ROBOT STÄUBLI RX 90

Jean-Louis Boimond
Université Angers

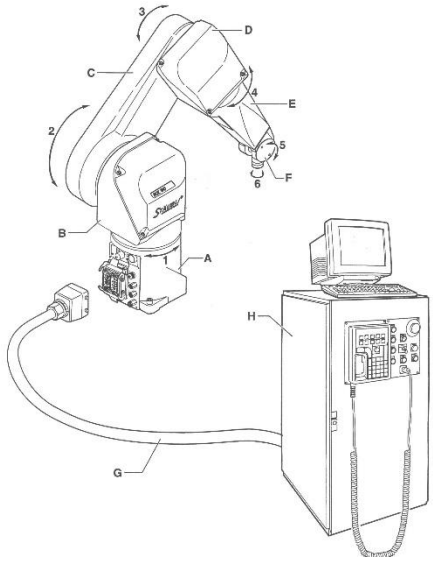
Table des matières

Ce document est largement inspiré des documents produits par Stäubli et livrés avec le robot.

1. DESCRIPTION DU ROBOT STÄUBLI RX 90

1.1 Description générale

Les robots de la famille RX série 90 sont du type polyarticulé à 6 degrés de liberté. Ils se composent, voir figure ci-dessous, d'un organe mécanique bras (A à F) et d'une baie de commande (H) (voir documentation « Caractéristiques Baie de Commande CS7 » pour plus détails), le tout étant relié par un câble de liaison (G).
Le bras est constitué de segments reliés entre eux par des articulations. Chaque articulation constitue un axe autour duquel deux segments pivotent. Les mouvements des articulations du robot sont générés par des servomoteurs (moteurs asservis) sans balais, couplés à des résolveurs (capteurs de précision). Les différents éléments du bras du robot sont le pied (A), l'épaule (B), le bras (C), le coude (D), l'avant-bras (E) et le poignet (F) (voir figure ci-dessous). L'ensemble bras du robot contient la motorisation, les freins, les mécanismes de transmission du mouvement, le faisceau de câbles, les circuits pneumatique et électrique pour l'utilisateur et le système d'équilibrage (effectué par un système intégré de ressorts) du bras.



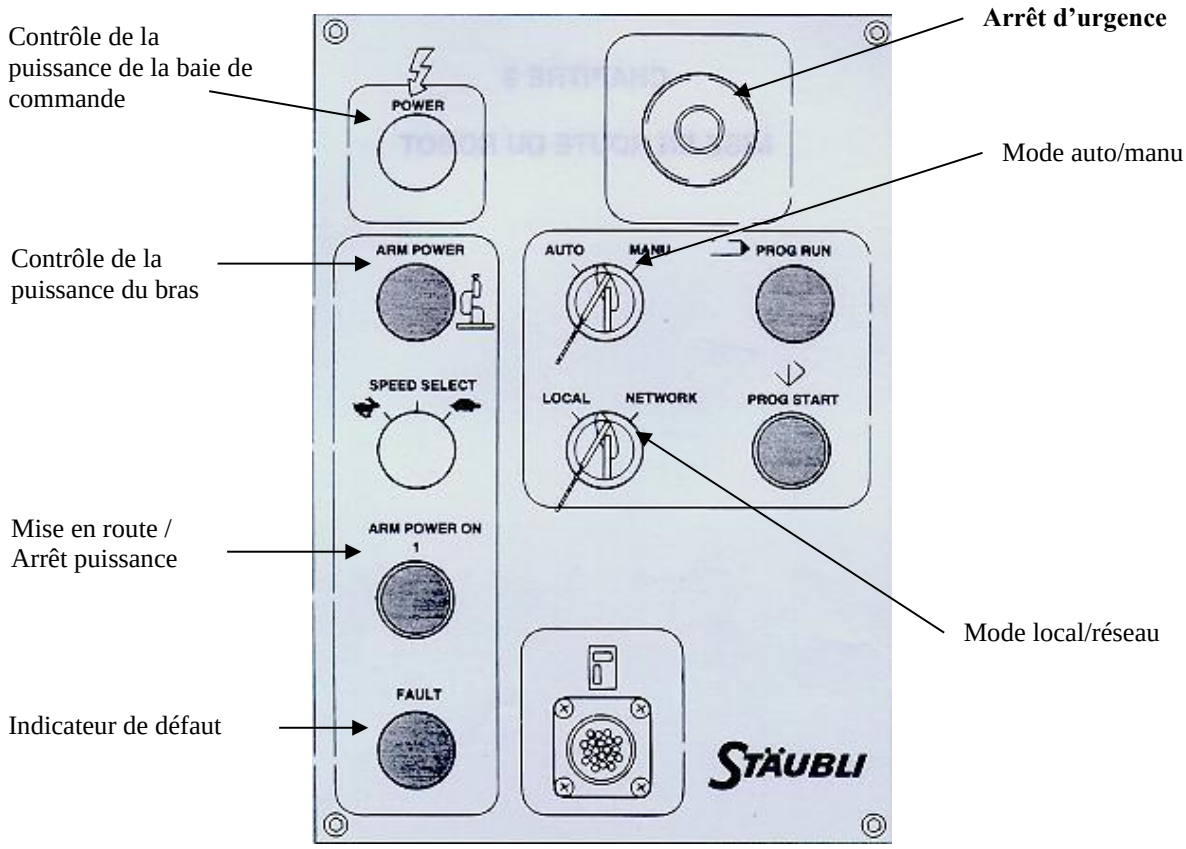
Quelques caractéristiques du robot RX 90 :

Nombre de degrés de liberté	4, 5 ou 6
Charge transportable nominale	6 Kg
Charge transportable maximale	12 Kg
Rayon d'action	985 mm
Répétabilité	+/- 0,02 mm
Langage de programmation	V+

Axe	1	2	3	4	5	6
Amplitude (°)	320 (+/- 160)	275 (+/- 137,5)	285 (+/- 142,5)	540 (+/- 270)	225 (+120/-105)	540 (+/- 270)
Vitesse nominale (°/s)	236	200	286	401	320	580
Vitesse maximale (°/s)	356	356	296	409	480	1125

