Présentation du robot YuMi®

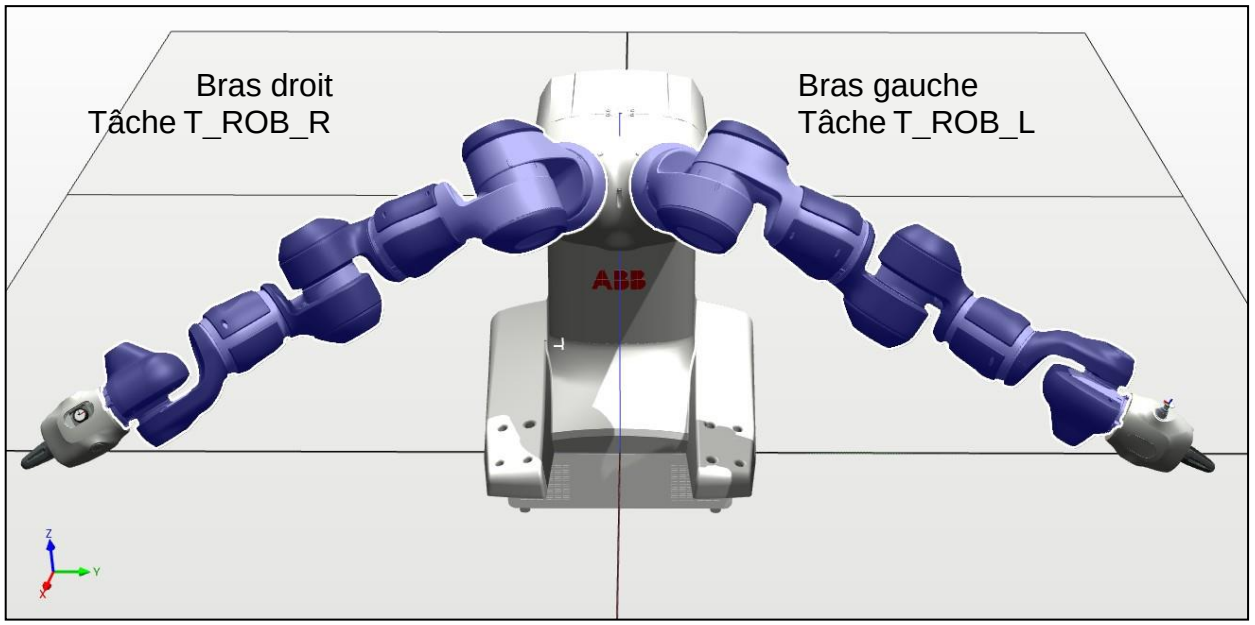
1 Unités mécaniques

YUMI est un robot collaboratif constitué d'une base sur laquelle repose deux bras

de 7 axes chacun.

Chaque bras est une unité mécanique avec sa propre tâche (T_ROB_L ou T_ROB_R).

Il y a un bras droit et un bras gauche (disposé comme sur un humain faisant face).



2 Utilisation des mains

Pour chaque combinaison de module sur la main, il y a donc des données de charge et des centres d'outils différents.

- Charges en présence de doigts / ventouse(s) / filtre(s)

Combinaison	Repère outil
Asservi	[TRUE, [[0, 0, 0], [1, 0, 0 ,0]], [0.230, [8.2, 11.7, 52.0], [1, 0, 0, 0], 0.00021, 0.00024, 0.00009]]
Servo + Vide	[TRUE, [[0, 0, 0], [1, 0, 0 ,0]], [0.248, [8.6, 11.7, 52.7], [1, 0, 0, 0], 0.00021, 0.00024, 0.00009]]
Servo + Vide 1 + Vide 2	[TRUE, [[0, 0, 0], [1, 0, 0 ,0]], [0.280, [7.1, 11.9, 47.3], [1, 0, 0, 0], 0.00025, 0.00029, 0.00012]]
Servo + Vision	[TRUE, [[0, 0, 0], [1, 0, 0 ,0]], [0.244, [7.5, 11.8, 52.7], [1, 0, 0, 0], 0.00021, 0.00023, 0.00008]]
Servo + Vision + Vide	[TRUE, [[0, 0, 0], [1, 0, 0 ,0]], [0.262, [7.8, 11.9, 50.7], [1, 0, 0, 0], 0.00022, 0.00024, 0.00009]]

- Charges en l'absence de doigt / ventouse / filtre

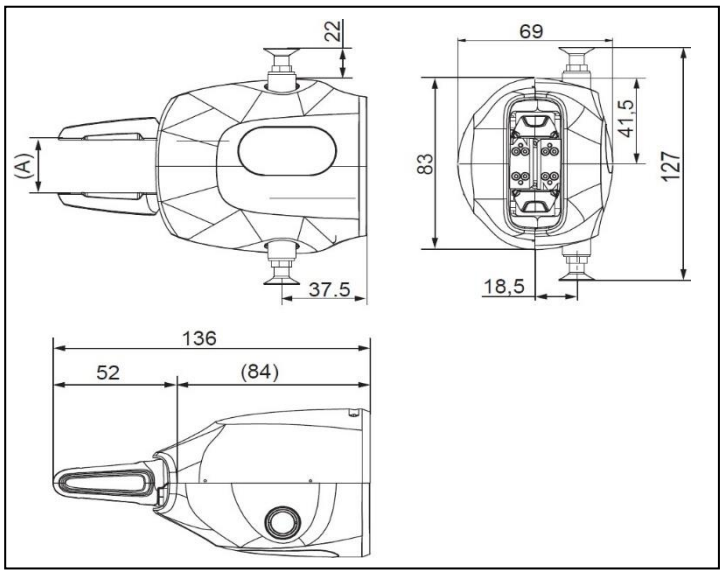
Combinaison	Repère outil
Asservi	[TRUE, [[0, 0, 0], [1, 0, 0 ,0]], [0.215, [8.7, 12.3, 49.2], [1, 0, 0, 0], 0.00017, 0.00020, 0.00008]]
Servo + Vide	[TRUE, [[0, 0, 0], [1, 0, 0 ,0]], [0.226, [8.9, 12.3, 48.7], [1, 0, 0, 0], 0.00017, 0.00020, 0.00008]]
Servo + Vide 1 + Vide 2	[TRUE, [[0, 0, 0], [1, 0, 0 ,0]], [0.250, [7.4, 12.4, 44.8], [1, 0, 0, 0], 0.00020, 0.00024, 0.00011]]
Servo + Vision	[TRUE, [[0, 0, 0], [1, 0, 0 ,0]], [0.229, [7.9, 12.4, 48.7], [1, 0, 0, 0], 0.00017, 0.00019, 0.00008]]
Servo + Vision + Vide	[TRUE, [[0, 0, 0], [1, 0, 0 ,0]], [0.240, [8.2, 12.5, 48.1], [1, 0, 0, 0], 0.00018, 0.00020, 0.00009]]

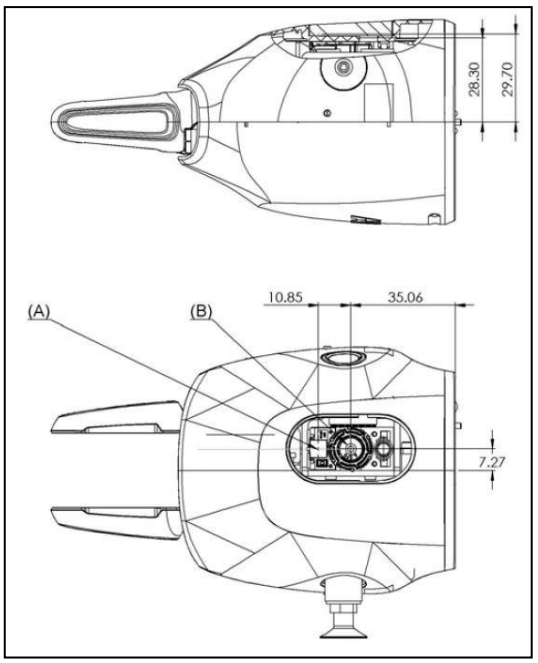
- Poids de la main selon combinaisons

Combinaison	Poids (g) sans doigts, ventouse(s) et filtre(s)[i]	Poids (g) du préhenseur intégral	Capacité de charge max. (g) sans doigts, ventouse(s) et filtre(s)[ii]	Capacité de charge max. (g) du préhenseur intégral[ii]
Asservi	215	230	285	270
Servo + Vide 1	225.5	248	274.5	252
Servo + Vide 1 + Vide 2	250	280	250	220
Servo + Vision	229	244	271	256
Servo + Vision + Vide 1	239.5	262	260.5	238

