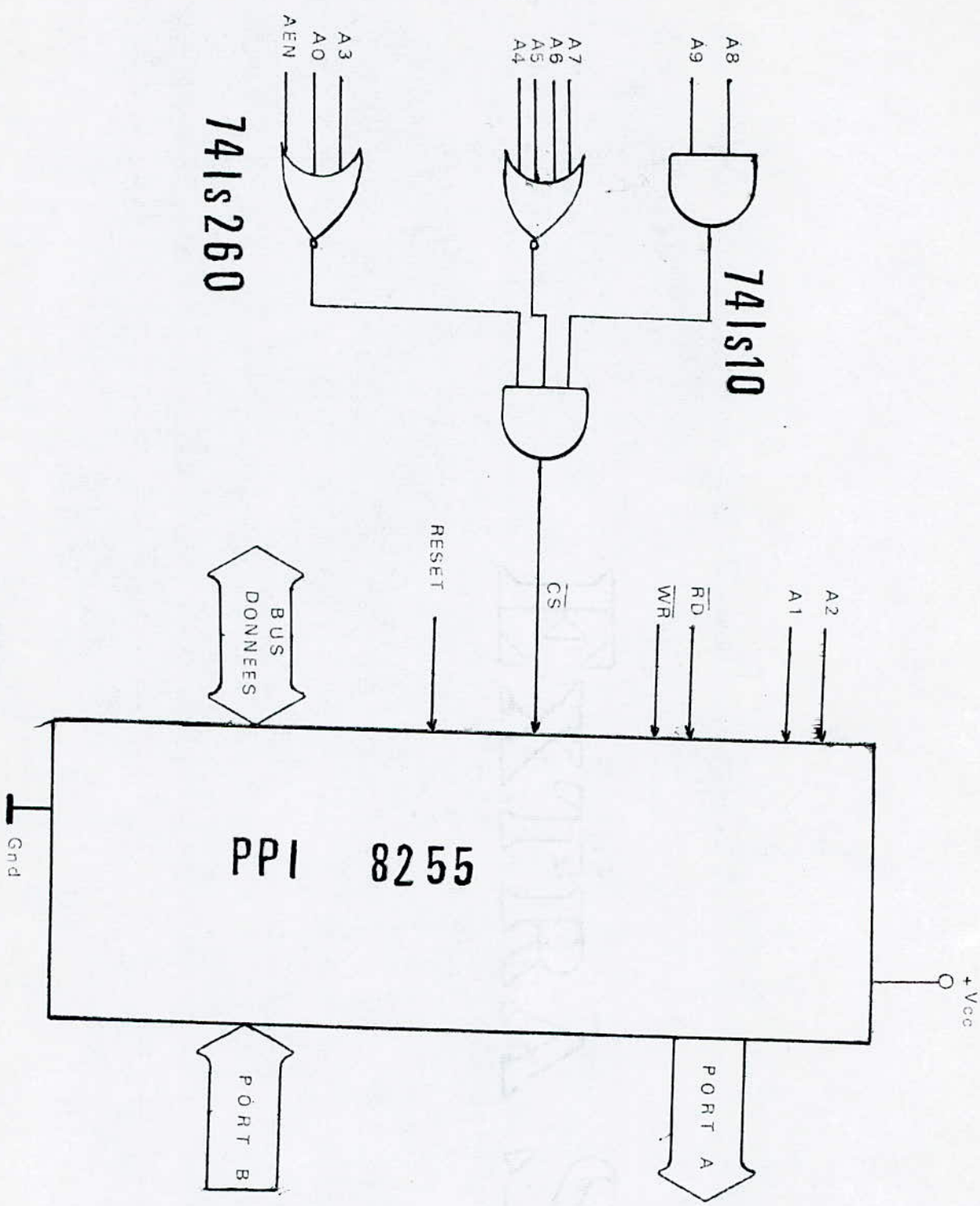
fig 4a

CARTE D'INTERFACE :

Cette carte (fig 4-b) a accés au bus d'extention du PC et est inserée donc sur un des slots vacants du micro-ordinateur ou sont disponibles tous les signaux utiles du micro-processeur .

Le tableau suivant énumère les signaux utiles à notre réalisation et prélevés sur ce slot à 62 broches .

PIN	FONCTION	PIN	FONCTION
A2	bus de donnée D7	B2	reset
A3	" " " D6	B9	+ 12 v
A4	" " " D5	B3	+ 5 v
A5	" " " D4	A31	bus d'adr A1
A6	" " " D3	A30	" " A2
A7	" " " D2	A29	" " A3
A8	" " " D1	A28	" " A4
A9	" " " D0	A27	" " A5
B1	gnd	A26	" " A6
B31	gnd	A25	" " A7
B10	gnd	A24	" " A8



(Carte d'interface paralléle)

fig 4b

B13	IOR lecture E/S	A23	"	" A9
B14	IOW écriture E/S	A22	"	" A10
B11	AEN	A21	"	" A11

LE PPI 8255 :

Ce circuit comprend trois ports d'entrées/sorties parallèles de 8 bits chacun et un registre de commande . Il permet en outre trois modes de fonctionnement :

MODE 0 : entrées/sorties de base

MODE 1 :entrées/sorties échantillonnées

MODE 2 :bus bidirectionnel d'entrées/sorties

Le mode adopté pour notre application est le mode 0 .

MODE 0 :

Le sens de transfert dans ce mode est impérativement le même à l'intérieur de chaque port (ou demi port pour le port C). Le port A sera programmé en sortie ;le port B sera programmé en entrée .On notera par ailleurs que si dans ce mode,les données sont mémorisées dans le PPI 8255 ,il n'en est pas de même pour les données en entrée dont il importe de les lire dès leur validation .

Le mot de commande adopté et qui découle du choix fait précedemment est le suivant :

1 0 0 0 X 0 1 X

Indicateur 1 = actif ----	1
	0
00 : *Mode 0 pour grA* ---	
	0
0 : *Port A en sortie* --	0
X : *Indifferent* -------	X
0 : *Mode 0 pour grB* --	0
1 : *Port B en entree* --	1
X : *Indifferent* -------	X

ADRESSAGE DE LA CARTE :

On adoptera , pour l'adressage de la carte , la technique d'entrées/sorties avec instructions d'entrées/sorties .

Un apperçu sur l'affectation des zones mémoires permet de constater l'existence d'un espace mémoire vacant entre les adresses 0300H et 031FH (destiné à recevoir les cartes d'extension).C'est dans cette zone mémoire que nous implanterons notre carte . Une logique de décodage adéquate permettera de pointer sur cette zone .

On notera d'autre part qu'il n'est pas nécessaire de décoder les 20 bits du bus d'adresse car lors d'un transfert relatif à un périphérique , dans cette structure d'adressage , les bits A19 à A10 sont toujours à zéro,seuls A0à A9 participent au décodage .L'affectation mémoire de chacun des registres du PPI 8255 est la suivante :

0300H ----------> PORT A
0302H ----------> PORT B
0304H ----------> PORT C
0306H ----------> REGISTRE DE COMMANDE

La sélection de l'un de ces registres se fait par les bits A1 et A2 selon le tableau suivant :

A2	A1	registre
0	0	port A
0	1	port B
1	0	port C
1	1	registre de commande

La sélection du boitier se traduit par l'équation suivante (fig 4) :

$$CS = A8A9 \cdot \overline{A0 + AEN} \cdot \overline{A7 + A6 + A5 + A4 + A3}$$

Le signal AEN intervient dans la décodage afin d'éviter un conflit sur le bus d'adresse lors d'un transfert par DMA . Une adresse ne sera validée que lorsque AEN est à l'état bas .

ADRESSAGE DES MOTEURS :

Comme il a été déjà dit, les 3 bits A0, A1 et A2 sélectionnent les six moteurs équipant le robot. La correspondance entre les adresses et les moteurs est la suivante :

BITS : A2	A1	A0	MOTEUR
0	0	1	avant-bras
0	1	0	poignet 2
0	1	1	base
1	0	0	main
1	0	1	épaule
1	1	0	poignet 1

CARTE BASE :

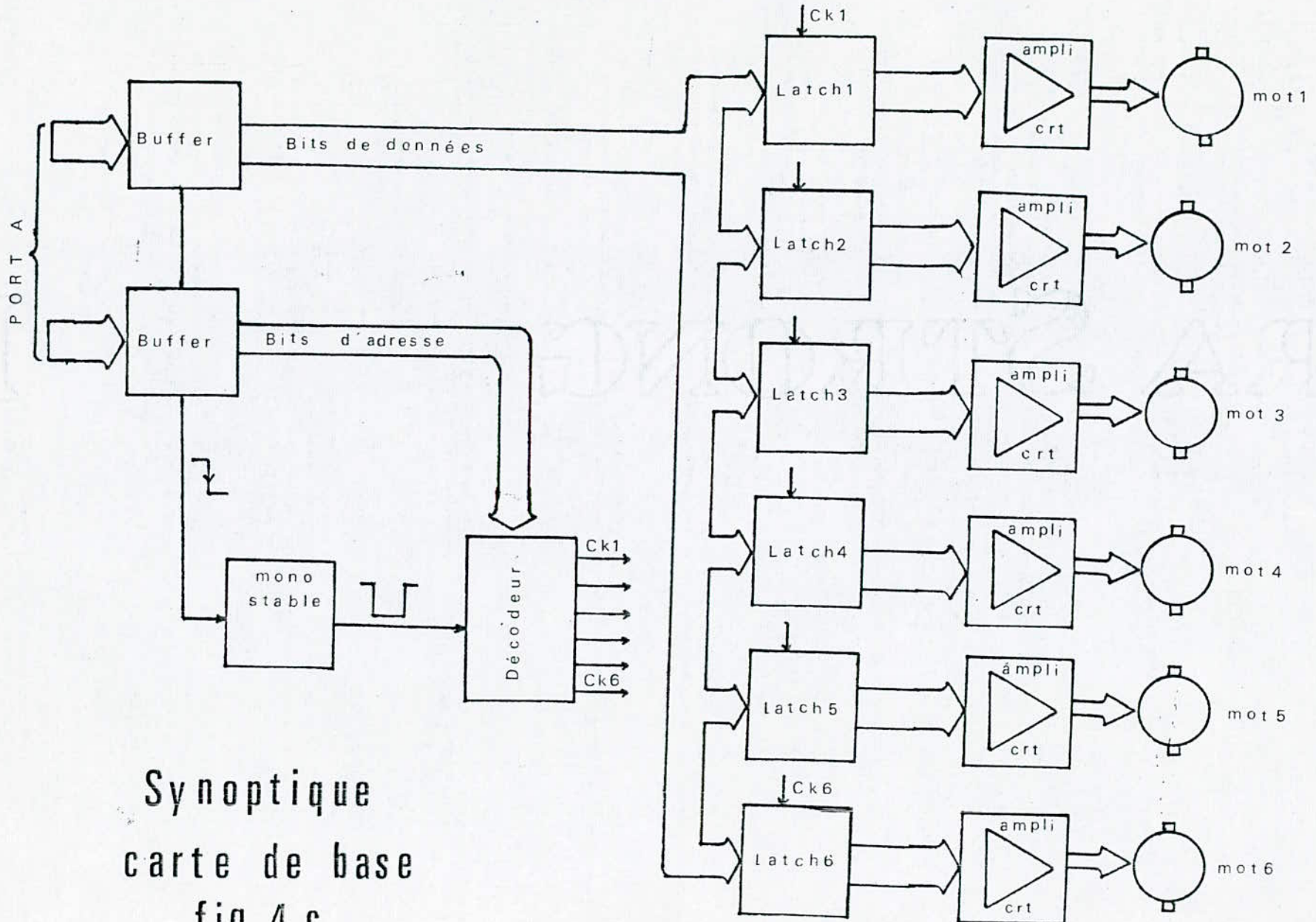
Cette carte contient la section de commande des six moteurs pas à pas du robot (fig 4-c) .