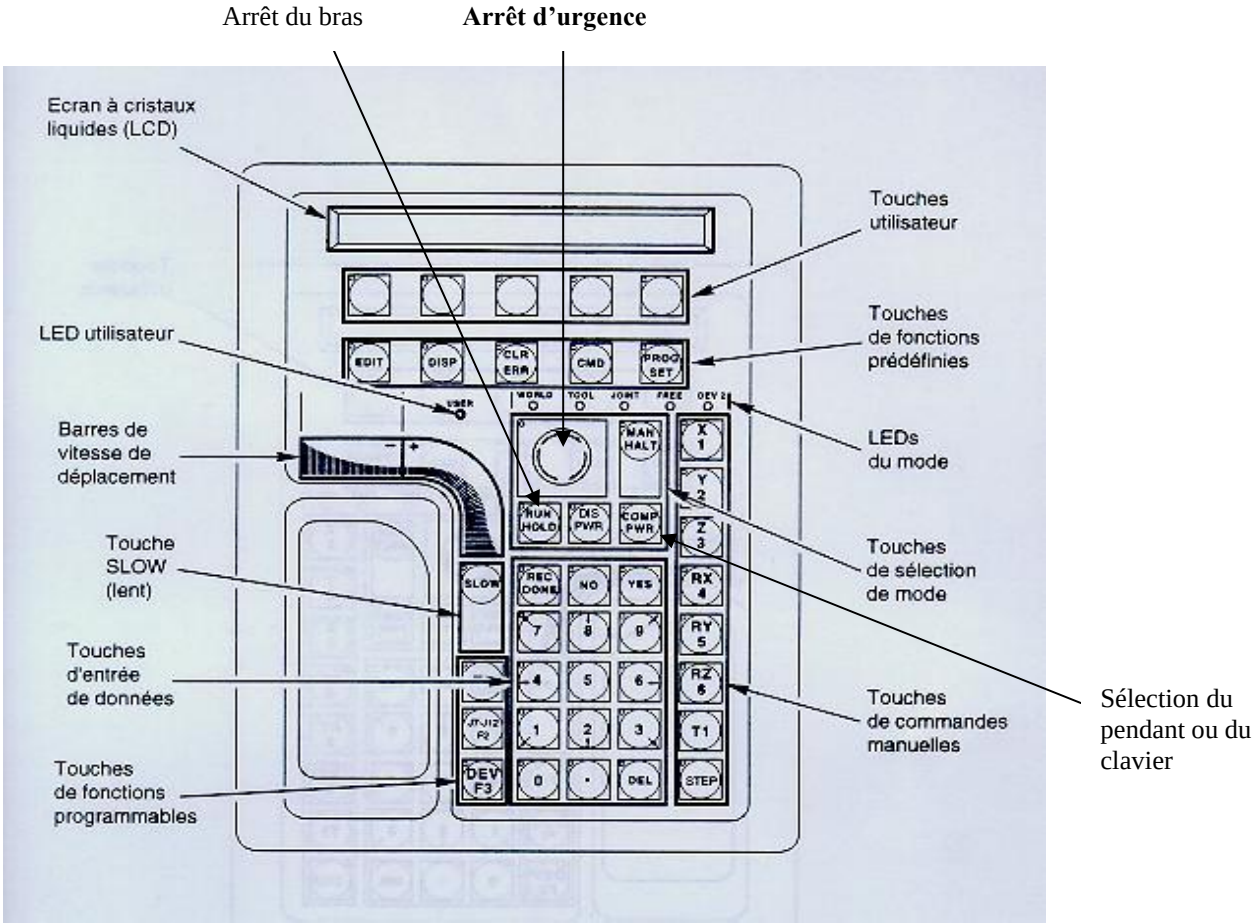
Les amplitudes (°) indiquées ci-dessus sont exprimées relativement à la configuration initiale du robot (issue d'une commande READY).

La face avant de la baie de commande CS7 se présente comme suit :



Elle permet la mise en marche, l'arrêt, la signalisation, …

Le pendant est le boîtier situé sur la face avant de la baie de commande et relié par un câble suffisamment long pour accéder au bras du robot. Il permet de commander manuellement le robot, il se compose comme suit :



La commande manuelle du robot se fait à partir du pendant, la commande automatique se faisant au clavier *via* des commandes spécifiques. L'activation du pendant, ou du clavier, se fait en appuyant sur le bouton **Comp/Pwr** du pendant.

Le pendant va permettre de définir des points de passage (appelés variables *points*) du bras du robot (*cf.* §2.3.3.b).

Règles de sécurité :
- *Personne ne doit être dans l'aire de travail du bras du robot ;*
- *La phase de test du robot s'effectue toujours en vitesse faible (***SPEED 10***) ;*
- *Durant la phase de test du robot, être prêt à appuyer sur le bouton «* **Run/Hold** *» du pendant pour arrêter le bras.*

1.2 Mise en route du système

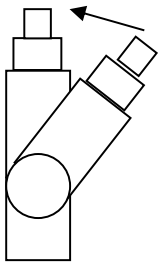
- Mettre l'interrupteur général (de couleur rouge), situé au dos de la baie de commande, sur **1**. Vérifier que le voyant ***Power***, situé sur la face avant, est allumé.
- Mettre l'écran du terminal en fonctionnement.
- Notons que le commutateur **Pendant/Terminal/Network**, situé sur la face avant de la baie, est positionné sur **Terminal**, ce qui rend le terminal actif.
- Attendre la fin du démarrage du système, c'est-à-dire, l'apparition du prompt « . » à l'écran du terminal.

- Afin de demander la mise sous puissance du bras du robot, au choix :
    - Taper au clavier l'instruction **ENABLE POWER** (**enter**),
    - Ou, appuyer sur le bouton **Comp/Pwr** du pendant.
- *Dans les 15 secondes qui suivent*, mettre la puissance sur le bras, en appuyant sur le bouton **Arm Power On** (qui clignote) de la face avant de la baie. La puissance est signalée par la présence sur la face avant de la baie du voyant ***Arm Power***. Dans le cas contraire et si le voyant ***Fault*** est allumé, vérifiez que les boutons d'arrêt d'urgence du pendant et de la face avant de la baie ne sont pas enclenchés et que le déblocage des freins (situé sur le pied du robot) est sur 0.
- Le système est prêt à accepter vos commandes. Si le terminal est actif, mettre une vitesse faible, en tapant au clavier l'instruction **SPEED 10** (**enter**).

1.3 Mise en position initiale du bras du robot

La configuration initiale du robot correspond à une position verticale du bras du robot (voir figure suivante), on a : $T_{0,6} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0{,}9 \\ 0 & 0 & 0 & 1 \end{pmatrix}$, *cf.* §3.8 du Cours. Les valeurs articulaires correspondantes sont :

$\theta_1 = 0, \theta_2 = -\frac{\pi}{2}, \theta_3 = \frac{\pi}{2}, \theta_4 = \theta_5 = \theta_6 = 0$.



A l'aide du terminal, taper au clavier (en mode *Commande*, signalé par le prompt « . ») les instructions :
- **SPEED 10 (enter)**
- **DO READY (enter)**

Remarque : le préfixe **DO** permet une exécution de la commande, en l'occurrence **READY**, en mode commande.

1.4 Arrêt du système

- *L'arrêt immédiat du bras (en mouvement)* est réalisé en appuyant sur le bouton **Run/Hold** du pendant (*l'arrêt après l'instruction en cours* d'exécution se fait en tapant au clavier l'instruction **ABORT** (**enter**)).
- La *coupure de la puissance du bras* est réalisée, au choix :
    - En appuyant sur le bouton d'arrêt d'urgence de la face avant de la baie ou celui du pendant.
    - En tapant l'instruction **DISABLE POWER** (**enter**) au clavier.