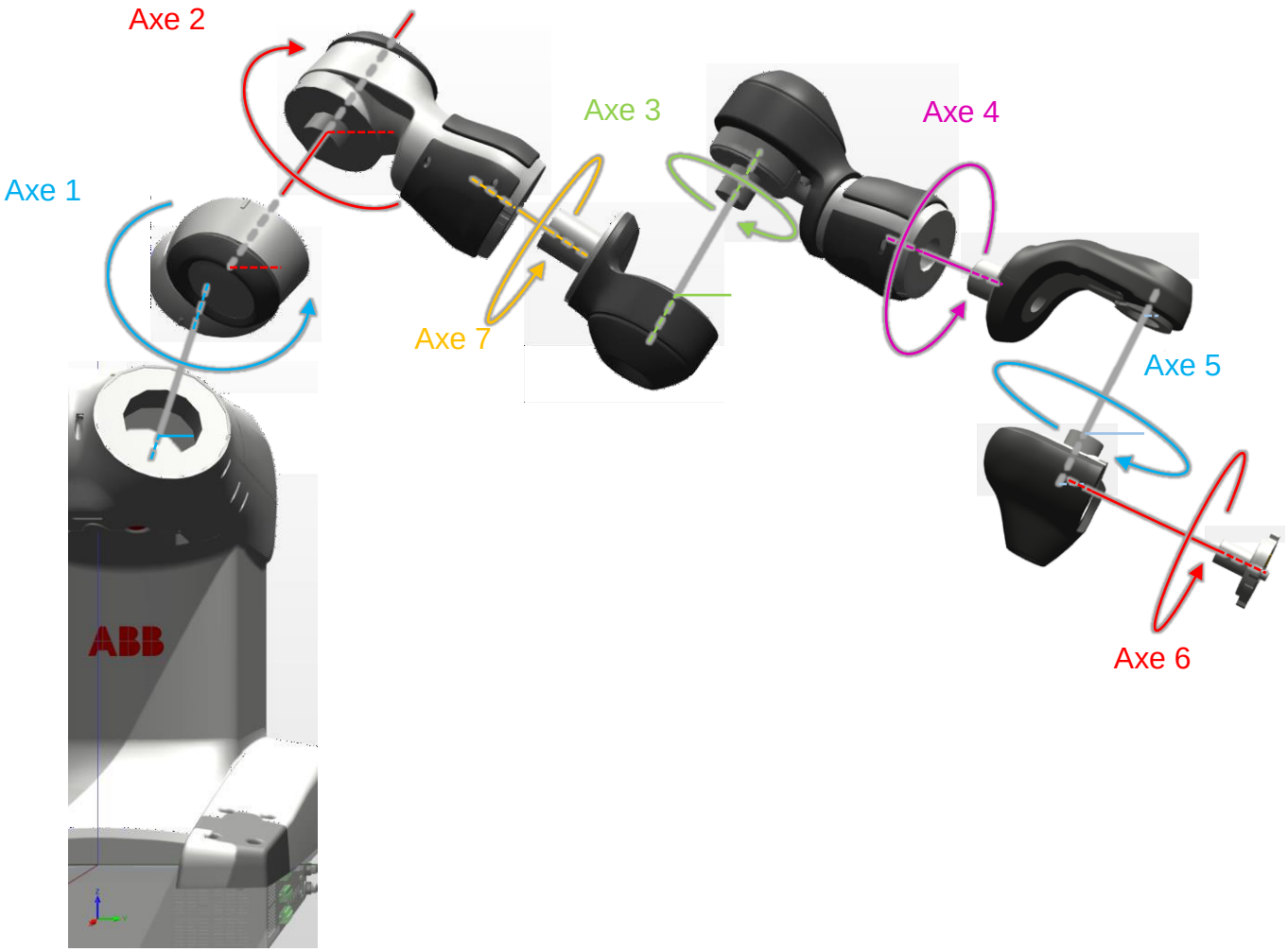
Note : Ces types de données (tooldata) portent uniquement sur la masse et le centre de gravité.

3 Plans des mains





4 Détail cinématique de YuMi®



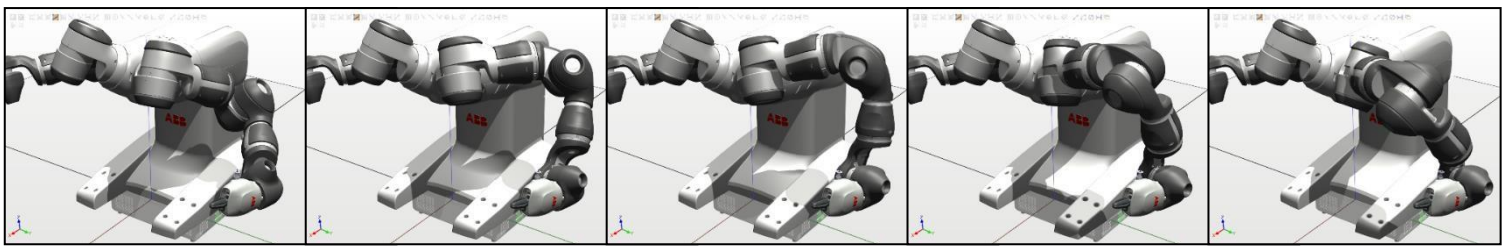
L'axe 7 se positionne entre l'axe 2 et 3 de la chaîne cinématique.

5 Importance du mouvement de coude

Grace à son 7ème axe, YUMI peut atteindre une position avec le coude orienté de différentes manières.

Il est donc important d'orienter le coude de la manière qui convient le mieux aux mouvements qui suivent ou précèdent.

Une bonne orientation du coude permettra d'obtenir un maximum de degrés de liberté.



Présentation de RobotStudio®

1 Généralités

RobotStudio® est le logiciel qui permet de travailler efficacement avec les robots ABB. Il permet de se connecter sur un vrai robot par l'intermédiaire d'une connexion Ethernet. On dit alors que l'on travail « En ligne ». Dans ce cas de figure, il est possible de :

- Créer et installer un système sur le robot
- Modifier les paramètres systèmes du robot
- Modifier le programme RAPID
- Accéder aux signaux d'entrée sortie
- Analyser les mouvements du robot par l'analyseur de signal
- Paramétrer les cartes de sécurité qui sont installées sur le robot

Il permet aussi de travailler avec un ou plusieurs robots virtuels. De ce fait, il est possible de définir l'environnement du robot en implantant dans son univers 3D des objets venant de la bibliothèque standard ou importés de fichiers sources 3D de différents formats. Cette implantation s'appelle une « Station ». Pour pouvoir organiser votre projet dans une structure de répertoire ordonnée, il est aussi possible de créer ce qu'on appelle une « Solution ». Dans ce cas de figure, il est possible de :

- Créer une Solution de projet et créer une Station
- Implanter des objets 3D dans l'environnement du robot
- Faire fonctionner un ou plusieurs robots en simulation
- Concevoir des trajectoires à partir des objets graphiques
- Créer des mécanismes complexes
- Concevoir des composants intelligents (Smart Component)
- Synchroniser tout cela sur un contrôleur de robot virtuel (un système virtuel)

Dans ce mode de fonctionnement il est important de différencier la partie qui simule le robot (le contrôleur virtuel aussi appelé système de commande virtuel) et la Station.