Les 8 bits de commande provenant du port A sont arrivent aux buffers intégrés IC1 et IC2 (LS 74 125).Le buffer du bit 3 de validation est relié à la pin 1 du monostable IC3 (74 LS 123) et va ainsi permettre de générer un front descendant sur cette pin , par l'envoi en deux temps du mot de commande (voir procédures MOTUP et MOTDW) : une première fois avec A3 égal à 1 et une seconde avec avec A3 égal à zéro .

On fera remarquer que les buffers IC1 et IC2 sont à l'état haute impédance lors du premier envoi du mot de commande et ils ne seront validés que lorsque la sortie du buffer du bit 3 passe à l'état bas et désactive ainsi l'état haute impédance de ces buffers .A3 est branché à une résistance R1 afin que cette ligne soit toujours haute .

Un apperçu sur les fiches données en annexe permet une meilleurs compréhension du fonctionnement de ces circuits .

La génération du front descendant sur la pin 1 du monostable va produire une impulsion d'horloge d'une durée détérminée par les valeurs de R2 et C2 selon la relation :



Synoptique
carte de base
fig 4 c

$t = 0.28\ R2 / C2 < 1 + 0.7 / R2 >$

De cette manière le décodeur IC4 (LS 74 138) produit une impulsion de mémorisation au latch de contrôle du moteur adressé (IC5 - IC10 LS 74 175) .

Les données relatives à la commande des phases des moteurs (bits A4,A5,A6,A7) vont en parallèle à tous les latchs.Seul celui qui est adressé par les bits A0,A1,A2 les transferera ensuite à la sortie pour commander effectivement les quatres enroulements du moteur .

Ces signaux à la sortie des six latchs attaquent les circuits de puissance IC11 à IC14 capablent de piloter les enroulements du moteur en fournissant à ces derniers un courant de l'ordre de 300 mA (12 v).

C H P I T R E V

1- Description et utilisation du programme

2- Organisation de l'aire buffer de données

3- Description du programme ROUT.ASM

4- Description du programme ROBOT.PAS

PROGRAMME DE GESTION DE LA CARTE :

Le programme développé pour gérer cette carte a été construit de façon modulaire . Il est composé de quelques routines bases écrites en language assembleur et d'une section de gestion de celles-ci écrite en TURBO PASCAL 5.0 . Les procédures écrites en assembleur sont déclarées dans le programme ROUT.ASM par la directive FAR indiquant qu'elles peuvent faire l'objet d'appel extra segment . La version objet de cette section ROUT.OBJ est ensuite liée à la partie PASCAL par la directive{ L $ ROUT.OBJ } .

Ce choix a été dicté d'une part par la nécessité d'adresser des organes d'entrées / sorties et d'autre part par la souplesse des directives de compilation et de lien offertes par le TURBO PASCAL 5.0 .
D'autre part on notera que les procédures écrites en assembleur peuvent etre utilisées dans n'importe quel autre programme .

DESCRIPTION ET UTILISATION DE LA CARTE :

Le programme complet de gestion du robot s'appelle :

ROBOT.PAS

Dès son lancement le programme propose les options suivantes:

MENU PRINCIPAL

1- RAZ MANUELLE
2- RAZ AUTOMATIQUE
3- ESSAI PAS PAR PAS
4- ETUDE D'UNE SEQUENCE
5- EXECUTION D'UNE SEQUENCE
6- EDITION D'UNE SEQUENCE
7- FIN

On analyse maintenant les différentes options du menu principal .

1- RAZ MANUELLE :

Cette section permet de porter manuellement le robot dans n'importe quelle position et de faire en sorte que celle ci soit la position zéro pour tous les mouvements ultèrieurs en positionnant à zéro tous les compteurs absolus .

2- RAZ AUTOMATIQUE :

En selectionnant cette option ,le programme remet tous les moteurs à la position zéro définie au départ .

3- ESSAI PAS PAR PAS :

Cette section permet de mouvoir le robot avec le clavier;un axe determiné correspond à chaque touche et

on peut inverser le sens du mouvement par l'intermediaire de la touche de CTRL .

4- ETUDE D'UNE SEQUENCE DE MOUVEMENT :

Une fois qu'on a effectué cette sélection ,le programme demande le nom de la séquence avec lequel on la mettra en mémoire . Le menu de correspondance entre les touches et les moteurs apparait alors et le programme se prépare pour accepter les pas de la séquence et pour leur sauvegarde sur disque . On peut mouvoir plusieurs axes en succession pendant chacun de ces pas . Le programme éxecute ensuite,pendant le mouvement sur séquence ,le mouvement simultanné des axes .

5- EXECUTION D'UNE SEQUENCE :

Cette section permet de mouvoir le robot grace à la commande d'une séquence enregistrée précedemment . Elle présente aussi plusieurs options d'éxecution :

- <C> mouvement continu sans arrêt .
- <P> mouvement par pas .
- <S> une seule éxecution de la séquence entiére .
- <C> éxecution cyclique de la même séquence .

6- EDIT D'UNE SEQUENCE :

Cette section du menu permet de controler et/ou de modifier les données d'une séquence déjà enregistrée .Cela est très utile si l'on veut apporter des modifications à une séquence ,sans devoir procéder à une nouvelle étude de celle ci . Le programme affiche pour chaque pas de la séquence le nombre de pas associés à chaque axe . On peut alors choisir une des options proposées en bas de l'écran :

- <N> on considére que la situation pas visualisée est exacte et on passe à la suivante .
- <M> on modifie les pas visualisés par les pas examinés .
- <A> on interrompt à ce point la séquence en éliminant tous les pas suivants .
- <ESC> on interrompt le processus d'édit sans rien modifier et on retourne au menu principal .

ORGANISATION DE L'AIRE BUFFER DE DONNEES :

L'aire buffer de données est constituée d'une part de varibles propres au programme ROUT.ASM et auxquelles le programme principal écrit en PASCAL n'a pas accès et d'autre part de variables déclarées dans le programme ROUT.ASM par la

directive EXTRN indiquant que ces dernieres sont déclarées et initialisées dans la section écrite en PASCAL et utilisées également par les procédures écrites en assembleur .

On notera une nouvelle fois les facilités offertes par le TURBO PASCAL 5.0, permettant de traiter des procédures écrites en assembleur de la même manière que celles écrites en PASCAL .

SEQTAB :C'est un tableau à huit positions contenant les séquences à suivre pour la commande du moteur .

ABSCO :C'est un tableau contenant les compteurs absolus des pas éxécutés par chaque moteur et permettant de palier à l'absence de capteur de la position zéro, offrant une gamme de comptage de -3200 a +3200 . Il est toujours mis à jour après chaque pas de mouvement .

RELCO :C'est un compteur contenant pour chaque moteur , un compteur relatif des pas qu'il faut effectuer (il est surtout utilisé par la routine command) .

CTPOS1 :C'est un tableau contenant pour chaque moteur le curseur au tableau SEQTAB , déterminant ainsi la position

actuelle des phases .

VITTAB :C'est un tableau contenant les paramètres de vitesse de mouvement de chaque axe .

MOTBUF :C'est une position mémoire , contenant l'information relative au numéro du moteur qu'il faut piloter .

DESCRIPTION DU PROGRAMME SOURCE ROUT.ASM :

L'archive ROUT.ASM contient les routines bases de mouvement du robot ; celle-ci sont complètement générales et peuvent etre utilisées également dans d'autres programmes .Ce sont :

MOTUP :Déplace en avant d'un pas le moteur détérminé par le contenu de la position MOTBUF .

MOTDW:Déplace en arrière d'un pas le moteur détérminé par le contenu de la position MOTBUF .

COMMAND :Commande le mouvement simultanné de tous les moteurs avec le nombre de pas specifié par le contenu des compteurs RELCO .

RESTOR :Repporte tous les axes du robot à zéro en exploitant les indications des compteurs absolus ABSCO .