Attention : Ne pas couper la puissance de la baie de commande (*via* l'interrupteur général situé au dos de la baie de commande) quand le bras est sous puissance.

- Couper l'alimentation de l'écran du terminal.
- Mettre l'interrupteur général, situé au dos de la baie de commande, sur 0.

2 MISE EN MOUVEMENT DU BRAS DU ROBOT STÄUBLI RX 90

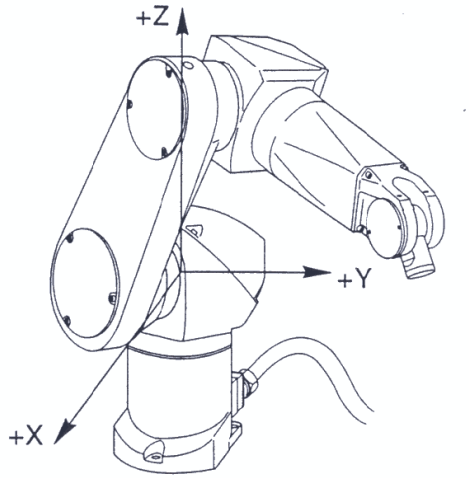
Une fois la mise en route du système effectuée (*cf.* §1.2), on dispose de 2 moyens pour mettre le bras en mouvement : Soit *via* le pendant, soit *via* l'exécution d'un programme (à l'aide du terminal).
Avant cela, décrivons les différents modes de déplacements possibles du bras.

2.1 Les modes de déplacement

La situation (position et orientation) de l'organe terminal d'un robot est déterminée à partir de 6 paramètres.

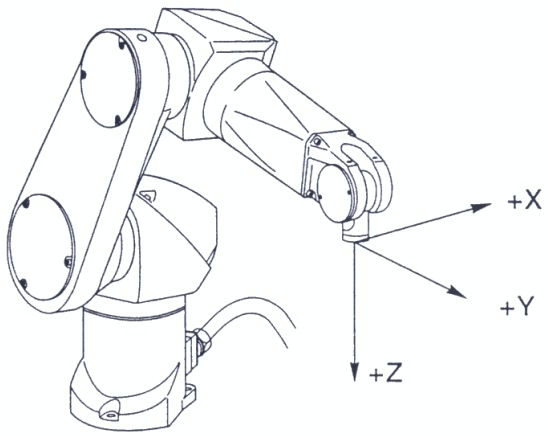
- **Le mode *World***

Tout déplacement est rapporté aux coordonnées universelles (*World*), correspondant au repère R_0 (*cf.* §3.8 du Cours) représenté par les vecteurs *X*, *Y*, *Z* dans la figure suivante. Les rotations *RX*, *RY*, *RZ* se font par rapport à ces coordonnées.



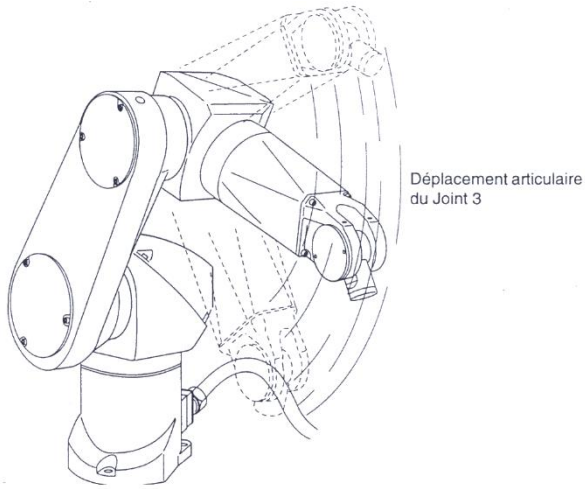
- **Le mode *Tool***

Tout déplacement est rapporté aux coordonnées d'outil (*Tool*), correspondant au repère (F, x_6, y_6, z_6) (*cf.* §3.8 du Cours) représenté par les vecteurs *X*, *Y*, *Z* dans la figure suivante. L'axe *X* est aligné avec la rainure rotative située dans la flasque. Les rotations *RX*, *RY*, *RZ* se font par rapport à ces coordonnées.



- **Le mode *Joint***

Le déplacement articulaire s'effectue autour des 6 différents axes selon les valeurs de $q_1, q_2, \cdots, q_6$ (*cf.* §3.8 du Cours), voir la figure suivante.

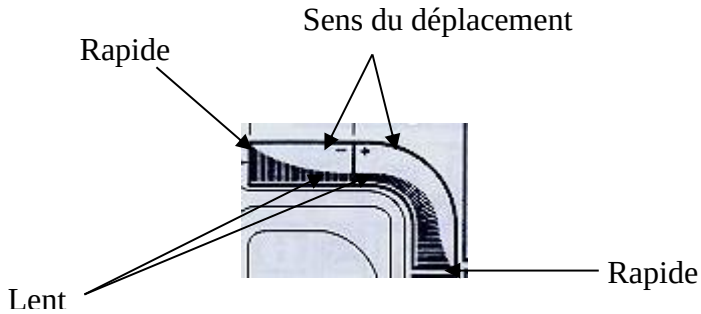


2.2 Contrôle du robot à partir du pendant

Les modes de déplacement (*cf.* §2.1) sont sélectionnés à l'aide de la touche **Man/Halt** du pendant. Après la mise sous tension du système, le système est en mode **World.** L'appui successif sur cette touche permet le passage d'un mode à l'autre (**World** ⇒ **Tool** ⇒ **Joint** ⇒ **World**).

Une fois le mode de déplacement sélectionné, on peut choisir l'axe sur lequel on veut se déplacer à l'aide des touches *X/1, Y/2, Z/3, RX/4, RY/5, RZ/6.* Les lettres *X*, *Y* et *Z* représentent les 3 axes ; *RX, RY, RZ* représentent les rotations autour de ces axes. Les chiffres 1 à 6 représentent, dans le contexte du mode **Joint**, les 6 axes du robot.

La barre de vitesse permet de choisir la vitesse et le sens (+/-) du déplacement manuel :



La touche **Slow** du pendant permet de choisir entre deux plages de vitesses différentes des barres de vitesses : On peut choisir une vitesse de déplacement normale ou lente.