La station : Contient toutes les modifications en termes de programmation à apporter au robot, permet d'écrire du code trajectoire en utilisant les données graphiques utilisées dans la station.

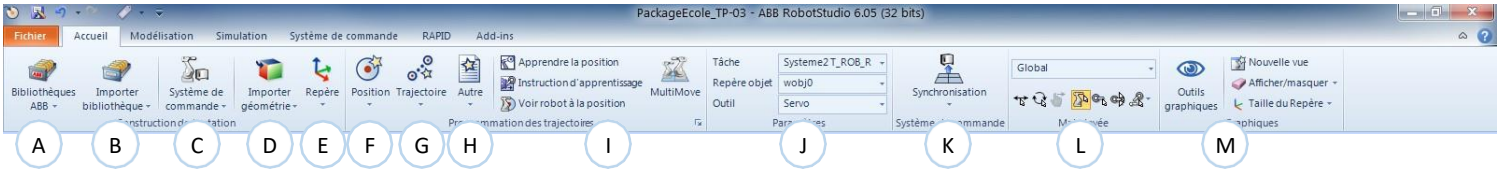
Le système de commande virtuel : Simule exactement le fonctionnement d'un robot, avec les mêmes caractéristiques de mouvement qu'un robot réel. Il exécute le programme qui y est chargé comme un vrai robot.

Ainsi il faut comprendre que pour tester ce qui est fait dans la station, il faut synchroniser les modifications vers le système de commande. De même si les modifications ont été apportées directement dans le système de commande (par l'onglet RAPID par exemple) il faut penser à synchroniser le contenu du système de commande vers la Station.

2 Onglet Accueil

Cette barre d'outils permet de gérer l'implantation dans l'espace d'objets 3D et de concevoir les trajectoires.

Cet onglet agit directement sur la station.



A. Importation d'une librairie mécanique standard

B. Importation d'une librairie standard

C. Création ajout d'un système de commande robot

D. Importation d'un objet 3D au format natif 3D (format dépendant des options RobotStudio® achetées)

E. Création d'un repère de construction

F. Création d'une position robot par pointage graphique

G. Création de trajectoire

H. Création de repère outil, repère de travail ou ajout d'une instruction RAPID autre

I. Activation de la visualisation du robot à la position (si atteignable) OU ajout d'une instruction d'apprentissage de position OU apprentissage de la position courante (sans instruction)

J. Activation de la tache courante, de l'outil courant et du repère de travail courant.

K. Synchronisation entre le RAPID et la STATION

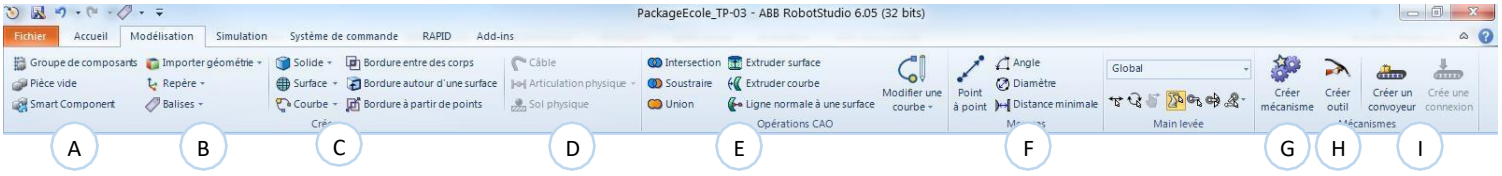
L. Manipulation des objets 3D en : déplacement linéaire, en orientation et si unité mécanique, mouvement linéaire et réorientation, ainsi que pilotage d'axes et multi robot. Définition du repère de référence pour ces déplacements.

M. Paramétrage de l'affichage graphique.

3 Onglet MODÉLISATION

Cette barre d'outils permet de créer et de gérer des éléments 3D basiques ainsi que des éléments mécaniques.

Cet onglet agit directement sur la station.



A. Création d'un objet vide, d'un groupe d'objet vide ou d'un « Smart Component »

B. Importation d'un objet 3D au format natif 3D (format dépendant des options RobotStudio® achetées)

C. Création d'objet 3D basique (objet non redimensionnable après création)

D. Ajout d'éléments liés aux lois physiques définies dans la station.

E. Manipulation des objets 3D.

F. Outils de mesures (distance, diamètre, angles, …)

G. Outil de conception de mécanisme

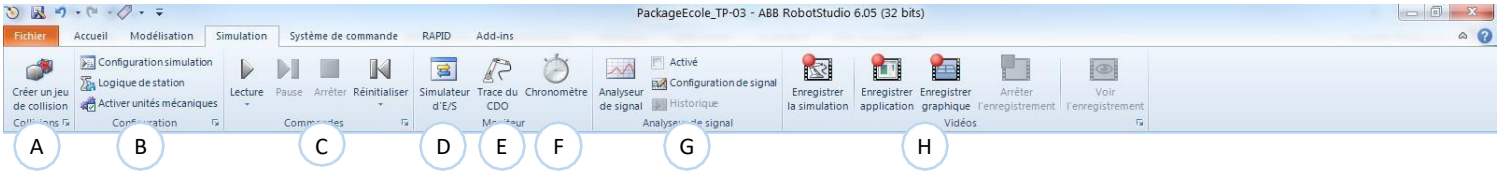
H. Outil de conception d'outils robot

I. Outils de conception de convoyeur

4 Onglet SIMULATION

Cette barre d'outils permet de paramétrer et lancer la simulation ou encore d'analyser les mouvements du robot. Il est possible aussi de faire une vidéo à partir de la simulation ou un enregistrement 3D.

Cet onglet agit directement sur la station.



A. Outils de création de jeu de collision

B. Paramétrage de la station (tache à exécuter, mode de fonctionnement cyclique ou non, …). Paramétrage de la logique de station avec les Smart Components qui peuvent la composer.

C. Table de lecture et arrêt de la simulation

D. Affichage du simulateur de signaux d'entrée et de sortie du système de commande

E. Affichage de la trace du centre d'outil dans l'environnement 3D

F. Activation d'un chronomètre

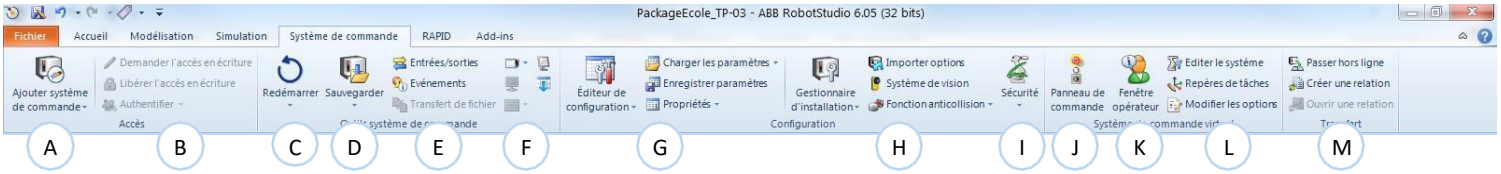
G. Analyseur de signaux.

H. Outils d'enregistrement de vidéo de la simulation.

5 Onglet SYSTÈME DE COMMANDE

Cette barre d'outils permet de paramétrer et lancer la simulation ou encore d'analyser les mouvements du robot. Il est possible aussi de faire une vidéo à partir de la simulation ou un enregistrement 3D.

Cet onglet agit directement sur le système de commande.



A. Permet de démarrer un système de commande virtuel ou de se connecter à un système de commande réel

B. Permet de gérer les droits d'accès au système de commande. Pour travailler sur un système de commande, il faut demander les accès en écriture (sauf sur un robot virtuel à moins que le pupitre virtuel ne soit affiché).