MOVIN :Contient la routine de mouvement au moyen du clavier. Cette routine lit le caractère entré au clavier et en

fonction de la touche pressée (conformèment au menu affiché) déplace l'axe correspondant .

```
;*****************************************************************************
;*********          SEGMENT DE DONNEES          ***************************
;*****************************************************************************
;
;
Dseg    SEGMENT PUBLIC 'DATA'
                ;
                ;
        M1UP     EQU 'Q'            ;Rotation de la base
        M1DW     EQU 'q'
        M2UP     EQU 'W'            ;    "     de l'epaule
        M2DW     EQU 'w'
        M3UP     EQU 'E'            ;    "     du bras
        M3DW     EQU 'e'
        M4UP     EQU 'R'            ;    "     de la pince h/b
        M4DW     EQU 'r'
        M5UP     EQU 'T'            ;    "     de la pince SAM
        M5DW     EQU 't'
        M6UP     EQU 'Y'            ;    "     fermeture de la pince
        M6DW     EQU 'y'
        SORT     EQU 27             ;sortie
        SORT1    EQU 45             ;sortie
        TRAM     DW (?)
        MOTBUF   EQU TRAM-24
        PORTA    EQU 0300H
        PORTB    EQU 0302H
        PORTC    EQU 0304H
        RC       EQU 0306H
EXTRN   ABSCO:WORD,RELCO:WORD,CTPOS1:BYTE,VITTAB:WORD,SEQTAB:BYTE,

MOTADR:WORD,pacom:word

Dseg    ENDS
        ;
        PAGE
```

```
        ;
        ;*****************************************************************************
        ;**********      PROGRAMME PRINCIPAL      **********************
        ;*****************************************************************************
        ;
        ;
COMMENT */   ROUTINES UTILITAIRES /*
        ;
        ;
        ;
Cseg    SEGMENT PUBLIC 'CODE'
        ;
        ;
        ASSUME CS:Cseg,DS:Dseg
        PUBLIC MOTUP,MOTDW,RESTOR,COMAND,MOVIN,RAZ1,RAZ2
        ;
INIT    :  mov ax,dseg
           mov ds,ax
        ;
        ;*****************************************************************************
        ;**********         PROCEDURE MOTUP       **********************
        ;*****************************************************************************
        ;
        ;
COMMENT */    Réalise l'avance d'un pas du moteur dont l'adr se trouve
                          dans MOTBUF
              Utilisée par les procédures MOVIN,COMMAND,RESTOR        /*
        ;
        ;
MOTUP   PROC FAR
        ;
        ;
start   :  xor ax,ax              ;init du reg ax
           mov bx,(motbuf)        ;adr du mot dans bx
           mov al,ctpos1[bx]      ;posit de la dern seq envoyée
           add al,01h             ;passage à la seq suivante
           cmp al,09h             ;vérif si fin du tab SEQTAB
           jne cak                ;continuer si non
           mov ax,0001h           ;remise du curseur au début de SEQTAB
cak     :  mov ctpos1[bx],al      ;sauvegarde de la position actuelle
           mov cx,absco[bx]       ;incrémentation du compt absolu
           add cx,0002h
           mov absco[bx],cx
           mov si,ax              ;mise de la val du dépiac dans SEQTAB dans SI
           mov al,seqtab[si]      ;envoi de la seq sur quartet fort de al
           add ax,bx              ;envoi de l'adr du mot sur quartet faible de al
           add al,08h             ;mise à 1 le bit de validation
           mov dx,porta           ;adr du port A sur dx
           out dx,ax              ;envoi du mot de commande sur port A
           sub al,08h             ;mise à 0 du bit de validation
           out dx,ax              ;envoi du mot de commande sur port A
           ret
motup      endp
        ;
```

```
        ;
        ;*************************************************************************
        ;*********          PROCEDURE MOTDW          **********************
        ;*************************************************************************
        ;
        ;
COMMENT */    Réalise le recul d'un pas du moteur dont l'adr se trouve
                        dans MOTBUF
            Utilisée par les procédures MOVIN,COMMAND,RESTOR        /*
        ;
        ;
MOTDW   PROC FAR
        ;
        ;
start1  :  xor ax,ax              ;init du reg ax
           mov bx,(motbuf)        ;adr du moteur dans bx
           mov al,ctpos1[bx]      ;position de la dern seq envoyée
           sub al,01h             ;passage à la seq suivante
           cmp al,00h             ;vérif si début du tab SEQTAB
           jne dow                ;continuer si non
           mov al,08h             ;remise du curseur à la fin de SEQTAB
dow     :   mov ctpos1[bx],al      ;sauvegarde de la position actuelle
           mov cx,absco[bx]       ;incrémentation de compt absolu
           sbb cx,0002h
           mov absco[bx],cx
           mov si,ax              ;mise de la valeur du dépi dans SEQTAB dans SI
           mov al,seqtab[si]      ;envoi de la seq sur le quartet fort de al
           add ax,bx              ;envoi de l'adr du moteur sur quartet faible de al
           add al,08h             ;mise à 1 du bit de validation
           mov dx,porta           ;adr du port A sur dx
           out dx,ax              ;envoi du mot de commande sur port A
           sub al,08h             ;mise à zéro du bit de validation
           out dx,ax              ;envoi du mot de commande sur port A
           ret
motdw      endp
        ;
        ;
        PAGE
```

```
        ;
        ;*************************************************************************
        ;**********          PROCEDURE COMAND          ******************
        ;*************************************************************************
        ;
        ;
COMMENT */    Routine de commande de tous les moteurs simultannément
            Utilisée jusqu'à la mise à zéro de tous les comp relatifs
                              RELCO                                    /*

        ;
        ;
COMAND  PROC FAR
        ;
        ;
rak1    :  mov bx,0001h          ;adr du mot 1 dans bx
rak2    :  mov ax,relco[bx]      ;valeur du compt relatif du mot1 dans ax
           cmp ax,0000h          ;vérific si ce compt est à zéro
           jne rak5              ;sauter si différent de zéro
rak3    :  inc bx                ;si à zéro passer au moteur suivant
           cmp bx,0007h          ;test si dernier moteur atteint
           jne rak2              ;saut si non
rak4    :  dec bl                ;vérification si tous les moteurs sont
           cmp bl,00h            ;à zéro pour sortir de cette procédure
           je fin1               ;si oui sortir
           cmp relco[bx],0000h
           jne rak1              ;si non continuer l'éxécution
           jmp rak4
rak5    :  and ax,8000h          ;test du bit signe et voir si avance ou recul
           jz rak6               ;si à zéro sauter
           mov motbuf,bx         ;adr du moteur dans MOTBUF
           call motdw            ;appel de MOTDW
           inc relco[bx]         ;incrémentation de RELCO du moteur choisi
           jmp rak3              ;passage au moteur suivant
rak6    :  mov motbuf,bx         ;adr du moteur dans MOTBUF
           call motup            ;appel de MOTUP
           dec relco[bx]         ;décrémentation du compt relatif RELCO
           jmp rak3              ;passage au moteur suivant
fin1    :  ret
comand     endp
        ;
        PAGE
```

```
        ;*************************************************************************
        ;**********          PROCEDURE RESTOR          **********
        ;*************************************************************************
        ;
        ;
COMMENT */  Remise à la position zéro du bras en utilisant les
            informations contenues dans les compteurs absolus de
            chaque moteur                                        /*
        ;
        ;
RESTOR  PROC FAR
        ;
        ;
dak1    : mov bx,0001h          ;adr du moteur 1 dans bx
dak2    : mov ax,absco[bx]      ;val du compt absolu du moteur 1 dans ax
          cmp ax,0000h          ;vérification si ce compteur est à zéro
          jne dak5              ;sauter si différent de zéro
dak3    : inc bl                ;si à zéro passer au moteur suivant
          cmp bl,07h            ;test si dernier moteur atteint
          jne dak2              ;saut si non
dak4    : dec bl                ;verification si tous les compt absolus sont
          cmp bl,00h            ;à zéro pour sortir de cette procédure
          je fin2               ;si oui sortir
          cmp absco[bx],0000h
          jne dak1              ;si non continuer l'éxécution
          jmp dak4
dak5    : and ax,8000h          ;test du bit signe et voir si avance ou recul
          jz dak6               ;si à zéro sauter
          mov motbuf,bx         ;adr du moteur dans MOTBUF
          call motdw            ;appel de MOTDW
          dec absco[bx]         ;décrémentation du compteur absolu
          jmp dak3              ;passage au moteur suivant
dak6    : mov motbuf,bx         ;adr du moteur dans MOTBUF
          call motup            ;appel de MOTUP
          inc absco[bx]         ;incrémentation du compt absolu
          jmp dak3              ;passage au moteur suivant
fin2    : ret
restor    endp
        ;
        PAGE
```

```
        ;*************************************************************************
        ;*********          PROCEDURE MOUIN          ********************
        ;*************************************************************************
        ;
        ;
COMMENT */ Mouvement du bras par certaines touches du clavier
          effectue la reconnaissance du moteur à mouvoir et du sens
          de son déplacement et fait ensuite appel aux procédures
          MOTUP OU MOTDW                                          /*
        ;
        ;
        ;
MOUIN PROC FAR
        ;
        ;
DEBUT   : mov ah,08h            ;fnt de lecture du clavier sans écho
          int 21h
          cmp al,m1up           ;comparaison du la touche pressée à Q
          jne piste1            ;sauter si non
          mov (motbuf),0001h    ;adr du moteur 1 dans MOTBUF
          call motup            ;appel de MOTUP
          jmp debut             ;retour pour lecture de la touche suiv
piste1  : cmp al,m1dw           ;comparaison de la touche pressée à q
          jne piste2            ;sauter si non
          mov (motbuf),0001h    ;adr du moteur 1 dans MOTBUF
          call motdw            ;appel de MOTDW
          jmp debut             ;retour pour lecture de la touche suivante
piste2  : cmp al,m2up
          jne piste3
          mov (motbuf),0002h
          call motup
          jmp debut
piste3  : cmp al,m2dw
          jne piste4
          mov (motbuf),0002h
          call motdw
          jmp debut
piste4  : cmp al,m3up
          jne piste5
          mov (motbuf),0003h
          call motup
          jmp debut
```

```
piste5  : cmp al,m3dw
          jne piste6
          mov (motbuf),0003h
          call motdw
          jmp debut
piste6  : cmp al,m4up              ;mouvement de la pince necessitant
          jne piste7               ;l'emploi de deux moteurs a chaque
          mov (motbuf),0004h       ;fois ( voir ch II )
          call motup
          mov (motbuf),0005h
          call motdw
          jmp debut
piste7  : cmp al,m4dw
          jne piste8
          mov (motbuf),0004h
          call motdw
          mov (motbuf),0005h
          call motup
          jmp debut
piste8  : cmp al,m5up
          jne piste9
          mov (motbuf),0004h
          call motup
          mov (motbuf),0005h
          call motup
          jmp debut
piste9  : cmp al,m5dw
          jne piste10
          mov (motbuf),0004h
          call motdw
          mov (motbuf),0005h
          call motdw
          jmp debut
piste10 : cmp al,m6up              ;fermeture de la pince
          jne piste11
          mov (motbuf),0006h
          call motup
          jmp debut
piste11 : cmp al,m6dw              ;ouverture de la pince
          jne piste12
          mov (motbuf),0006h
          call motdw
          jmp debut
piste12 : cmp al,sort
          je fin4
          jmp debut
 fin4   : ret
movin     endp
          ;
          PAGE
```

```
        ;*************************************************************************
        ;********          PROCEDURE RAZ1          **********************
        ;*************************************************************************
        ;
        ;
COMMENT */    Met tous les LATCHS des moteurs à zéro pour permettre
           leur positionnement manuel dans la position zéro voulu /*
        ;
        ;
RAZ1 PROC FAR
        ;
         mov al,00h
  rea  : inc al                  ;adr du moteur dans al
         add al,08h               ;constitution du mot de commande
                                 ;avec mise à zéro du quartet fort de al
         mov dx,porta             ;adr port A dans dx
         out dx,ax                ;envoi du mot de commande sur port A
         sub al,08h               ;mise à zéro du bit de validation
         out dx,ax                ;envoi du mot de commande sur port A
         cmp al,06h               ;test si dernier moteur atteint
         jne rea                  ;si non alors continuer
         ret
  raz1   endp
       ;
       PAGE
       ;
       ;*************************************************************************
       ;**********          PROCEDURE RAZ2          **********************
       ;*************************************************************************
       ;
       ;
COMMENT */    Initialise les LATCHS à la première séquence du tableau
                                SEQTAB                              /*
       ;
       ;
RAZ2 PROC FAR
       ;
       ;
         mov al,160               ;première val de SEQTAB dans quart fort de al
  mad  : inc al                   ;adr du moteur dans quartet faible de al
         add al,08h               ;mise à 1 du bit de validation
         mov dx,porta             ;adr du port A sur dx
         out dx,ax                ;envoi du mot de commande sur le port A
         sub al,08h               ;mise à zéro du bit de validation
         out dx,ax                ;envoi du mot de commande sur le port A
         cmp al,166               ;test si dernier moteur atteint
         jne mad                  ;si non continuer
         ret
  raz2   endp
Cseg     ends
```

DESCRIPTION DU PROGRAMME ROBOT.PAS :

Le programme ROBOT.PAS s'occupe de la gestion complète du robot ,au moyen de l'appel des routines bases décrites précedemment . Il est écrit en language TURBO PASCAL (version 5.0).Il est constitué de plusieurs procédures de service .