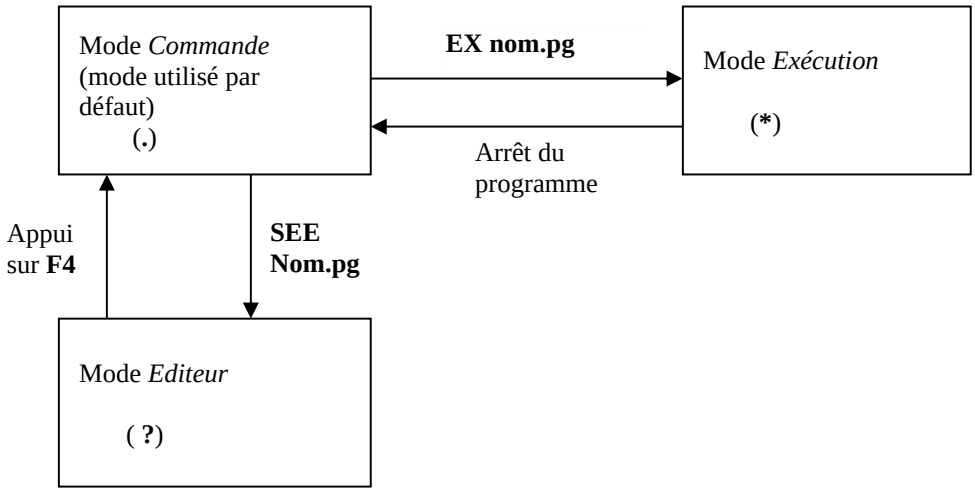
2.3 Contrôle du robot à partir d'un programme

Il existe 3 prompts à l'écran :

- Le prompt « **.** » pour le mode *Commande*. On fournit au robot des instructions du type **DIR, FDIR, FORMAT**,
- Le prompt « ***** » pour le mode *Exécution*. On fournit des instructions du type **ABORT**, **EXECUTE**, **PANIC**, **RETRY**, **XSTEP**,

- Le prompt « **?** » pour le mode *Editeur*. On fournit des instructions *programme*, ou des commandes *éditeur*.

On peut passer d'un mode à l'autre de la manière suivante :



Remarque : Le préfixe **DO com** permet l'exécution de l'instruction **com** en mode *Commande* (par exemple **DO READY**).

2.3.1 Le programme du robot existe déjà

Pour exécuter le programme **nom.pg** (situé dans la mémoire vive de la machine), utiliser l'instruction **EXECUTE nom.pg** (**enter**). Le nom du programme doit commencer par une lettre et comporte au maximum 15 caractères.
Le fait de taper, avant d'utiliser l'instruction **EXECUTE** ou en cours d'exécution du programme, l'instruction **ENABLE TRACE** (**enter**) fait apparaître à l'écran les différentes instructions au fur et à mesure de leurs exécutions. L'instruction **DISABLE TRACE** (**enter**) permet un retour en mode normal.

L'instruction **XSTEP nom.pg** (**enter**) permet l'exécution du programme en mode pas à pas. Le fait de taper **X** (**enter**) permet d'exécuter le pas suivant.

Notons que la touche **Run/Hold** du pendant provoque un arrêt *immédiat* du mouvement du bras, ainsi que du programme, alors que l'instruction **ABORT** (**enter**) tapée au clavier provoque l'arrêt qu'à la fin de l'instruction en cours.
La reprise du cycle se fait *via* l'instruction **RETRY** (**enter**).

2.3.2 L'éditeur SEE

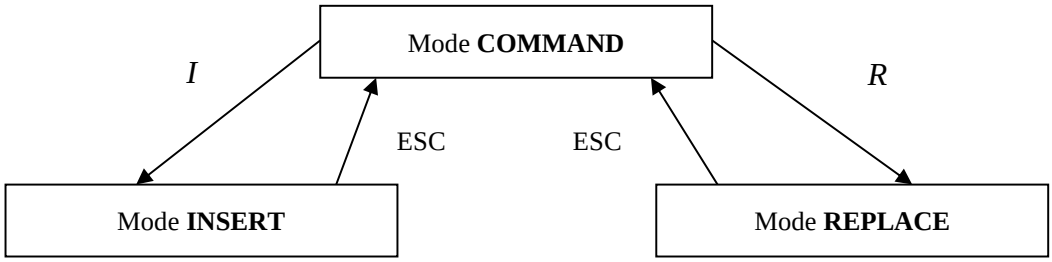
L'instruction **SEE nom.pg** (**enter**) provoque le passage du mode *Commande* au mode *Editeur*. **SEE** est un éditeur pleine page qui permet, en insérant, ou modifiant des instructions, de créer un programme. Appuyer sur la touche **F4** pour quitter l'éditeur.

Cet éditeur a 3 modes d'édition : **COMMAND, INSERT, REPLACE**.

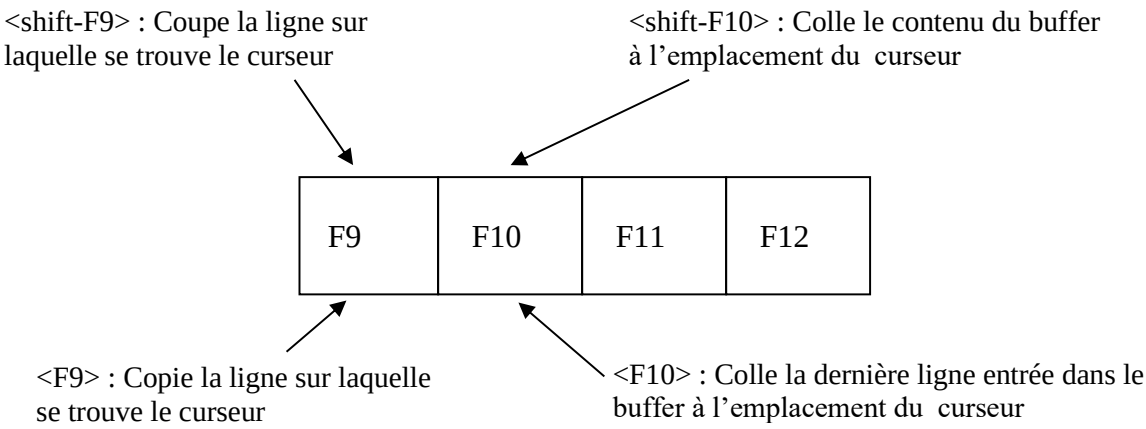
- Le mode **COMMAND** est celui par défaut. Dans ce mode, le code programme n'est pas entré, seules les commandes spéciales éditeur peuvent aboutir.
- Le mode **INSERT** permet à partir de l'emplacement du curseur l'insertion de nouveaux caractères dans le programme.

- Le mode **REPLACE** permet à partir de l'emplacement du curseur le remplacement du texte existant par un nouveau texte.

En mode **COMMAND**, on peut lire les données (il est possible de supprimer un ou des caractères en appuyant sur la touche **D**).
Pour écrire ou remplacer, il suffit de taper sur la touche **I** (mode **INSERT**) ou **R** (mode **REPLACE**). Schématiquement, on a :



Remarque : Lors de l'écriture d'un programme, on peut effectuer des *copier/coller* :



Remarque : En fait, les programmes générés sont sous forme de fichiers texte, ce qui permet l'utilisation d'un traitement de texte tel que le Bloc-notes (*NotePad*) de Windows.

2.3.3 Quelques commandes

La programmation du robot se fait *via* le langage V+, qui est un langage de programmation interprété, structuré de haut niveau, multitâche. Seuls quelques éléments de base sont donnés.