C. Permets de redémarrer le système de commande selon le mode de redémarrage souhaité.

D. Permets de charge ou de restaurer une sauvegarde aussi appelé Backup.

E. Affichage des fenêtres d'entrée sortie ou les archives. Si le système est en ligne, permet aussi le transfert de fichiers.

F. Affichage du pupitre virtuel (si robot virtuel, alors ce sera un pupitre interactif. Si le système est en ligne, alors ce ne sera qu'un affichage de l'écran du pupitre sans interaction)

G. Raccourcis vers les paramètres systèmes du robot.

H. Permet de créer des systèmes réels ou virtuels à partir des clés de licence ABB fourni. Permets aussi de modifier les options sur les systèmes déjà créés. Contient l'outil de migration de fichiers de la version RW 5.xx vers la version RW 6.xx.

I. Affichage de l'outil de configuration des cartes de sécurité si installée (SafeMove ou EPS)

J. Affichage du panneau de commande (pour robot virtuel)

K. Permets d'exécuter des programmes RAPID interactif sans démarrer le FlexPendant (pupitre) virtuel

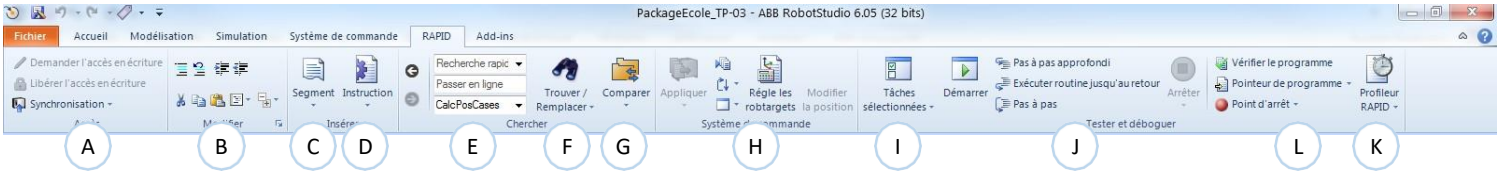
L. Modification de certaines options du robot.

M. Permets de créer une station virtuelle à partir d'un robot réel.

6 Onglet RAPID

Cette barre d'outils permet d'accéder à de nombreux outils simplifiant la programmation RAPID. L'arborescence associée permet un accès rapide aux différentes routines des différents modules

Cet onglet agit directement sur le système de commande.



A. Permets d'obtenir les accès et de synchroniser le code RAPID modifié avec celui de la station active.

B. Outils de mise en forme (copier, coller, commenter, mettre en forme, …).

C. Permets d'accéder à des morceaux de code pré-écris ou d'en ajouter de nouveaux.

D. Liste les instructions RAPID disponible par famille. Écrit l'instruction souhaitée avec la syntaxe exacte. Ne reste qu'à renseigner les arguments nécessaires.

E. Outil pour se déplacer facilement dans un module.

F. Outil de recherche.

G. Outil de comparaison.

H. Paramétrage du mode de fonctionnement. Réglage de positions si changement de repère.

I. Sélection des taches robot à exécuter.

J. Gestion du démarrage des programmes robots (si accès en écriture).

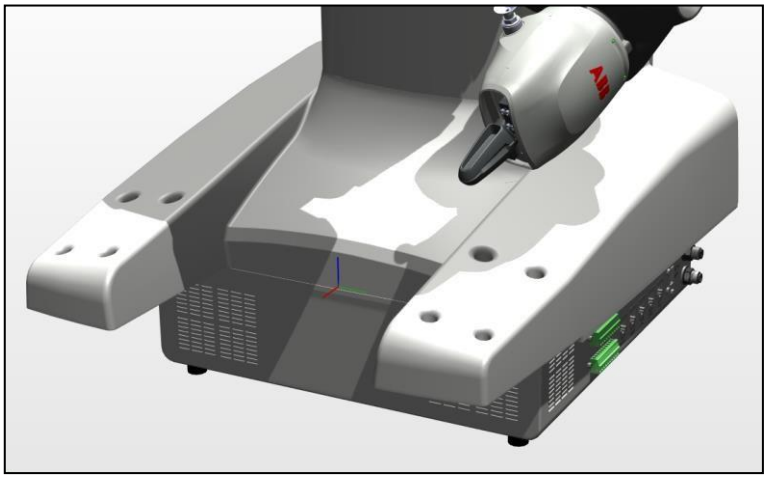
K. Outil de vérification du code RAPID. Choix du mode de suivi du pointeur programme.

L. Profileur RAPID qui génère un fichier espion.

Création d'un repère de travail dans RobotStudio®

1 Utilité du repère de travail ?

Un repère de travail permet de définir des positions dans l'espace. Il existe un repère de travail originel au robot qui s'appelle « Repère Atelier ».



Il est placé entre les deux systèmes de fixation du robot YuMi. Pour d'autre type de robot, il est confondu avec la Base du robot.

L'axe Z est dirigé vers le haut, le X vers l'avant du robot et le Y vers la gauche.

Ce repère est une donnée RAPID représenté par le nom « wobj0 ».

Tous les repères appris dans l'environnement du robot seront tous positionné par rapport à ce repère.

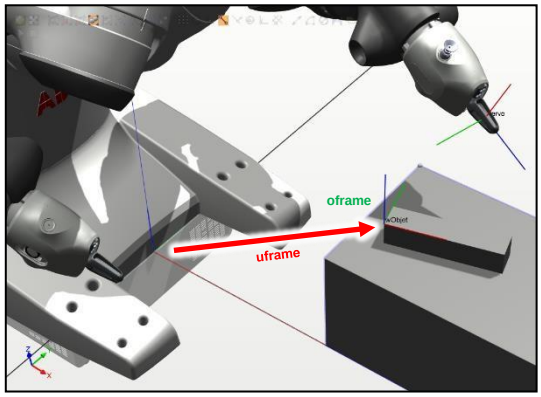
Un repère permet donc d'apprendre des positions par rapport à des repères physiques du montage. Si le montage est déplacé, il suffit de réapprendre ce repère pour que l'ensemble des positions l'utilisant soient directement recalée dans l'espace.

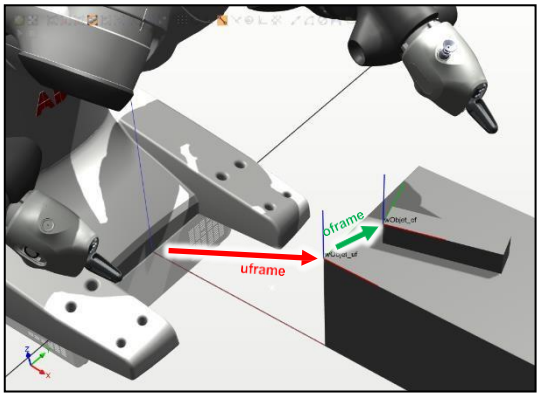
Avant donc d'apprendre des positions il est important de définir le repère de travail qui servira à les repositionner le cas échéant.

2 Repère utilisateur et repère objet

Pour positionner la zone de travail, un repère de travail dispose d'une composant Translation/rotation nommée **Repère Utilisateur** (uframe) qui est défini depuis le repère wobj0.

Mais pour disposer un objet dans la zone de travail, il est possible d'ajouter une composante supplémentaire nommée **Repère Objet** qui se défini dans le repère de travail (utilisateur). Si la pièce n'est pas amenée à être recalée lors de la production, cette composante n'est pas nécessaire.