PROG1 : Réalise la remise à zéro manuelle du mini-robot en faisant appel aux procédures RAZ1 et RAZ2 .

PROG2 : Réalise le retour à la position zéro définie précedemment en faisant appel à la procédure RESTOR .

PROG3 : Réalise la commande du bras par clavier en faisant appel à la procédure MOVIN .

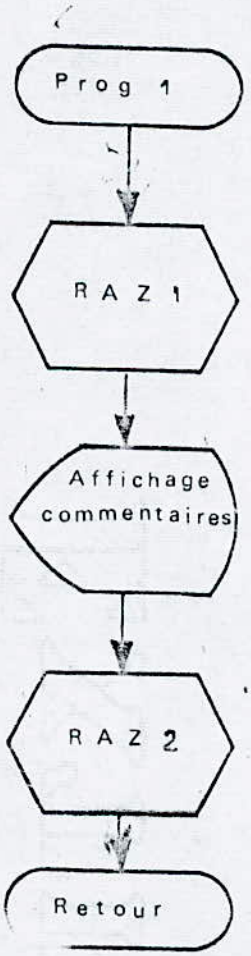
PROG4 : Réalise la lecture d'une séquence et l'enregistre dans un fichier .Elle fait appel à la procédure MOVIN .

PROG5 : Réalise l'éxecution automatique d'une séquence déjà enregistrée en faisant appel à la procédure COMMAND .

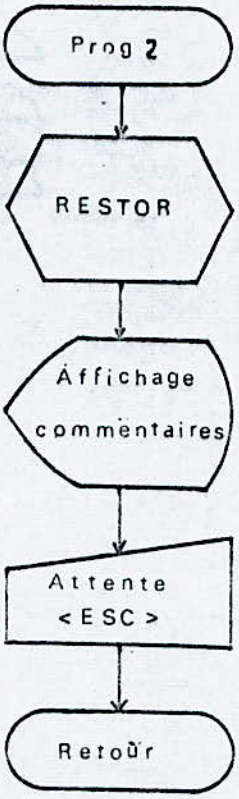
PROG6 : Réalise l'édition des pas d'une séquence et permet leur modification sans devoir recourir à une

nouvelle étude de celle-ci .

On présentera ci-aprés les organigrammes de chacune de ces procédures ainsi que le listing complet de ce programme :