Le signe « **; »** permet d'insérer un commentaire.

a) Eléments standard de programmation

Listons tout d'abord les éléments standards à un langage :

1) Les *identificateurs* commencent obligatoirement par un caractère alphabétique, suivi d'un nombre quelconque de lettres, chiffres ou points (qui jouent le rôle du « _ » en Pascal, C). Ils n'ont pas à être déclarés, le type étant donné par le contexte (3 types sont possibles (voir ci-dessous)).

2) Il existe différentes classes d'*opérateurs*, notamment des opérateurs :

- mathématiques : +, -, *, /, MOD,
- relationnels : <, <=, = =, <>, >=, >,
- logiques : AND, OR, NOT.

3) *Structures algorithmiques* (les plus classiques)

L'instruction **GOTO** permet une exécution non séquentielle des instructions. Elle permet, par exemple, la création de boucle *via* l'utilisation de *label* et de l'instruction **GOTO** *label*.

Structure **CASE … OF**

```
CASE  2*i + 1  OF
VALUE 1, 2:
   :
VALUE 3:
   :
ANY;     (sans mettre ":")
   :
END
```

Structure **IF … THEN … ELSE … END**

```
IF (i MOD 2 == 0) OR (i > j) THEN
   :
ELSE
   :
END
```

Structure **FOR … END**

```
FOR i = max TO min STEP -1
   T[i]=i
END
```

Structure **WHILE … DO … END**

```
WHILE (i < max) AND (t[i] <> 0) DO
   :
END
```

Structure **DO … UNTIL**

```
DO
   :
UNTIL t[i] == 0
```

L'instruction **WAIT *condition*** interrompt l'exécution du programme jusqu'à ce que l'expression ***condition*** (par exemple, **SIG 1002**) soit valide.

b) Déclarations de variables

Il existe **3 types de variables** :

- Une variable de type **point** permet de mémoriser un *point* du bras du robot, défini par 6 paramètres indépendants : 3 pour la position, 3 pour l'orientation. Une variable **point** peut être considérée **comme un repère donné par son origine et son orientation**, ou comme une transformation

entre deux repères. Une variable **point** peut être définie dans l'espace opérationnel (voir *i*) ou articulaire (voir 2*i*) :

***i*)** Une manière simple pour mémoriser la *situation de l'organe terminal (ou de l'outil)*, correspondant au point de passage courant du bras du robot (atteint par exemple *via* le pendant), dans une variable **A** (de type **point**) consiste à taper au clavier :

HERE A (**enter**)
X Y Z y p r
374 520 355 55 134 1
Change ? (**enter**)

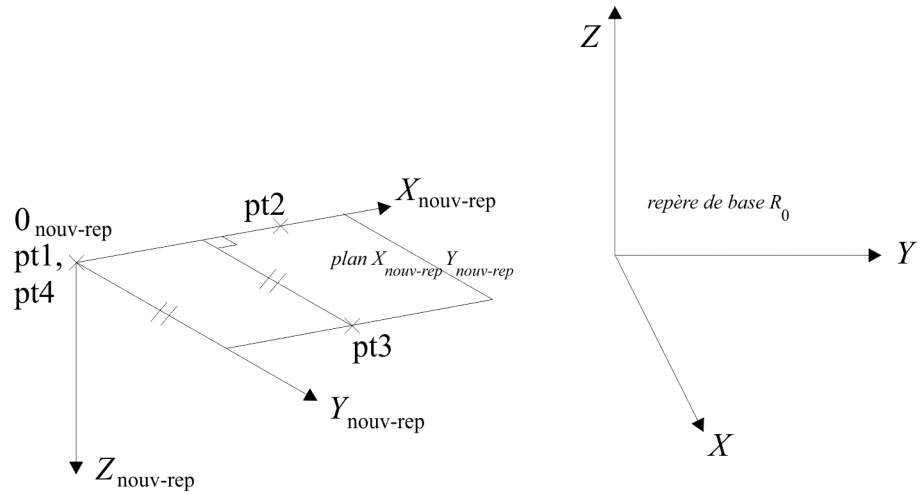
Ainsi, la variable point **A** contient les données X, Y, Z, y, p, r, permettant ainsi de mémoriser la situation de l'organe terminal (associé au repère outil ($F, x_6, y_{6,} z_6$)).

Alors qu'il est aisé de définir une position *X*, *Y*, *Z* du robot (à travers l'origine, *F*, du repère outil), il est souvent difficile de définir précisément une orientation. Par exemple pour certaines applications (manipulation de pièce, palettisation, …), il s'avère utile de définir avec précision un **repère associé** (par exemple à une palette) **à partir duquel les déplacements vont pouvoir se référer** (et donc faciliter la programmation des déplacements). Pour cela, la fonction **FRAME** peut être utilisée du fait qu'elle permet de définir un repère (position et orientation) à partir de 4 points. L'instruction

SET nouv-rep = FRAME(pt1, pt2, pt3, pt4)

permet de définir le repère **nouv-rep** tel que :
- l'origine du repère est défini par le point **pt4** ;
- l'axe X est défini par une droite parallèle à la droite passant par les points **pt1** et **pt2** (le sens positif étant dirigé vers **pt2**) ;
- l'axe Y est défini par une droite perpendiculaire à l'axe X passant par le point **pt3** (le sens positif étant dirigé vers **pt3**).

Ainsi le point **pt4** appartient au plan XY lequel est parallèle au plan contenant les points **pt1**, **pt2**, **pt3**. Le point **pt4** est souvent confondu avec le point **pt1** pour simplifier la prise de points, voir la figure suivante.



2i) Une autre variable de type point existe. Elle permet de mémoriser *la posture du bras du robot* (en mémorisant les variables *J*1, *J*2, …, *J*6), et donc également la situation de l'organe terminal

(ce que faisait la précédente variable point). Pour cela, l'identifiant de la variable *doit* commencer par un dièse (#). Par exemple :

HERE #A (enter)
J1 J2 J3 J4 J5 J6
54 72 164 181 -40 -182
Change ? (**enter**)

permet de mémoriser la posture courante du bras du robot dans la variable **#A** (attention, la variable créée est **#A** et non **A**).

L'instruction **SET B = A** permet d'affecter à la variable point **B** la valeur **A** (correspondant à un point).
L'instruction **SET C = HERE** permet d'affecter à la variable point **C** *le point de passage courant* du bras du robot.
L'instruction **SET loc1 = TRANS(550, 450, 750, 0, 180, 45)** permet d'affecter à la variable point **loc1** le point correspondant aux *coordonnées opérationnelles X* = 550 *mm*, *Y* = 450 *mm*, *Z* = 750 *mm*, *y*(*aw*) = 0°, *p*(*itch*) = 180°, *r*(*oll*) = 45°.
L'instruction **SET #loc2=#PPOINT(80, -20, 120, 0, 76, -182)** permet d'affecter à la variable point **#loc2** le point correspondant aux *coordonnées articulaires J*1 = 80°, *J*2 = -20°, …, *J*6 = -182° (attention, les # sont obligatoires et il n'y a pas d'espace de part et d'autre du signe =).
L'instruction **SET arrivee = DEST** permet d'affecter à la variable point **arrivee** *le point de destination courante* du bras du robot.