Structure de programmation RAPID

1 Les instructions

- Un "programme de production" est constitué d'une succession d'instructions parcourues une par une par le pointeur programme (**PP**) lors de l'exécution du programme.
- Une instruction correspond à une opération élémentaire
- On ne peut mettre une instruction que dans une routine (entre un PROC / ENDPROC, un FUNC / ENDFUNC ou un TRAP / ENDTRAP).
- Les instructions permettent de déplacer le robot, de dialoguer avec l'environnement et avec l'opérateur. Elles sont exécutées une par une
  - Exemples :

    Arrêt de l'exécution du programme : **Stop;**

    Déplacement du robot vers un point : **MoveL … , … … ;**

```
MoveL pCase9,v100,fine,Servo\WObj:=wPlateau;
```

2 Les routines

- Une routine regroupe un ensemble d'instructions
- Le programmeur décompose son programme en un ensemble de routines
- Chaque routine permet de décrire une fonctionnalité ou une phase du programme. Elle comprend une suite d'instructions

DeposeMag

Instruction ;
Instruction ;
Instruction ;
…

FermetureP

Instruction ;
Instruction ;
Instruction ;

ControlePiece

Instruction ;
Instruction ;
Instruction ;
…

Traj58

Instruction ;
Instruction ;
ControlePiece ;
Instruction ;
…

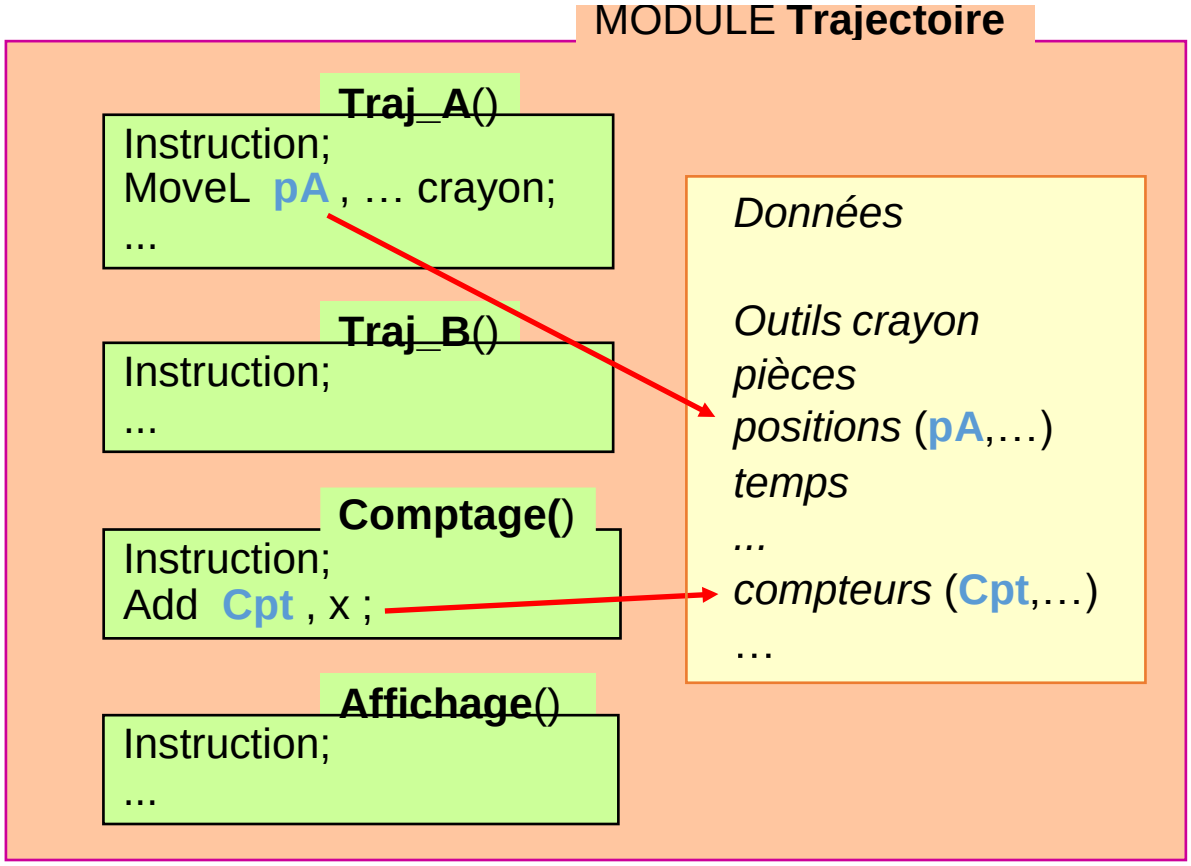
PrisePiece

Instruction de mouvement ;
FermetureP ;
Instruction ;
Traj58 ;
…

```
PROC CalcPosRatelierB()
    pRatelierB2 := Offs(pRatelierB1,0,40,0);
    pRatelierB3 := Offs(pRatelierB1,0,80,0);
    pRatelierB4 := Offs(pRatelierB1,0,120,0);
    pRatelierB5 := Offs(pRatelierB1,0,160,0);
ENDPROC
```

3 Les modules et les données

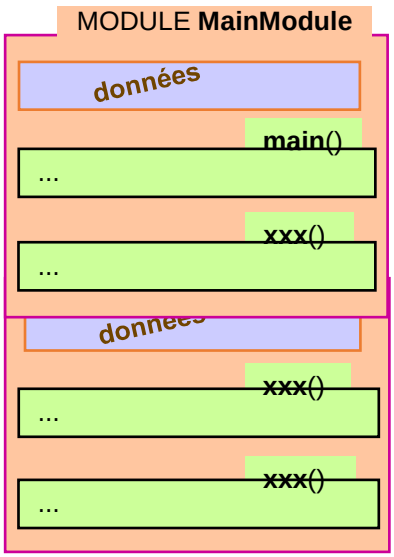
- Ces **routines** sont regroupées dans un **module**
- Les **données** utilisées par les instructions sont également stockées dans les **modules**

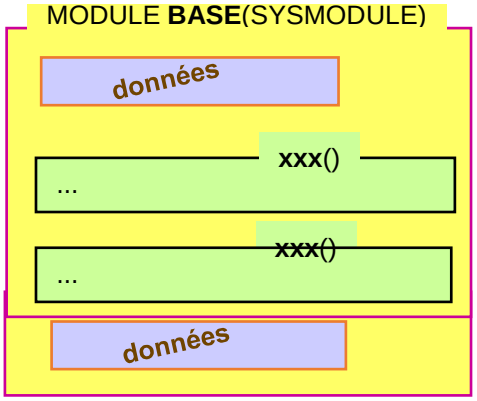


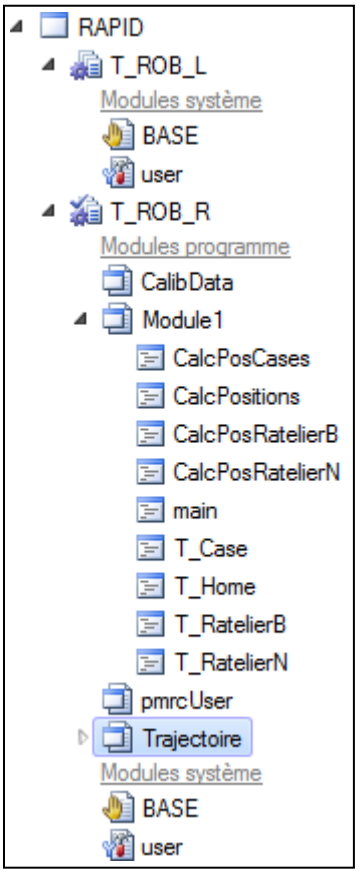

```
MODULE Trajectoire

    PERS robtarget PA:=[[40.113011661,200.09976423,30],[0,0.382790696,0.923835095,0],[1,-2,1,4],[-147.945205479,9E+09,9E+09,9E+09,9E+09,9E+09]];
    PERS robtarget PB:=[[80.113011661,200.09976423,30],[0,0.382790696,0.923835095,0],[1,-2,1,4],[-147.945205479,9E+09,9E+09,9E+09,9E+09,9E+09]];
    PERS robtarget PC:=[[120.113011661,200.09976423,30],[0,0.382790696,0.923835095,0],[1,-2,1,4],[-147.945205479,9E+09,9E+09,9E+09,9E+09,9E+09]];
    VAR num Cpt;

    PROC Traj_A()
        MoveL PA, v1000,z50,tool0;
        MoveL PB, v1000,z50,tool0;
        MoveL PC, v1000,z50,tool0;
    ENDPROC

    PROC Comptage()
        Add Cpt, 5;
    ENDPROC

ENDMODULE
```