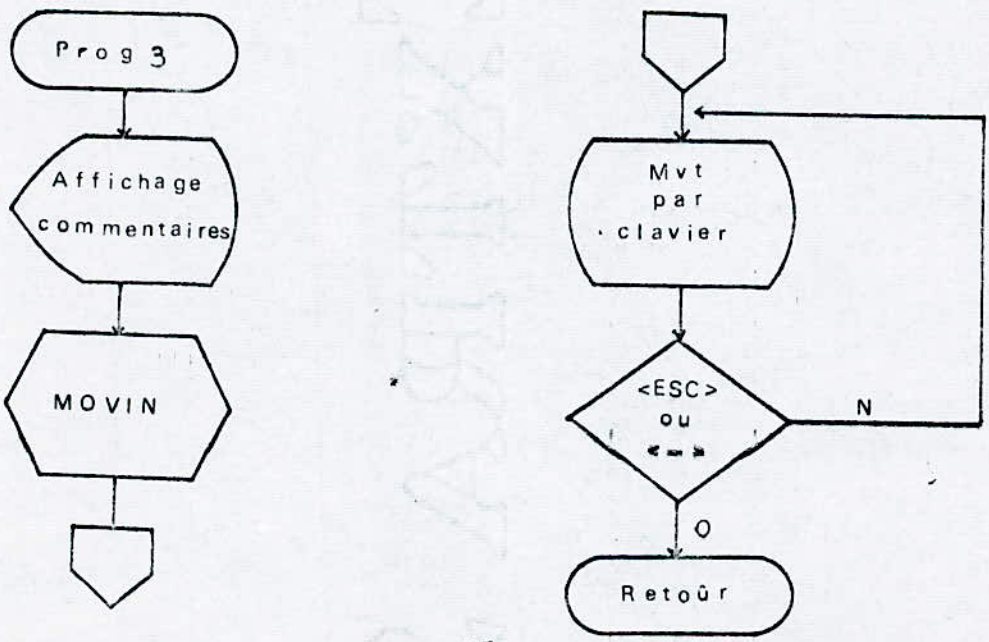


Prog 1

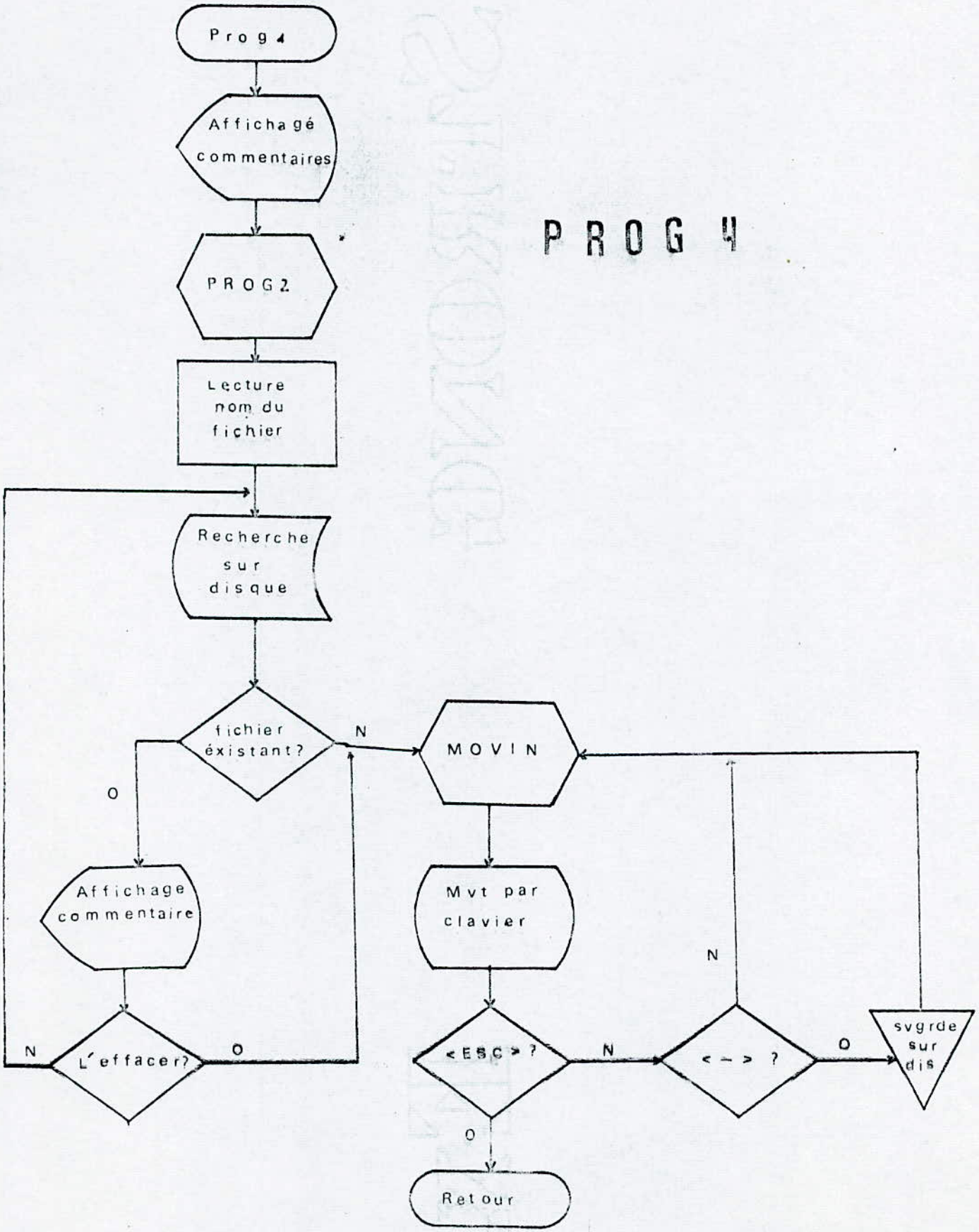

Prog 2
RESTOR
Affichage
commentaires
Attente
< ESC >
Retoûr

Prog 2

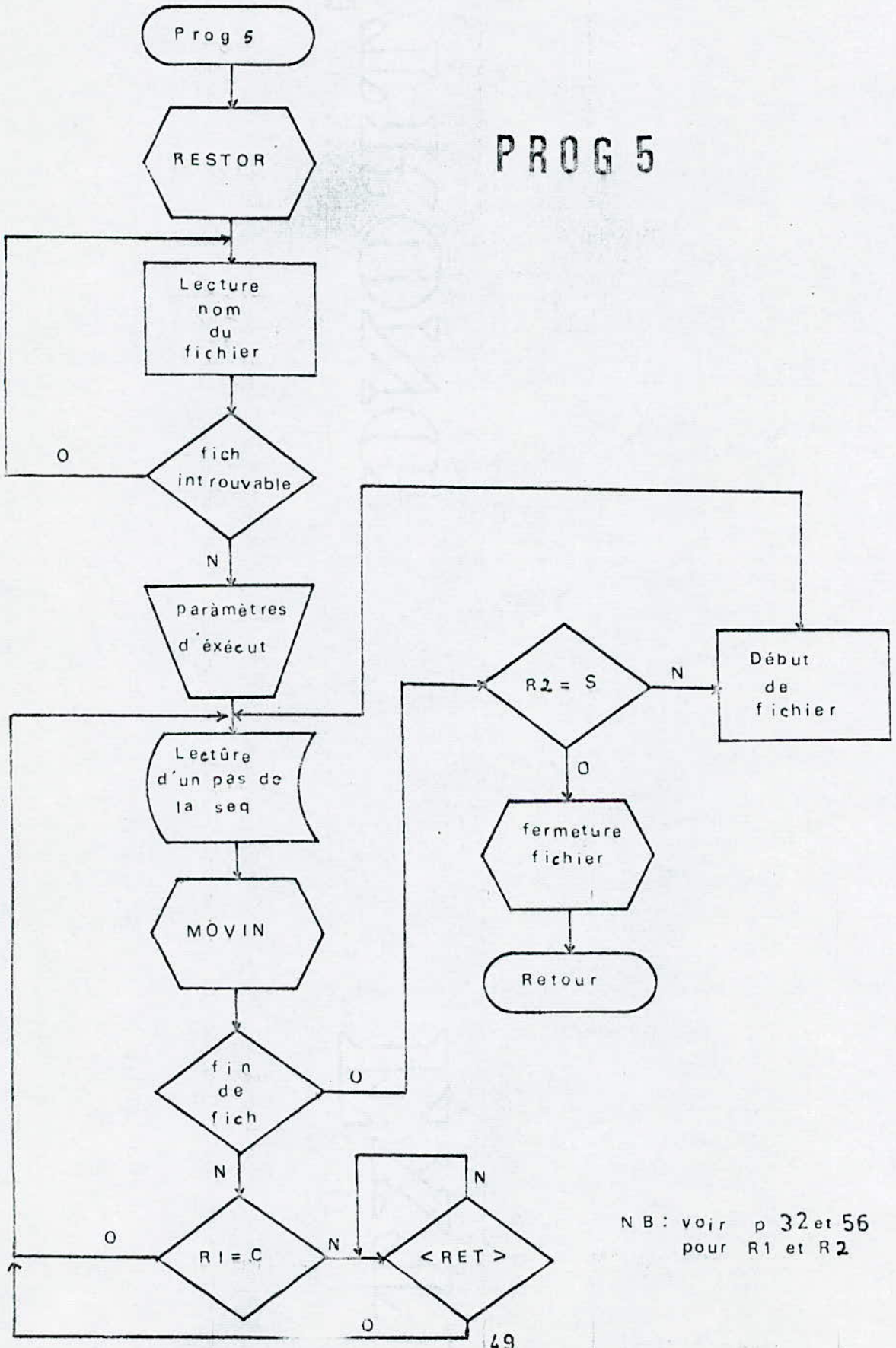
Prog 3


Prog 3
Affichage
commentaires
MOVIN
Mvt
par
clavier
<ESC>
ou
N
O
Retoûr

PROG 4

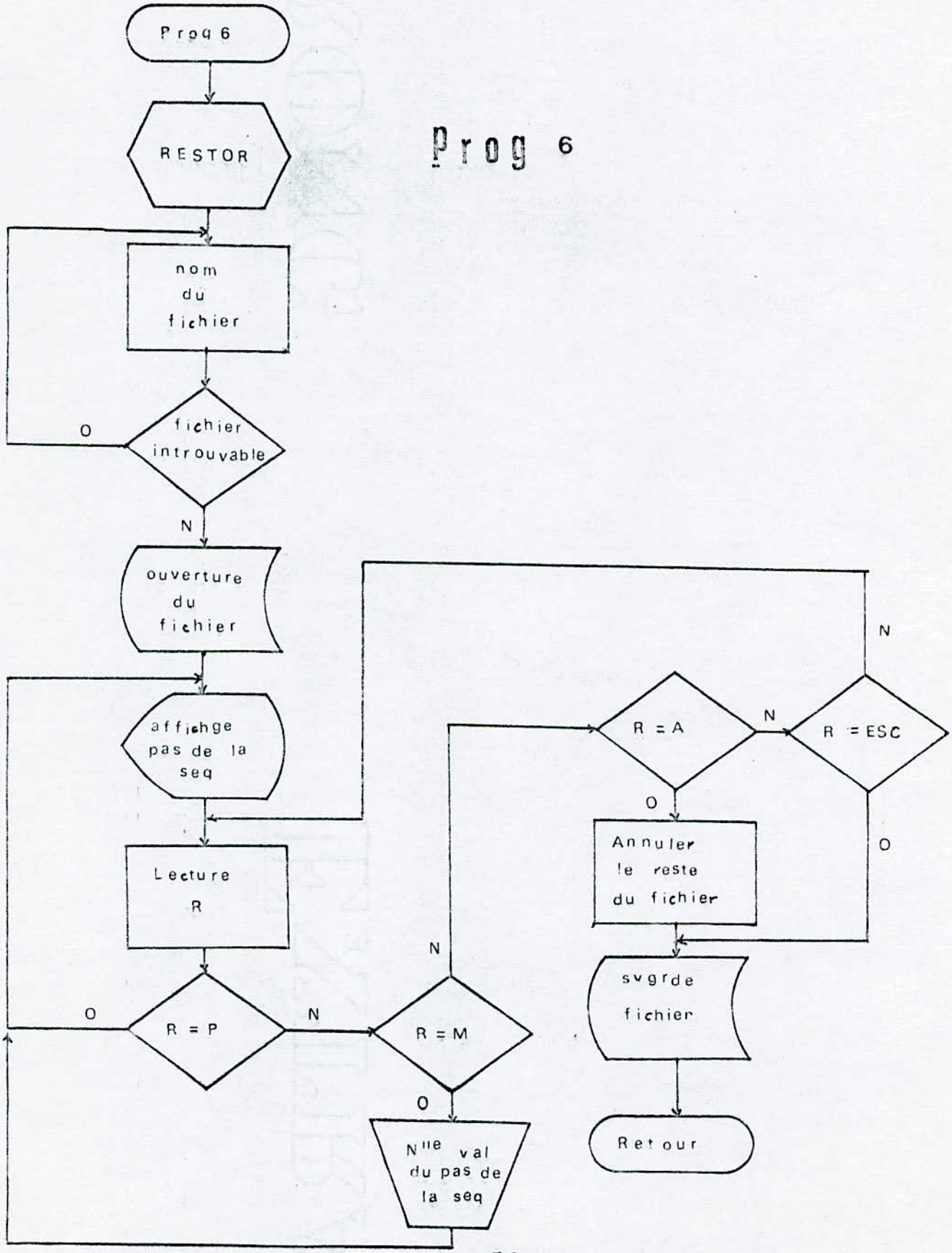

Prog 4
Affichage commentaires
PROG2
Lecture nom du fichier
Recherche sur disque
fichier éxistant?
N
O
MOVIN
Affichage commentaire
Mvt par clavier
L'effacer?
N
O
<ESC> ?
N
<—> ?
N
O
svgrde sur dis
O
Retour