L'instruction **POINT X** permet de créer une variable point **X** en introduisant directement au clavier ses coordonnées (sans se référer à la position courante du bras du robot). Par exemple :

POINT y (enter)
Change ? 550, 0, 400 (**enter**)

permet d'affecter à la variable point y le point correspondant aux coordonnées opérationnelles :
X = 550 *mm*, Y = 0 *mm*, Z = 400 *mm*, y = p = r = 0°.

- Les variables **réelles** ou **entières** (entre –16 777 216 et 16 777 215) :
  Exemple : **A = 12**
  B = 2.356
  C = B

- Les variables **chaînes** définies par un **$** en première position :
  Exemple : **$réponse = « Répondre par Oui ou Non »**

c) Commandes associées à la mémoire vive

Les programmes, ainsi que les variables, doivent être placés dans la mémoire vive de la machine afin d'être exécutés.

Pour visualiser les titres des programmes situés dans le répertoire courant, taper **DIR** (**enter**).

Pour visualiser les contenus (instructions disponibles uniquement en mode *Commande*) :
- du programme **nom.pg**, taper : **LISTP nom.pg** (**enter**),
- des coordonnées des points **A** et **#B**, taper : **LISTL A,#B** (**enter**),

- des variables réelles **X** et **Y**, taper : **LISTR X, Y** (**enter**),
- des variables chaînes **$réponse** et **$nom**, taper : **LISTS $réponse, $nom** (**enter**).

Pour copier un programme, taper **COPY nom du nouveau prog = nom de l'ancien prog** (**enter**)

Pour renommer un programme, taper **RENAME nom du nouveau prog = nom de l'ancien prog** (**enter**)

Pour supprimer :

- le programme **Nom.pg**, taper : **DELETEP nom** (**enter**),
- les coordonnées du point **nom_point**, taper : **DELETEL nom_point** (**enter**),
- la variable réelle **var_réelle**, taper : **DELETER var_réelle** (**enter**),
- la variable chaîne **$var_chaîne**, taper : **DELETES $var_chaine** (**enter**).

Pour vérifier l'existence du programme toto.pg, taper : **TESTP toto.pg** (**enter**).

L'instruction **ZERO** permet de vider le contenu total de la mémoire vive.

d) Opérations de sauvegarde sur une disquette ou un disque dur

Les programmes, ainsi que les variables, peuvent être sauvegardés sur une disquette et/ou un disque dur.

Pour sauvegarder, sur une disquette et/ou un disque, **un programme**, appelé **nom**, **et ses variables associées** éventuelles**,** on utilise l'instruction **STORE nom**. Il en résulte un fichier **nom.V2**.

On utilise les instructions **STOREP**, **STOREL**, **STORER**, **STORES** pour sauvegarder uniquement respectivement des programmes, des points, des réels, des chaînes.

Pour charger un fichier (situé sur une disquette et/ou un disque) **en mémoire vive**, on utilise l'instruction **LOAD**. Par exemple :

LOAD nom.pg charge le fichier **nom.pg** situé dans le répertoire courant en mémoire vive.

L'instruction **FDIRECTORY** permet :

- De **visualiser les fichiers** contenus dans le répertoire courant.
- De **créer un répertoire** à partir du répertoire courant :

  Exemple : Créer le répertoire c:\DESS\TD, sachant que le répertoire courant est c:\, taper :

  FDIR/C c:\DESS

  puis

  FDIR/C c:\DESS\TD

- De **supprimer un répertoire** à partir du répertoire courant :

  Exemple : Supprimer le répertoire TD, situé dans le répertoire DESS, taper :

  FDIR/D c:\DESS\TD

L'instruction **CD** permet de **changer de répertoire** du disque dur (c :) ou de la disquette (a :), par exemple :

CD=c:\Repertoire1\Repertoire2

Remarque : **CD ..** permet de remonter d'un répertoire.

L'instruction **FLIST** permet de **visualiser le contenu d'un fichier** (situé sur une disquette et/ou un disque), par exemple :

FLIST c:\nom.pg

L'instruction **FDELETE** permet de **supprimer un fichier** (situé sur une disquette et/ou un disque), par exemple :

FDELETE nom.pg

L'instruction **FCOPY** permet de **copier un fichier** (situé sur une disquette et/ou un disque), par exemple :

FCOPY fichier cible = fichier source

Attention, l'ordre des fichiers *cible* et *source* est inversé par rapport à DOS !

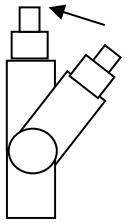
L'instruction **FRENAME** permet de **changer le nom d'un fichier** (situé sur une disquette et/ou un disque), par exemple :

FRENAME nouveau nom fichier = ancien nom fichier

e) Quelques instructions *programme*

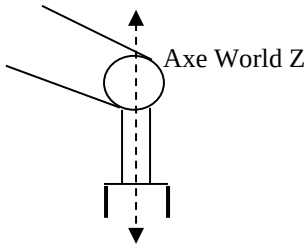
Les mouvements _____

- ✓ (**DO**) **READY** (**DO** permet une exécution en mode *Commande*) remet le bras du robot dans sa configuration initiale.



- ✓ L'instruction **WHERE** permet de connaître la configuration du robot et l'état du préhenseur au moment où cette instruction est exécutée. Par exemple, si la variable **A** de type point est présente, on peut afficher ses valeurs *via* l'instruction **WHERE A**.

- ✓ L'instruction **ALIGN** permet de placer l'axe Z du repère outil (terminal), correspondant à l'axe z_6, parallèlement à l'axe le plus proche du référentiel de base (World) (à savoir x_0, y_0 ou z_0). Cette instruction évite de faire de nombreuses manipulations *via* le pendant afin de rendre l'axe Z du repère outil colinéaire avec, par exemple, l'axe Z du repère de base (World) :



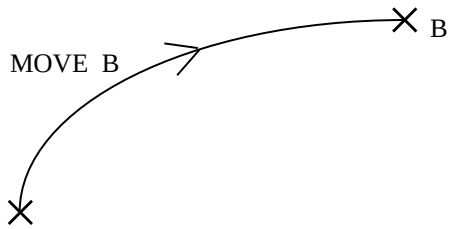
- ➢ On spécifie toujours la destination du repère outil du robot, et éventuellement des contraintes sur la façon d'atteindre la destination (par exemple en ligne droite). La destination est toujours un point (ou un repère), ainsi le mouvement impose d'atteindre une position donnée, avec une orientation donnée. Il n'existe pas de moyen de demander un mouvement en laissant libre, par exemple, l'un des angles d'orientation.