4 La tâche et le programme

- Il existe 2 types de modules : les modules programmes et les modules systèmes. Une tache regroupe les modules programmes et/ou les modules systèmes
  - Les modules de type programmes sont :
    - en général liés à la production
    - Ils seront supprimés de la mémoire lors :
      - du chargement d'un autre programme
      - de la création d'un nouveau programme
      - de la fermeture du programme
  - Les modules de type systèmes sont
    - en général liés au robot et à son environnement.
    - Le contenu est commun à tous les programmes puisque le chargement de programme ne déchargera pas les modules systèmes de la mémoire
- Un programme regroupe l'ensemble des modules programmes. Cette notion est utilisée lors des sauvegardes et chargements







Les instructions de mouvement RAPID

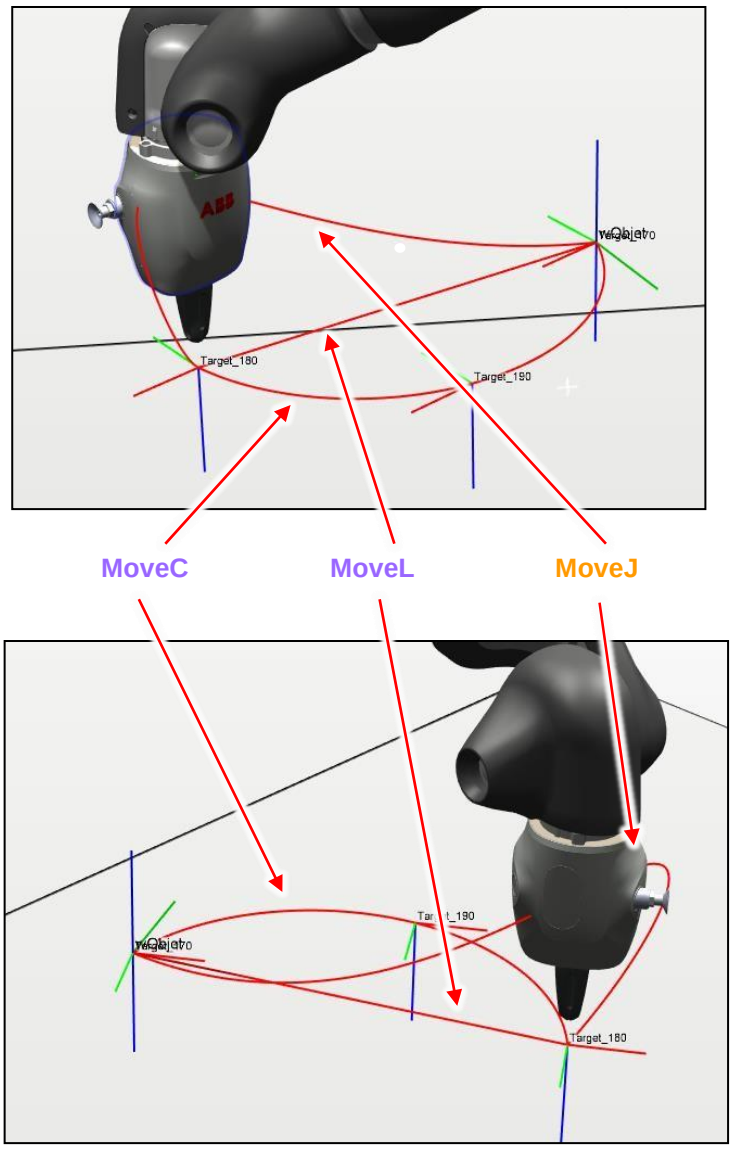
1 Les instructions de mouvement

Il existe 3 instructions de mouvement selon 3 types de déplacement différents :

MoveJ : Les mouvements des différents axes sont synchronisés pour amener. le robot sur la position visée. Le tronçon est libre. Les axes démarrent et terminent en même temps leur mouvement. Le temps du mouvement est donc imposé par l'axe le plus lent. Les éléments portés par le poignet du robot ont donc un déplacement quelconque dans l'espace avec une vitesse difficilement déterminable.

MoveL : L'outil est déplacé linéairement avec une vitesse à peu près constante.

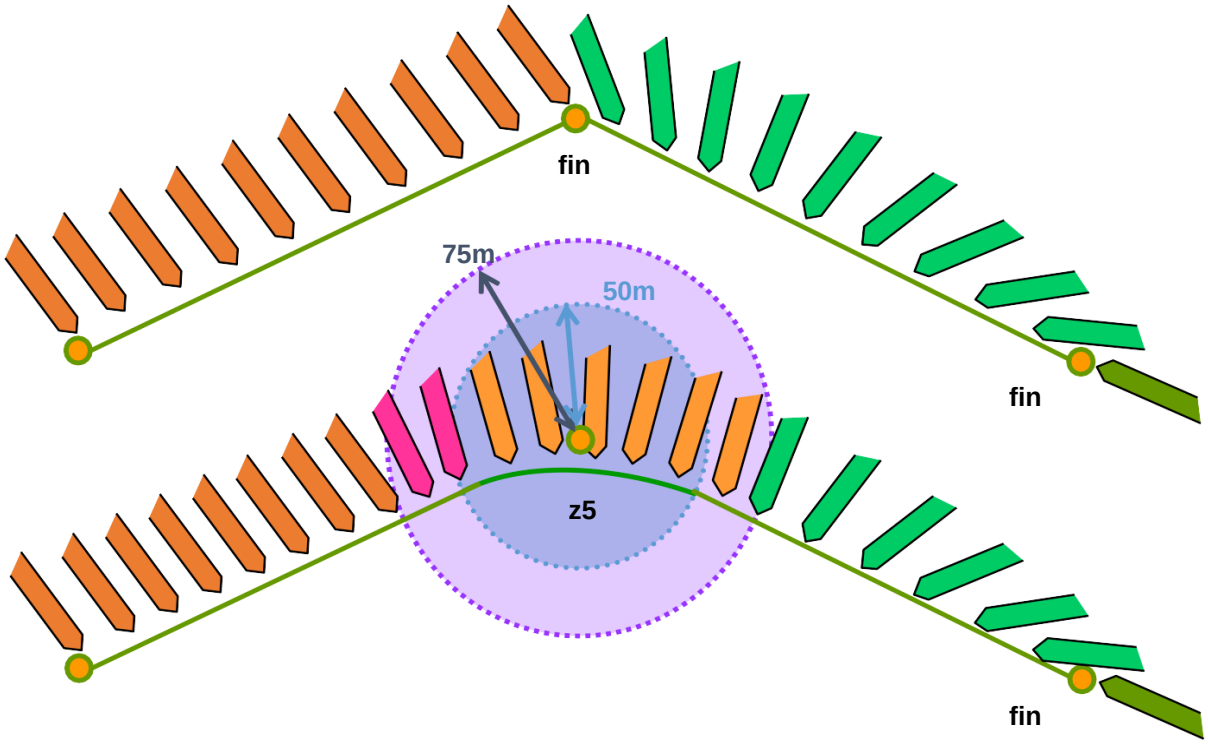
MoveC : L'outil décrit un arc de cercle entre sa position de départ, un point intermédiaire et la position d'arrivée.



2 Les arguments de base des instructions de mouvements

Arguments/Données						
	robtarget	**robtarget**	**speeddata**	**zonedata**	**tooldata**	**\option wobjdata**
Instructions	Position et orientation de l'outil		Vitesses du centre outil et de l'orientation de l'outil	Zone d'anticipation vers le mouvement suivant	Dimension, orientation, poids et centre de gravité de l'outil	Origine du calcul de position de l'outil (wobj0 est le repère Atelier par défaut)
MoveJ	*		, v1000	, fine	**, fraise**	
MoveL	pt1		, v2000	, z0	**, fraise**	\WObj:= **tablette**
MoveC	*	, pt2	, v1500	, z1	**, fraise**	

3 La représentation d'une zone



Les données

1 Définition et types

1) Une donnée est une mémoire nommée qui permet de stocker une information sous un format spécifique.