PROG 5



NB : voir p 32 et 56 pour R1 et R2

Prog 6


Prog 6
RESTOR
nom du fichier
fichier introuvable
O
N
ouverture du fichier
affichge pas de la seq
Lecture R
R = P
O
N
R = M
N
O
N'ie val du pas de la seq
R = A
N
R = ESC
N
O
O
Annuler le reste du fichier
svgrde fichier
Retour

```
{*************************************************************************}
{*********                                                      ************}
{*********          PROGRAMME DE GESTION DE LA CARTE BASE       ************}
{*********             SECTION EN TURBO PASCAL 5.0              ************}
{*********                                                      ************}
{*************************************************************************}
PROGRAM essai1;
{$D+}
USES   crt,dos;
LABEL  ram;
TYPE   tab1=array[1..8] of byte;
  tab2=array[1..6] of integer;
  enrgtr=record
       m1,m2,m3,m4,m5,m6:integer;
      end;
VAR    x,i,x1,c,w,m,pacom:integer;
       arch:file of enrgtr;
       y:enrgtr;
       nom:string[16];
       absco,relco,vittab,motadr,ctpos1:tab2;
       seqtab,seq1tab:tab1;
       r:pathstr;
       r1,r2,t:char;
{***********  DECLARATION DES ROUTINES EN ASSEMBLEUR ********************}
PROCEDURE motup;external;
PROCEDURE motdw;external;
PROCEDURE comand;external;
PROCEDURE restor;external;
PROCEDURE movin;external;
PROCEDURE raz1;external;
PROCEDURE raz2;external;
{$L rout1.obj}
```

```
{$L rout1.obj}
  {**************  INITIALISATION DES TABLEAUX ************************}
PROCEDURE initialisation;
 begin
    pacom:=0;
    seqtab[1]:=160;seqtab[2]:=128;seqtab[3]:=144;
    seqtab[4]:=16;seqtab[5]:=80;seqtab[6]:=64;seqtab[7]:=96;
    seqtab[8]:=32;
     for i:=1 to 6 do begin
       absco[i]:=0;
       relco[i]:=0;
       ctpos1[i]:=i;
       end;
 end;
{************** PROCEDURE DE PRESENTATION DES FNTS DES TOUCHES ********}

PROCEDURE presentation;
  begin
     window(20,10,70,17);textbackground(0);clrscr;
     window(10,20,70,25);textbackground(0);clrscr;
     window(0,0,80,25);textbackground(0);
     clrscr;
     window(4,7,78,9);
     gotoxy(5,5);
     writeln('Le mini robot est controllé par le clavier avec les touches suivant
     window(8,10,72,22);
     textbackground(9);textcolor(14);
     highvideo;
     writeln('<Q>=Rot de la base en SAM      <q>=mouvement opposé');
     writeln('<W>=Elevation de l'épaule      <w>=mouvement opposé');
     writeln('<E>=Elevation  du bras         <e>=mouvement opposé');
     writeln('<R>=Elevation de la pince      <r>=mouvement opposé');
     writeln('<T>=Rot pince en SAM           <t>=mouvement opposé');
     writeln('<Y>=Fermeture de la pince      <y>=mouvement opposé');
     writeln('                                                  ');
     writeln;
     writeln('   APPUYER SUR <ESC> POUR SORTIR');
  end;
```

```
{************  PROCEDURE D'INITIALISATION DES TABLEAUX  *****************}
PROCEDURE INITIA ;
begin
   pacom:=0;
   seqtab[0]:=160;seqtab[1]:=128;seqtab[2]:=144;seqtab[3]:=16;
   seqtab[4]:=80;seqtab[5]:=64;seqtab[6]:=96;seqtab[7]:=32;
   for i:=1 to 6 do
    begin
      absco[i]:=1;
      relco[i]:=0;
      ctpos[i]:=1;
    end;
end;
{************  PROCEDURE DE REMISE A ZERO MANUEL   **********************}
PROCEDURE prog1 ;
LABEL lire;
begin
    clrscr;
    textbackground(0);
    textcolor(14);
    highvideo;
    gotoxy(25,6);
    raz1;
    writeln(' REMISE A ZERO MANUELLE ');
lire:  textbackground(4);Textcolor(128);
    gotoxy(10,20);
    writeln(' METTRE LE MINI-ROBOT A LA POSITION ZERO ');gotoxy(10,21);
    writeln('        ENSUITE APPUTER SUR <ESC>        ');textbackground(0);
    textcolor(0);
    t:=readkey;
    if t<> chr(27) then goto lire;
    raz2;
    end;
```

```
gotoxy(9,17);write('ENTRER NOM DU FICHIER S.V.P : ');
textbackground(9);textcolor(14);readln(nom);
textbackground(4);textcolor(14);
r:=fsearch(nom,GetEnv('path'));
if r='' then
  begin
    br1 : textbackground(0);clrscr;textcolor(14);
                assign(arch,nom);
                rewrite(arch);
                gotoxy(15,3);
                write(' LECTURE DE LA SEQUENCE <<  ',nom,'  >> ');
                gotoxy(5,7);textcolor(14);
                writeln(' La séquence est constituée de pas produisant le déplac
                presentation;
                gotoxy(10,23);textbackground(9);x1:=0;
                writeln(' APPUYER SUR (-) A LA FIN DE CHAQUE ETAPE  ');writeln;
           br5: repeat
                x1:=x1+1;
                window(54,22,70,24);textbackground(14);clrscr;textcolor(128);
                gotoxy(3,3);writeln(' PAS N° : ',x1);
                textbackground(0);writeln;
                fori:=1 to 6 do relco[i]:=0;
                movin;
                with y do
                 begin
                   m1:=relco[1];m2=relco[2];m3:=relco[3];
                   m4:=relco[4];m5:=relco[5];m6:=relco[6];
                 end;
                 write(arch,y);
                 until (pacom=27);
                 close(arch);
                 clrscr;
                 end
                 else
                 br3 : begin
                   gotoxy(6,19);textbackground(0);textcolor(14);
                   write('Fichier existant déja l''effacer O/N : ');
      textbackground(9);readln(r1);
      case r1 of
       chr(79),chr(111):goto br1;
       chr(78),chr(110):goto br2;
      else goto br3;
    end;
  end;
   end;
```

```
{************ PROCEDURE D'EXECUTION AUTO D'UNE SEQUENCE ****************}
PROCEDURE prog5;
LABEL cl1,cl2,cl3,cl4,cl5,cl6,cl10,cl11;
    begin
  clrscr;textbackground(0);textcolor(14);
  gotoxy(25,6);
  writeln('EXECUTION AUTO D''UNE SEQUENCE');
  restor;
    cl1:gotoxy(15,15);textbackground(9);textcolor(14);
  write('Entrer le nom de la sequence S.V.P :');readln(nom);
  r:=fsearch(nom,GetEnv('path'));
  if r='' then
    begin
      gotoxy(0,19);textbackground(4);
      textcolor(128);
      writeln('Fichier introuvable !! ');textcolor(0);
            delay(2000);
            gotoxy(0,19);writeln;
      goto cl1;
  end
        else
         begin
         assign(arch,nom);
  gotoxy(1,15);textbackground(0);textcolor(0);delline;
  gotoxy(15,15);writeln;gotoxy(15,15);textbackground(9);textcolor(14);
  writeln('Paramètres d''éxécution : ');
    cl2:gotoxy(0,17);writeln;
  gotoxy(12,17);
  textbackground(4);textcolor(15);
  write('Mouvement continu <C>  Mouvement par pas <P> ');
  readln(r1);
        case r1 of
         chr(80),chr(112),chr(67),chr(99):goto cl3;
         else goto cl2;
        end;
    cl3:gotoxy(12,18);
  write('Une seule éxécution <S>  Exécution cyclique <C> ');
  readln(r2);
        case r2 of
          chr(67),chr(99);chr(83),chr(115):goto cl6;
          else goto cl3;
        end;
    cl6:reset(arch);
  while not eof do
    begin
      if upcase(r1)='P' then
        begin
    gotoxy(20,22);textbackground(0);
    writeln('Appuyer sur <RET> pour passer au pas suivant');
                  y);
```

```
      with y do
    begin
       relco[1]:=m1; relco[2]:=m2;relco[3]:=m3;
       relco[4]:=m4; relco[5]:=m5; relco[6]:=m6;
    end;
    end;
    movin;
    if upcase(r1)='C' then goto cl4;
     cl10:t:=readkey;
         case t of
                chr(27),chr(13): goto cl5;
               end;
               goto cl10;
         cl5:if t=chr(27) then
     begin
       close(arch);
       goto cl11;
     end;
         goto cl4;
     end;
     if upcase(r2)='S' then
       begin
         close(arch);
         gotoxy(20,23);textbackground(14);textcolor(128);
         writeln('Fin d''éxécution de la séquence ');
         gotoxy(21,23);writeln(nom);
         delay(3000);
               end;
               cl11:end;
               end;
           end;
end;
{************* PROCEDURE D'EDIT D'UNE SEQUENCE *************************}
PROCEDURE prog6;
LABEL dl1,dl2,dl3;
  begin
    clrscr;textbackground(0);textcolor(14);gotoxy(25,6);
    writeln('EDIT D''UNE SEQUENCE ');
    restor;initia;
  dl1:gotoxy(15,15);textbackground(9);textcolor(14);
      write('Entrer le nom de la séquence S.V.P :');readln(nom);
      gotoxy(15,15);delline;
      r:=fsearch(nom,GetEnv('path'));
      if r='' then
   begin
```

```
     gotoxy(1,19);textbackground(4);
     writeln;
     textcolor(128);
     gotoxy(15,20);writeln('Fichier introuvable !! ');
     delay(4000);
     textbackground(0);textcolor(14);delline;
     goto dl1;
end
       else
         begin
          gotoxy(15,9);
    textbackground(14);textcolor(9);
    writeln(' SEQUENCE EDITEE : <<  ',nom,'  >>  ');
          assign(arch,nom);
    reset(arch);
    gotoxy(1,22);textbackground(4);textcolor(15);
          writeln;writeln;writeln;
          writ(chr(213));
          for i:=2 to 79 do write(chr(205));
          writeln(chr(184));
    gotoxy(25,23);writeln(' Options d''édition :');
    gotoxy(1,24);textcolor(14);write('P');textcolor(15);write('asser au pas sui\
          textcolor(14);write('  M');textcolor(15);write('odifier');
    textcolor(14);write('  A');textcolor(15);write('nnuler le reste du fichier '
          gotoxy(80,24);write(chr(179));
          gotoxy(1,25));textcolor(14);
          write('<ESC>');textcolor(15);write(' retour au menu principal  ');
          gotoxy(80,25);write(chr(179));write(chr(192));
          for i:=2 to 79 do write(chr(196);
          write(chr(213));
          gotoxy(7,13);
          textbackground(9);textcolor(14);
    writeln('PAS   MOT.1   MOT.2   MOT.3   MOT.4   MOT.5   MOT.6 ');
    while not eof do
    begin
      dl3:c:=filepos(arch);
      read(arch,y);i:=10;
      gotoxy(i,15);write(c);
            with y do
            begin
             m:=m1;i:=i+7;gotoxy(i,15);write(m);
             m:=m2;i:=i+7;gotoxy(i,15);write(m);
             m:=m3;i:=i+7;gotoxy(i,15);write(m);
             m:=m4;i:=i+7;gotoxy(i,15);write(m);
             m:=m5;i:=i+7;gotoxy(i,15);write(m);
             m:=m6;i:=i+7;gotoxy(i,15);write(m);
end;
dl2:t:=readkey;
    if t<>'p' {and t<>'m' and t<>'a' and t<>chr(27)} then goto dl2;
    if t='p' then goto dl3;
    if t='m' then
```

```
{*************************************************************************}
{***********        DEBUT DU PROGRAMME PRINCIPAL        ***************}
{*************************************************************************}
begin
    initialisation;
    w:=2;
    while w=2  do
    begin
     assigncrt(input);reset(input);
    assigncrt(output);rewrite(output);
    textbackground(0);
    clrscr;

    textcolor(14);
    gotoxy(25,6);
    highvideo;
    writeln('   MENU PRINCIPAL ');
    textcolor(15);
    gotoxy(15,10);
    writeln('  1- RAZ MANUELLE '); gotoxy(15,11);
    writeln('  2- RAZ AUTO'); gotoxy(15,12);
    writeln('  3- ESSAI PAR PAS'); gotoxy(15,13);
    writeln('  4- ETUDE D''UNE SEQUENCE'); gotoxy(15,14);
    writeln('  5- EXECUTION D''UNE SEQUENCE'); gotoxy(15,15);
    writeln('  6- EDIT D''UNE SEQUENCE '); gotoxy(15,16);
    WRITELN('  7- FIN ');
    highvideo;
    gotoxy(10,23);
    textcolor(14);
    writeln;
    write('    Entrer votre selection S V P :');readln(x) ;
ram  :case x of
      1:prog1;
      2:prog2;
      3:prog3;
      4:prog4;
      5:prog5;
      6:prog6;
      7:w:=3
    else
      begin
  gotoxy(10,25);
  textcolor(128);
  sound(10);textbackground(14);
  write('ATTENTION SELECTION 1--6!!');
  delay(4000);
  nosound;
  delline;textbackground(0);textcolor(14);
  gotoxy(10,24);
  delline;textcolor(14);
  write('Entrer votre selection S V P : ');readln(x);
  goto ram;
      end;
      clrscr;
   end;
end;
```

CHAPITRE VI

Conclusion

CONCLUSION :

Ce mini-robot est un exemple à une échelle réduite comme dimension, d'un bras articulé typique de robots industriels, pouvant être commandé par un calculateur.

Il peut ainsi servir comme milieu pilote pour l'évaluation et l'analyse économique de l'installation de robots industriels dans les entreprises.

Il peut aussi servir pour le développement d'applications dans le domaine de l'intelligence artificielle (commande vocale, vision artificielle,...).

Notre travail peut être considérer comme une plate-forme offrant toutes les possibilités d'étude et d'initiation à ces applications et à ces techniques.