- ✓ L'instruction **ABOVE** (resp. **BELOW**) permet, lors du prochain mouvement, de forcer le coude (3ème articulation) à être en position haute (resp. basse).

✓ L'instruction **RIGHTY** (resp. **LEFTY**) permet, lors du prochain mouvement, de forcer l'épaule (deux premières articulations) à être à droite (similaire à l'épaule droite humaine) (resp. à gauche).

✓ Cinq types de primitives de mouvement sont listés ci-dessous : **MOVE(S)**, **APPRO(S)**, **DEPART(S)**, **DRIVE**, **DELAY**.

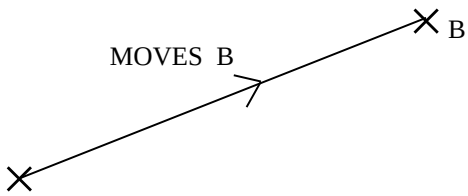
1) L'instruction **MOVE point**, ou **MOVE #point**, (précédée de **DO** si l'exécution est en mode commande) ordonne un déplacement du repère outil du robot vers le point spécifié selon une trajectoire non imposée par l'utilisateur (dépendante de la morphologie du manipulateur) :



L'instruction **MOVE B** est telle qu'à l'issue du mouvement, il y a **correspondance entre le repère outil** $(F, x_6, y_{6,} z_6)$ **et le repère défini par le point B**.

L'instruction **MOVE #B** est telle qu'à l'issue du mouvement, il y a **correspondance entre la posture du robot** (fonction de *J*1, *J*2, …, *J*6) **et les valeurs angulaires définies par le point #B** (ceci permet également de fixer le repère outil).

L'instruction **MOVES point**, ou **MOVES #point**, (précédée de **DO** si l'exécution est en mode commande) est similaire à l'instruction précédente, mais est telle que la trajectoire de l'origine du repère terminal est *une droite* :

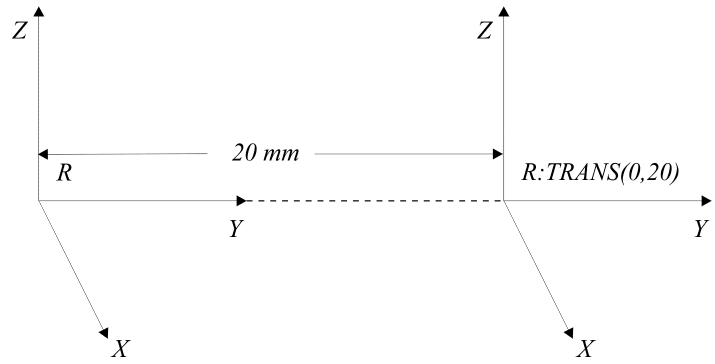


Voir §4.2 du Cours concernant le déplacement induit par une séquence d'instructions de mouvement.

1.1) Instruction **TRANS**

Il est possible de modifier, relativement à un repère donné *R* (point correspondant défini dans l'espace opérationnel, par exemple, à travers l'instruction **HERE** ou **FRAME**), la situation du repère outil en programmant, par exemple :
- une translation de 20 mm le long de l'axe *Y* du repère *R via* l'instruction

MOVE *R*:TRANS(0, 20)



- une translation de 10 mm le long de *X* et de 30 mm le long de *Z* du repère *R via* l'instruction

MOVE *R*:TRANS(10, 0, 30)

En fait, l'instruction **TRANS(val_X, val_Y, val_Z, val_y, val_p, val_r)** permet de modifier la situation du repère outil (en position le long de *X, Y, Z* et en orientation autour de *y, p* et *r*).

1.2) Instruction **SHIFT**

L'instruction **SHIFT** permet de modifier les coordonnées de **position** définies dans R_0, soit *X, Y, Z,* d'un point(repère). Par exemple, si le point H a pour coordonnée :

(200, 150, 100, 10, 20, 30)

alors l'exécution de l'instruction :

SET I = SHIFT(H BY 0, 0, 50)

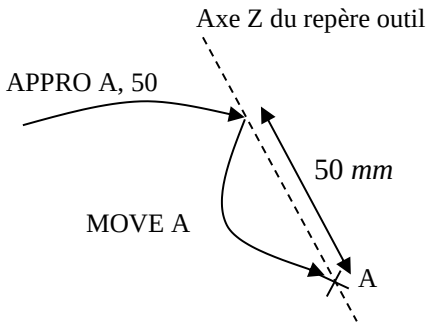
fait que le point I a pour coordonnée :

(200, 150, 150, 10, 20, 30),

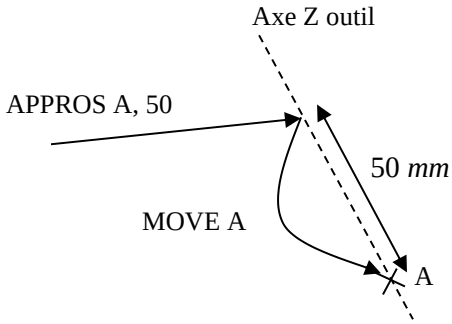
ce qui correspond à une translation de 50 *mm* selon l'axe z_0 par rapport au point(repère) I.

Remarque : l'instruction **SHIFT** ne permet pas de modifier l'orientation (*y, p, r*) d'un point(repère).

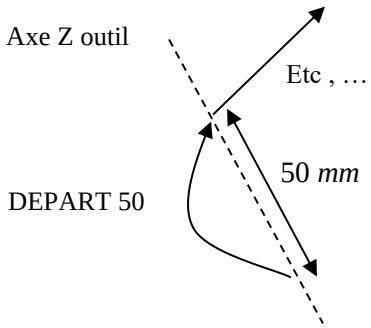
2) L'instruction **APPRO point, distance**, ou **APPRO #point, distance**, généralement suivie de l'instruction **MOVE**, permet au robot d'approcher le point spécifié, en restant en retrait de la distance spécifiée (en *mm*), selon l'axe Z du repère outil (voir lien avec l'instruction **ALIGN**).



L'instruction **APPROS** est similaire à l'instruction précédente, mais est telle que la trajectoire de l'origine du repère terminal est *une droite* :



3) L'instruction **DEPART distance** réalise la fonction inverse de l'instruction **APPRO** ; elle permet de se dégager de la distance indiquée (en *mm*) selon l'axe Z du repère outil, par rapport à la position courante (*i.e.,* la position juste avant l'exécution de l'instruction **DEPART**).
Soit **B** la position juste avant l'exécution de l'instruction « **DEPART 50** ». Soit **A** la position du repère outil résultant du mouvement, alors l'instruction « **APPRO A, 50** » permet de retourner en **B**.



L'instruction **DEPARTS** est similaire à l'instruction précédente, mais est telle que la trajectoire de l'origine du repère terminal est *une droite.*

4) L'instruction **DRIVE articulation, angle, pourcentage de vitesse** permet d'assurer un déplacement articulaire suivant l'articulation désignée. Par exemple :

DRIVE 1, 30, 50

crée un déplacement suivant l'articulation 1 de 30 degrés à 50% de la vitesse moniteur.