Exemples :

- Un **compteur** est une mémoire de type **num** (numérique). Elle sera composée d'un simple chiffre décimal signé.
- Un **choix opérateur** (vrai ou faux) sera de type **bool** (booléen). Cette mémoire prendra la valeur TRUE ou FALSE.
- Un **message à afficher** sera de type **string** (chaine de caractère). Elle contiendra des caractères entre " ".

Ces **données** sont la base élémentaire de la programmation.

2) L'ensemble des autres données RAPID sont la combinaison élémentaire de ces 3 types.

- Une position XYZ du robot est de type **pos** : *[num, num, num]*
- Une orientation est de type **orient** : *[num, num, num num]*
- Une configuration d'axe est de type **confdata** : *[num, num, num, num]*
- Les valeurs d'axe externe sont de type **extjoint** *[num, num, num, num, num, num]*

3) Certaines données imbriquent ainsi différents types de données pour être formées :

- Un repère dans l'espace (représenté par le type **pose**) est constitué d'un **pos** et d'un **orient**.
  Il sera composé comme suit :
  [pos , orient]
  ou aussi noté comme ceci :
  [[num, num, num], [num, num, num, num]] (les sous-types sont contenus entre crochets)
- Une position robot (représenté par le type **robtarget**) est constituée d'un **pos**, un **orient**, un **confdata** et un extjoint.
  Elle sera composée comme suit :
  [pos , orient , confdata , extjoint]
  ou aussi noté comme ceci :
  [[num, num, num] , [num, num, num, num] , [num, num, num, num] , [num, num, num, num, num, num]]
- Un repère de travail (représenté par le type **wobjdata**) est constitué de deux **bool**, un **string**, et de deux **pose**
  Il sera composée comme suit :
  [bool , bool , string , pose , pose]
  ou aussi noté comme ceci :
  [bool, bool, string , [[num,num,num] , [num,num,num,num]] , [[num,num,num] , [num,num,num,num]]]
- Les valeurs de chaque élément et leurs significations sont disponibles dans la notice RAPID dans la partie « Types de données »

2 La nature

Une donnée est définie aussi par sa nature.

La nature d'une donnée définie comment cette donnée peut évoluer en cours d'exécution du programme.

1) Les **CONSTANTES**

Une donnée de nature **CONSTANTES** possède une valeur définie lors de sa déclaration. Cette valeur ne peut pas être modifiée par une instruction programme. On ne trouvera donc jamais une constante à gauche du symbole « := ».

Une donnée **CONSTANTE** peut servir, par exemple, de référence de comparaison.

2) Les **VARIABLES**

Une donnée de nature **VARIABLES** possède une valeur définie lors de sa déclaration. Cette valeur peut être modifiée par une instruction programme.

La **VARIABLE** reprendra sa valeur de déclaration quand le programme sera réinitialisé. (« PP vers Main » ou « PP vers Routine »).

3) Les **PERSISTANTES**

Une donnée de nature **PERSISTANTES** possède une valeur définie lors de sa déclaration. Cette valeur peut être modifiée par une instruction programme.

Lors d'une réinitialisation du programme, la **PERSISTANTES** garde sa valeur courante. Le seul moyen d'initialiser une donnée **PERSISTANTE** est d'utiliser « := ».

Si une donnée **PERSISTANTE** est déclarée avec le même nom et la même valeur initiale dans plusieurs tâches, alors cette donnée est automatiquement partagée entre toutes les tâches. Si la donnée évolue dans une tache, les autres tâches pourront voir l'évolution de la donnée. Cette fonctionnalité est très pratique pour partager des informations de n'importe quel type entre deux tâches.

3 La portée

La portée d'une donnée définit la zone de programme où la donnée peut être lue :

1) GLOBALE

La donnée peut être accessible de toutes les routines de n'importe quel module de la tâche courante.

2) **LOCAL** (module)

La donnée ne sera accessible que depuis les routines du module dans lequel sa déclaration est stockée.

3) **LOCAL** (routine)

La donnée ne sera accessible que depuis la routine dans laquelle sa déclaration est stockée.

4) TASK

La portée **TASK** n'est utile que pour les **PERSISTANTES**. En définissant une **PERSISTANTE** en mode **TASK**, alors si un lien existe avec une donnée de même nom dans une autre tâche, ce lien est détruit. Les deux données

ne seront plus liées et pourront évoluer librement l'une par rapport à l'autre.

La déclaration d'une donnée doit obligatoirement être stockée dans un module. Le choix est donné de ranger cette déclaration où bon vous semble. Faire cependant attention à la portée de la donnée

TP

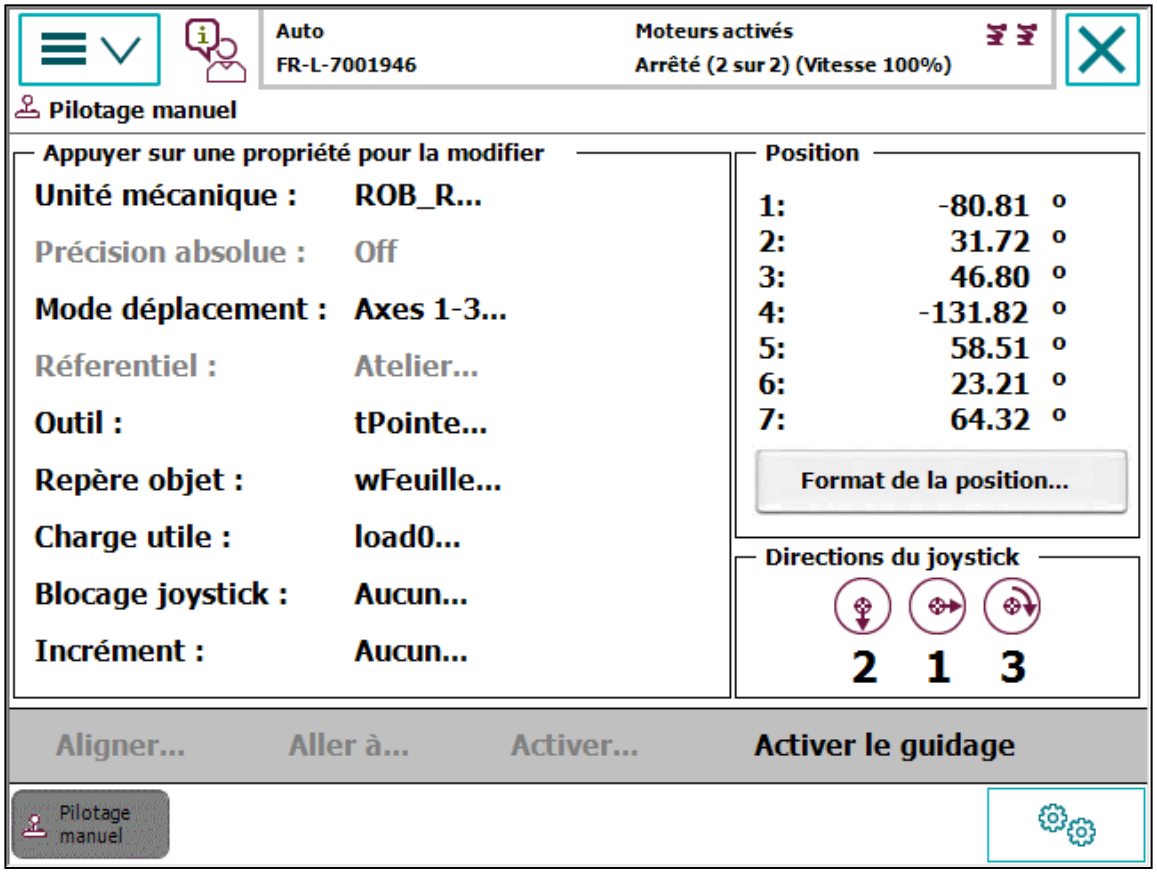
Présentation

YuMi est un robot collaboratif dont l'utilisateur programme la position des bras et l'activation des mains selon son désir.