CHAPITRE VII

Annexes

Quad 3-State Buffer

TYPE	TYPICAL PROPAGATION DELAY	TYPICAL SUPPLY CURRENT (Total)
74125	10ns	32mA
74LS125	8ns	11mA
74126	10ns	36mA
74LS126	9ns	12mA

ORDERING CODE

PACKAGES	COMMERCIAL RANGES V_{CC} = 5V ± 5%; T_A = 0°C to +70°C	MILITARY RANGES V_{CC} = 5V ± 10%; T_A = −55°C to +125°C
Plastic DIP	N74125N • N74LS125N N74126N • N74LS126N	
Ceramic DIP	N74125F • N74LS125F N74126F • N74LS126F	S54LS125F S54126F • S54LS126F
Flatpack		S54LS125W S54126W • S54LS126W

FUNCTION TABLE '125

INPUTS		OUTPUT
C	A	Y
L	L	L
L	H	H
H	X	(Z)

FUNCTION TABLE '126

INPUTS		OUTPUT
C	A	Y
H	L	L
H	H	H
L	X	(Z)

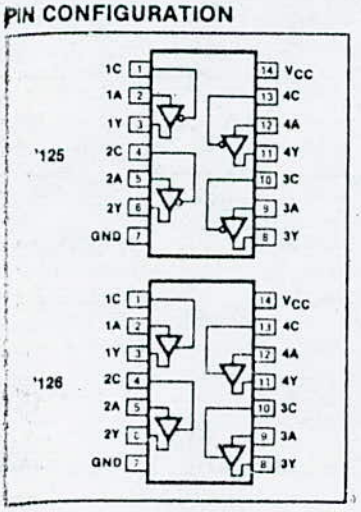
H = HIGH voltage level
L = LOW voltage level
X = Don't care
(Z) = HIGH impedance (off)

INPUT AND OUTPUT LOADING AND FAN-OUT TABLE

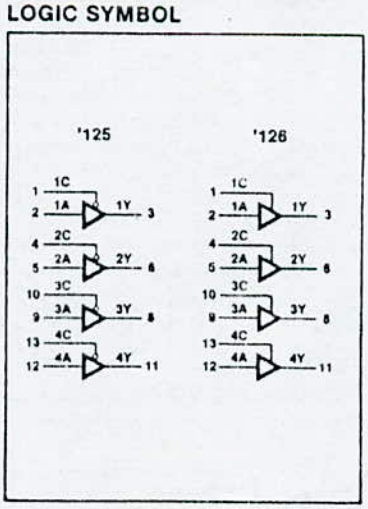
PINS	DESCRIPTION	54/74	54/74LS
All	Inputs	1ul	1LSul
All	Outputs	10ul	30LSul

NOTE
Where a 54/74 unit load (ul) is understood to be 40μA I_{IH} and −1.6mA I_{IL}, and a 54/74LS unit load (LSul) is 20μA I_{IH} and −0.4mA I_{IL}.

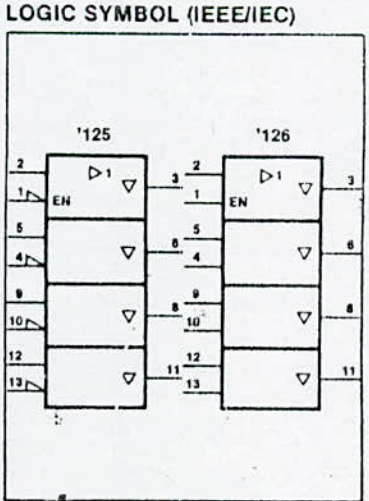
PIN CONFIGURATION



LOGIC SYMBOL



LOGIC SYMBOL (IEEE/IEC)



MULTIVIBRATOR

54/74123

Dual Retriggerable Monostable Multivibrator

- DC triggered from active HIGH or active LOW inputs
- Retriggerable for very long pulses—up to 100% duty cycle
- Direct reset terminates output pulse
- Compensated for V_{CC} and temperature variations

TYPE	TYPICAL PROPAGATION DELAY	TYPICAL SUPPLY CURRENT (Total)
74123	24ns	46mA

ORDERING CODE

PACKAGES	COMMERCIAL RANGES V_{CC} = 5V ±5%; T_A = 0°C to +70°C	MILITARY RANGES V_{CC} = 5V ±10%; T_A = −55°C to +125°C
Plastic DIP	N74123N	
Ceramic DIP	N74123F	S54123F
Flatpack		S54123W

DESCRIPTION

The '123 is a dual retriggerable monostable multivibrator with output pulse width control by three methods. The basic pulse time is programmed by selection of external resistance (R_{ext}) and capacitance (C_{ext}) values. Once triggered, the basic pulse width may be extended by retriggering the gated active LOW going edge input ($\overline{A}$) or the active HIGH going edge input (B), or be reduced by use of the overriding active LOW reset.

The basic output pulse width is essentially determined by the values of external capacitance and timing resistance. For pulse widths when $C_{ext} \leq 1000pF$, see Figure A.

When $C_{ext} > 1000pF$, the output pulse width is defined as:

$$t_W = 0.28\ R_{ext}/C_{ext}\ (1 + \frac{0.7}{R_{ext}})$$

The external resistance and capacitance are normally connected as shown in Figure B. If an electrolytic capacitor is to be used with an inverse voltage rating of less than 1V then Figure C should be used. (Inverse voltage rating of an electrolytic is normally specified at 5% of the forward voltage rating.) If the inverse voltage rating is 1V or more (this includes a 100% safety margin) then Figure B can be used. Note that if Figure C is used the timing equations change as follows:

$$t_W \cong 0.25\ R_{ext}/C_{ext}\ (1 + \frac{0.7}{R_{ext}})$$

FUNCTION TABLE

INPUTS			OUTPUTS	
R_D	$\overline{A}$	B	Q	$\overline{Q}$
L	X	X	L	H
X	H	X	L	H
X	X	L	L	H
H	L	↑	⎍	⎍
H	↓	H	⎍	⎍
↑	L	H	⎍	⎍

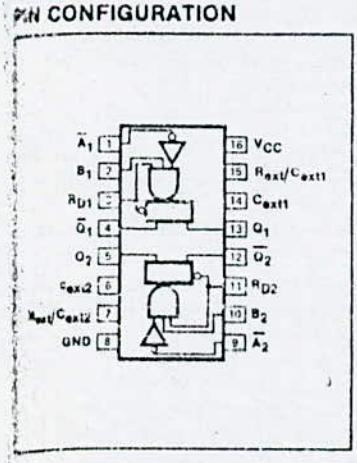
H = HIGH voltage level
L = LOW voltage level
X = Don't care
↑ = LOW-to-HIGH transition
↓ = HIGH-to-LOW transition
⎍ = One HIGH-level pulse
⎍ = One LOW-level pulse

INPUT AND OUTPUT LOADING AND FAN-OUT TABLE

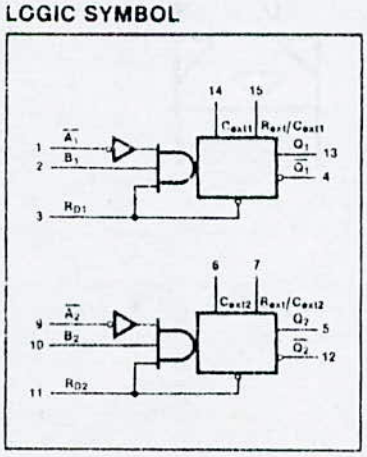
PINS	DESCRIPTION	54/74
$\overline{A}$, B	Inputs	1ul
R_D	Input	2ul
Q, $\overline{Q}$	Outputs	10ul

NOTE
A 54/74 unit load (ul) is understood to be 40μA I_{IH} and −1.6mA I_{IL}.

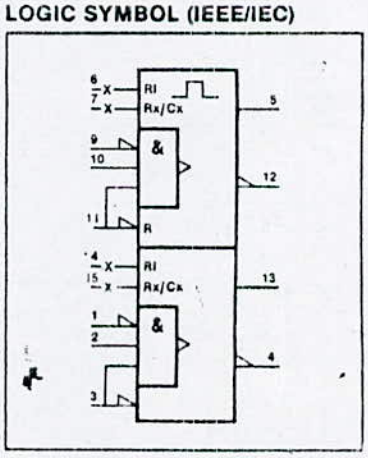
PIN CONFIGURATION



LOGIC SYMBOL



LOGIC SYMBOL (IEEE/IEC)



DECODERS/DEMULTIPLEXERS

54/74LS138, S138

1-Of-8 Decoder/Demultiplexer

- **Demultiplexing capability**
- **Multiple input enable for easy expansion**
- **Ideal for memory chip select decoding**
- **Direct replacement for Intel 3205**

TYPE	TYPICAL PROPAGATION DELAY	TYPICAL SUPPLY CURRENT (Total)
74LS138	20ns	6.3mA
74S138	7ns	49mA

DESCRIPTION

The '138 decoder accepts three binary weighted inputs (A_0, A_1, A_2) and when enabled, provides eight mutually exclusive, active LOW outputs ($\overline{0}$-$\overline{7}$). The device features three Enable inputs: two active LOW ($\overline{E}_1$, $\overline{E}_2$) and one active HIGH (E_3). Every output will be HIGH unless $\overline{E}_1$ and $\overline{E}_2$ are LOW and E_3 is HIGH. This multiple enable function allows easy parallel expansion of the device to a 1-of-32 (5 lines to 32 lines) decoder with just four '138s and one inverter.

The device can be used as an eight output demultiplexer by using one of the active LOW Enable inputs as the Data input and the remaining Enable inputs as strobes. Enable inputs not used must be permanently tied to their appropriate active HIGH or active LOW state.