5) **DELAY temps** permet au robot de s'arrêter pendant le temps stipulé dans l'instruction. Ce temps doit être exprimé **en seconde** et ne peut être inférieur à 16 *ms*.
Soit par exemple, l'instruction : **DELAY 30**

➢ L'instruction **DELAY** est considérée comme une instruction de mouvement sans déplacement (soit encore un mouvement vers le point courant).

✓ Instructions de mouvements de la pince :

OPENI ; permet d'ouvrir la pince,
CLOSEI ; permet de fermer la pince.

Important : Par défaut les mouvements du bras du robot sont ***asynchrones***, autrement dit une ***instruction de mouvement*** **se termine dès que le mouvement est entamé**. Aussi, le programme se poursuit avec l'évaluation de l'instruction suivante, **sauf si celle-ci est elle-même une instruction de mouvement** (rappelons que c'est le cas de l'instruction **DELAY**). Cette notion - qui existe dans tous langages de programmation de haut niveau de robots - constitue une différence essentielle entre la programmation d'un robot et la programmation d'une application informatique classique.

✓ Pour que le programme ne se poursuive qu'après achèvement d'un mouvement, il faut le demander explicitement à l'aide de l'instruction **BREAK**. En effet, cette instruction suspend l'exécution du programme jusqu'à ce que le mouvement ait atteint sa destination. Cette instruction est utile pour éviter l'exécution d'instructions de calculs, de gestions entrées/sorties, … (autres que des instructions de mouvements), situées entre deux mouvements.
Comparons par exemple les 2 portions de code suivantes :

MOVE A	**MOVE A**
TYPE "le mouvement commence"	**TYPE "le mouvement commence"**
	BREAK
	TYPE "le mouvement est terminé"

✓ On peut également donner des indications de vitesse, soit en pourcentage de la vitesse maximale pour un mouvement en articulaire, soit en millimètres par seconde pour un mouvement en cartésien.

Il est possible d'introduire, au sein du programme, des instructions de vitesse. Ces vitesses sont systématiquement multipliées par la vitesse dite « moniteur », introduite au clavier avant l'exécution du programme.
Ainsi, si la vitesse indiquée dans le programme est de 40% alors que la vitesse « moniteur » est de 80%, alors la vitesse résultante du manipulateur sera de $4 \times 8 = 32\%$ de la vitesse nominale du robot. Il en va de même pour une vitesse exprimée en *mm/sec* (noter que celle si sera respectée que pour une vitesse « moniteur » égale à 100%).

Par exemple :
- **SPEED 50 ALWAYS** $\Rightarrow$ vitesse de 50% de la vitesse « moniteur » pour un mouvement en articulaire. Si **ALWAYS** n'est pas indiqué, seul le prochain mouvement en articulaire est concerné. Sinon, l'indication est valable jusqu'à la prochaine indication de vitesse en articulaire.
- **SPEED 500 MMPS ALWAYS** $\Rightarrow$ vitesse de 500 *mm/sec* pour un mouvement en cartésien. Si **ALWAYS** n'est pas indiqué, seul le prochain mouvement en cartésien est concerné. Sinon, l'indication est valable jusqu'à la prochaine indication de vitesse en cartésien.

Les sous-programmes _____

Seule la notion de sous-programmes sans paramètre existe. Un sous-programme est défini par un identificateur et se termine par **RETURN**. L'appel du sous-programme s'effectue par l'instruction **CALL nom_programme**. Toutes les variables sont globales.
Par exemple, l'instruction **:**

CALL prg.pg( )

permet d'appeler le programme **prg.pg**.

Entrées/sortie écran/clavier _____

La commande **PROMPT** permet la lecture de données introduites au clavier.
Par exemple, l'instruction :

PROMPT « Donner la position : » , x, y, z

est telle que le message « **Donner la position :** » s'affiche à l'écran et le programme attend que trois valeurs soient introduites avant de poursuivre.

La commande **TYPE** permet d'écrire à l'écran un message.
Par exemple, l'instruction :

TYPE « La vitesse est : », v, « et l'acceleration est : », gamma

permet l'écriture à l'écran du message entre « », accompagné des valeurs des variables **v** et **gamma**.

Entrées/sorties digitales (tout ou rien) _____

En général, un robot interagit avec d'autres systèmes. C'est le cas du robot situé en salle 215, au sens où le robot communique avec le système de transfert.

- ✓ **Programmation des sorties :**
  Les adresses des sorties sont 1 pour la sortie n°1, 2 pour la sortie n°2, …, 8 pour la sortie n°8 (sachant que les sorties 7 et 8 sont généralement réservées et que le nombre de sorties peut être étendu).

  L'instruction **SIG** permet d'écrire (d'actionner le contact) sur une sortie digitale. Par exemple, l'instruction :

  SIG 2, -1, 4

  permet de mettre à l'état 1 les sorties n° **2** et **4**, et à l'état 0 la sortie **1**.

  L'instruction **RESET** remet à zéro toutes les sorties.

- ✓ **Les entrées :**
  Les adresses des entrées sont 1001 pour l'entrée n°1, 1002 pour l'entrée n°2, …, 10012 pour l'entrée n°12 (leur nombre peut être étendu).

  L'instruction **SIG** permet de lire l'état des entrées digitales.
  Par exemple, l'instruction :

  SIG 1002, 1004

  permet la lecture des entrées 2 et 4.

- ✓ **Scrutation des entrées :**

WAIT SIG(1001)	; attendre que *E1* soit à l'état *1*
WAIT SIG(1001) AND SIG(-1002)	**;** attendre que *E1* soit à l'état 1 et *E2* à l'état *0*
WAIT SIG(1001) OR SIG(1002)	; attendre que *E1* ou *E2* soit à l'état *1*
IF SIG(1001) THEN …	; si *E1* est à l'état *1* alors …

- ✓ **Réaction aux entrées :**

L'instruction

REACT 1001, SPRG1

permet l'exécution du sous-programme **SPRG1** si *E1* est à l'état *1*. Le mouvement en cours ne sera pas interrompu. Si le sous-programme contient des instructions de mouvement, elles ne seront exécutées qu'à la fin du mouvement en cours. Le retour au programme interrompu s'effectue sur l'instruction qui suit l'instruction après laquelle le déroutement a eu lieu.

L'instruction

REACTI 1002, SPRG2

permet d'aller **immédiatement** exécuter le sous-programme **SPRG2** si *E2* est à l'état *1*. Le mouvement en cours est interrompu. Après exécution de **SPRG2**, le programme interrompu est poursuivi à l'instruction suivante, aussi le mouvement interrompu n'est pas terminé. Pour terminer le mouvement, le programmeur peut, par exemple, mémoriser la destination au moment de l'interruption et demander l'exécution du mouvement à la fin du sous-programme **SPRG2**.