La mémorisation des positions de bras nécessite que l'utilisateur amène au préalable les bras dans la position à apprendre ou qu'il récupère le travail effectué sous RobotStudio.

Le but de l'exercice sera de comprendre comment le robot s'articule et d'apprendre à déplacerun robot ABB avec le joystick ou en déplaçant le robot à la main.

Vous pourrez comprendre comment le robot se repère dans l'espace.



Partie 1

1) Prendre en main le pupitre

Prise en main, exploration du pupitre. Ouverture de la fenêtre de pilotage manuel.

2) Piloter le robot manuellement axe par axe

Faire bouger les axes d'un bras robot pour rejoindre une position et une orientation en utilisant le pilotage axe par axe.

3) Piloter dans les repères cartésiens

Amener précisément le second bras sur une autre position dans l'espace en utilisant le pilotage linéaire et la réorientation.

4) Piloter du robot par le guidage manuel

Activer le guidage manuel des bras.

5) Vérifier de la synchro des bras

Placer le robot en position de synchronisation. Vérifier que les repères de chaque axe sont bienen face.

Effectuer une synchronisation des axes (attention, pas d'étalonnage !) par la méthode « d'appel de l'étalonnage ».

Partie 2 : Découverte du RAPID & Conception des trajectoires

6) Subdiviser la trajectoire

Par l'intermédiaire de l'éditeur RobotStudio®, accéder à la trajectoire disponible. Créer les routines suivantes

nommées :

- « main »
- « T_Cases »
- « T_RatelierB »
- « T_RatelierN »
- « T_Home »

Dans chacune de ces routines, y placer les instructions de mouvement s'y référant.

Utiliser la routine « main » pour appeler les autres routines et permettre un enchainement de trajectoires.

7) Ajouter l'approche et le dégagement

Pour chaque position, ajouter un mouvement vers une position d'approche et de dégagement qui sera relative à la position ciblée. Le dégagement devra se faire dans l'axe de prise avec un recul de 50 mm.

Utiliser les fonctions dédiées à ces calculs. Appliquer une zone de passage de 10 mm sur ces positions ajoutées.

8) Synchroniser vers la station

Synchroniser les modifications apportées dans le système de commande vers la station de travail.

Partie 3 : Découverte du RAPID et pilotage de YUMI

9) Piloter le YuMi®

Piloter les bras du robot YuMi en utilisant les outils de pilotage linéaire, réorientation et mouvement du coude.

Activer les différents repères outils disponible sur chaque main et constater la modification auniveau des mouvements.

10) Apprendre repères outils

Définir la position des repères outil préhenseur avec les mors 3D importés.

Définir la position du repère outil pointe qui servira au palpage du repère objet sur le vrai robot. (À ne faire que sur un seul bras)

11) Apprentissage du repère de travail

Définir la position du repère de travail défini par les marques X1, X2 et Y1. Ce repère doit êtredisponible pour les deux bras.

12) Création de positions robot

Créer les 19 positions cartésiennes (5 du côté blanc, 5 du côté noir et 9 au centre) dans le repère du plateau et une position articulaire dans le repère de l'atelier. Vérifier l'orientation des positions afin que la main du robot soit bien orientée sur chaque emplacement.

Renommer chaque position en « pRatelierB1 » à « pRatelierB5 », « pRatelierN1 » à « pRatelierN5 », « pCase1 » à « pCase9 » et la position home en « pHome ».

13) Conception d'une trajectoire

Créer une nouvelle trajectoire.

Ajouter dans la trajectoire une instruction de mouvement absolue en « MoveAbsJ » vers la position « pHome ».

Ajouter ensuite des instructions de mouvement en « MoveL » vers les différentes positions apprises. (Râtelier puis cases). Les mouvements vers ces positions devront être précis et la vitesse de déplacement du robot devra être de 100 mm/s.

Vérifier la configuration des axes pour les positions de la trajectoire. (Clic droit sur la trajectoireConfiguration\configuration des axes auto).

Choisir la bonne configuration en fonction de la position « pHome » choisie afin d'atteindre toutes les positions.

14) Simulation des mouvements

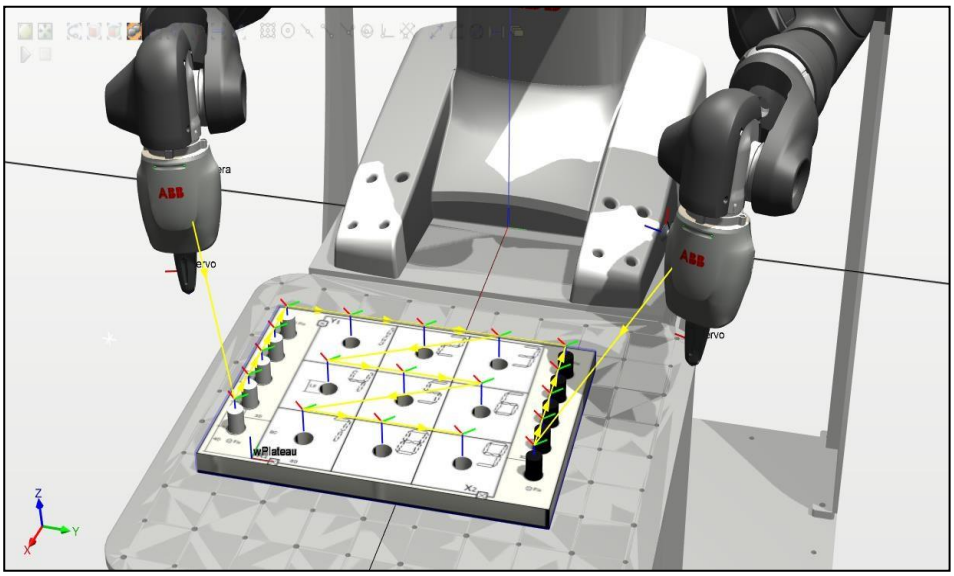
Synchroniser la trajectoire préparée dans la station avec le système de commande. Paramétrer la simulation pour lancer la trajectoire lors du lancement de la simulation.

Vérifier que le robot accède correctement à toutes les positions sans trop s'orienter après le « pHome ».

15) Copie de trajectoire

Copier la trajectoire que nous venons de créer pour l'autre robot.

Paramétrer la simulation pour lancer la trajectoire lors du lancement de la simulation uniquement sur l'autre robot.



Partie 4 : Synchronisation de mouvements multi robot

16) Charger les programmes dans les bras du robot

Prendre le temps d'analyser et de comprendre le programme.Vérifier et corriger les positions si nécessaire.

Créer un nouveau module nommé MULTIMOVE qui contiendra toutes les routines et données nécessaires à ce TP.

17) Mouvements indépendants

Modifier le programme pour que chaque bras prenne une pièce dans le râtelier de sa couleuret la dépose dans un emplacement libre du jeu.

Ensuite il devra la reprendre et la replacer dans le râtelier.

Le début de la prise de chaque bras devra toujours commencer simultanément.

18) Mouvements synchronisés

Synchroniser les mouvements entre la prise d'un bras et la dépose de l'autre bras et vice versa.

Nota : les mouvements sont synchronisés quand un mouvement d'un bras est associé à un mouvement de l'autre bras. Visuellement, on s'en rend compte car ses mouvements commencent et finissent en même temps.

19) Transfert de pions (Bonus)

Modifier le programme pour que le robot déplace automatiquement tous les pions noirs dans l'atelier blanc et inversement.

20) Echange de Pion (Bonus)

Modifier le programme pour que le robot passe la pièce d'une main à l'autre main et inversement.