ORDERING CODE

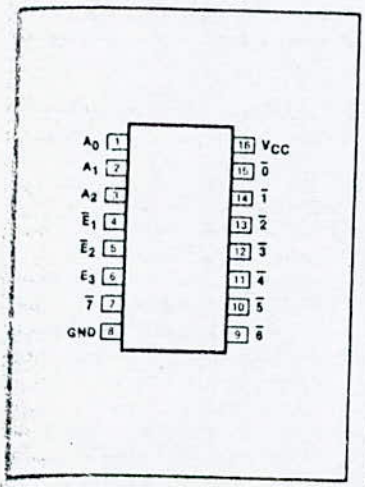
PACKAGES	COMMERCIAL RANGES V_{CC} = 5V ± 5%; T_A = 0°C to +70°C	MILITARY RANGES V_{CC} = 5V ± 10%; T_A = −55°C to +125°C
Plastic DIP	N74S138N • N74LS138N	
Ceramic DIP	N74S138F • N74LS138F	S54S138F • S54LS138F
Flatpack		S54S138W • S54LS138W

INPUT AND OUTPUT LOADING AND FAN-OUT TABLE

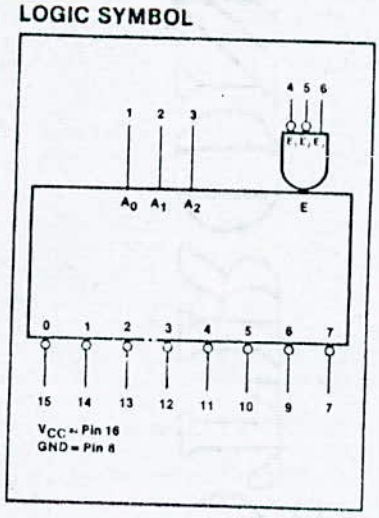
PINS	DESCRIPTION	54/74S	54/74LS
All	Inputs	1Sul	1LSul
All	Outputs	10Sul	10LSul

NOTE
Where a 54/74S unit load (Sul) is 50μA I_{IH} and −2.0mA I_{IL}, and a 54/74LS unit load (LSul) is 20μA I_{IH} and −0.4mA I_{IL}.

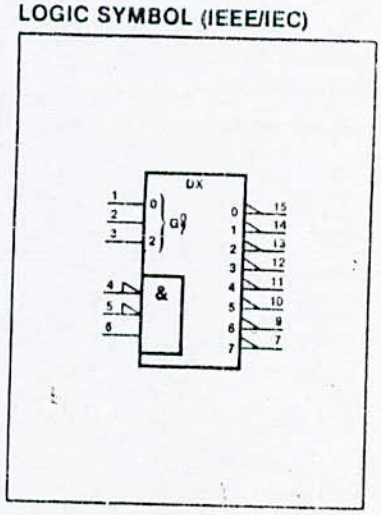
PIN CONFIGURATION



LOGIC SYMBOL



LOGIC SYMBOL (IEEE/IEC)



FLIP-FLOPS

54/74175, LS175, S175

Quad D Flip-Flop

- Four edge-triggered D flip-flops
- Three speed-power ranges available
- Buffered common clock
- Buffered, asynchronous Master Reset

TYPE	TYPICAL f_{MAX}	TYPICAL SUPPLY CURRENT (Total)
74175	35MHz	30mA
74LS175	40MHz	11mA
74S175	110MHz	60mA

DESCRIPTION

The '175 is a quad, edge-triggered D-type flip-flop with individual D inputs and both Q and $\overline{Q}$ outputs. The common buffered Clock (CP) and Master Reset ($\overline{MR}$) inputs load and reset (clear) all flip-flops simultaneously.

The register is fully edge triggered. The state of each D input, one setup time before the LOW-to-HIGH clock transition, is transferred to the corresponding flip-flop's Q output.

All Q outputs will be forced LOW independently of Clock or Data inputs by a LOW voltage level on the $\overline{MR}$ input. The device is useful for applications where both true and complement outputs are required, and the Clock and Master Reset are common to all storage elements.

ORDERING CODE

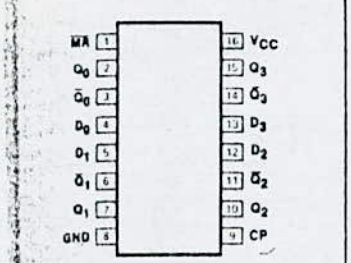
PACKAGES	COMMERCIAL RANGES V_{CC} = 5V ±5%; T_A = 0°C to +70°C	MILITARY RANGES V_{CC} = 5V ±10%; T_A = −55°C to +125°C
Plastic DIP	N74175N • N74LS175N N74S175N	
Ceramic DIP	N74175F • N74LS175F N74S175F	S54175F • S54LS175F
Flatpack		S54175W • S54LS175W

INPUT AND OUTPUT LOADING AND FAN-OUT TABLE

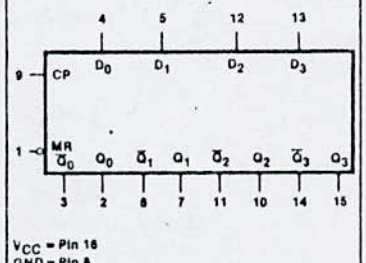
PINS	DESCRIPTION	54/74	54/74S	54/74LS
All	Inputs	1ul	1Sul	1LSul
All	Outputs	10ul	10Sul	10LSul

NOTE
Where a 54/74 unit load (ul) is understood to be 40μA I_{IH} and −1.6mA I_{IL}, a 54/74S unit load (Sul) is 50μA I_{IH} and −2.0mA I_{IL}, and a 54/74LS unit load (LSul) is 20μA I_{IH} and −0.4mA I_{IL}.

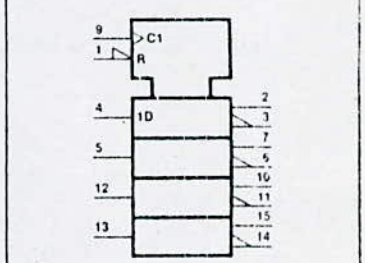
PIN CONFIGURATION

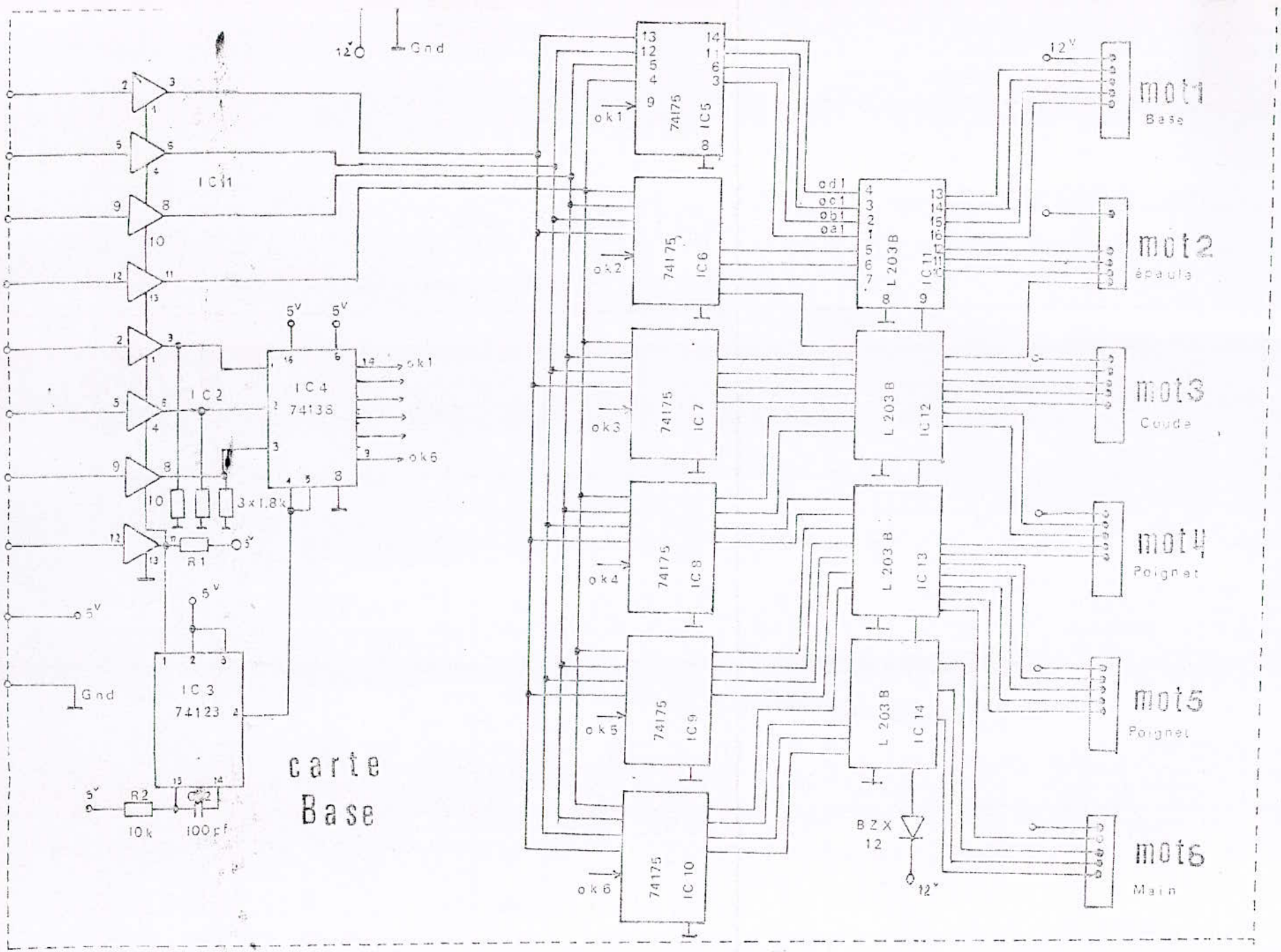


LOGIC SYMBOL



LOGIC SYMBOL (IEEE/IEC)




12V
Gnd
IC11
IC2
IC4
74138
IC3
74123
R1
3x1.8k
R2
10k
C2
100pf
5V
ok1
ok2
ok3
ok4
ok5
ok6
74175
IC5
IC6
IC7
IC8
IC9
IC10
L 203 B
IC11
IC12
IC13
IC14
BZX
12
mot1
Base
mot2
épaule
mot3
Coude
mot4
Poignet
mot5
Poignet
mot6
Main
carte
Base

CHAPITRE VIII.

Bibliographie

BIBLIOGRAPHIE :

1 INTRODUCTION A LA ROBOTIQUE T2

Pierre lopez et Jean Numa Fulc

ed TESTS 1984

2 INDUSTRIELS ROBOTS

(Computer interfacing and control)

Wesley E.Snyder

ed PRENTICE HALL 1985

3 MICROPROCESSEURS 16 BITS

M.Aumiaux

ed MASSON

4 LES MOTEURS PAS A PAS

Jean Jacquin

ed DUNOD TECHNIQUE BORDAS 1974

5 CONTRIBUTION A LA COMMANDE DYNAMIQUE D'UN MANIPULATEUR EQUIPE DE MOTEUR PAS A PAS

Thèse de docteur ingenieur -NANCY < fev 1984 >

6 CONTRIBUTION A LA REALISATION ,A L'ALIMENTATION ET A LA COMMANDE DES MOTEURS PAS A PAS

C.Coeldel

Thèse de doctorat CINP de LORAINE MARS 1984)

7 TURBO PASCAL 5.0

(Manuel de référence)

Par BORLAND 1984