# Prise en main du robot Stäubli TX2-40/Cs9
# Teach pendant SP2, Programmation VAL 3

Jean-Louis Boimond
Université d'Angers

**Mots-clés :** robot Stäubli, bras TX2-40, contrôleur Cs9, teach pendant SP2, programmation VAL 3.

Ce document propose une initiation au robot Stäubli TX2-40, équipé d'un contrôleur Cs9, en utilisant le *Teach Pendant* SP2 (et non le logiciel *Stäubli Robotics Suite* (SRS)) en abordant les thèmes suivants : la mise en mouvement de manière manuelle du bras ; l'accès aux coordonnées du *Tool Center Point* ; l'utilisation d'une application VAL 3 (VAL 3 étant le langage de programmation de Stäubli) permettant la mise en mouvement de manière automatique du bras avec une brève présentation sur les variables (les types standards, leurs créations et leurs initialisations) ; la mise en mouvement de manière manuelle du bras vers un point articulaire ou cartésien donné.

## Table des matières

# 1) Prise en main du robot

## 1.1) Démarrage du contrôleur Cs9

La mise en route du contrôleur Cs9 (utilisé avec le bras du robot) se fait en commutant sur la position « 1 » le sectionneur général situé sur la face avant du contrôleur, entouré en rouge dans la figure qui suit.


Figure 1 : Face avant du contrôleur Cs9.

Attendre environ 2 $mn$ l'affichage du menu principal sur le *Teach Pendant*, comme illustré dans la figure qui suit.

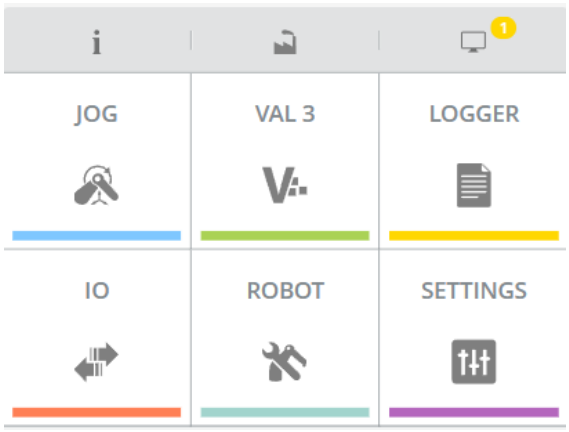



Figure 2 : Menu principal du *Teach Pendant*.

**N.B. :**

- La touche **Home**  , située en haut à gauche du *Teach Pendant*, permet de revenir au menu principal,
- La touche **Back**  , située vers le haut à gauche du *Teach Pendant*, permet de revenir à la page précédemment atteinte.

## 1.2) Mise sous puissance du bras avec un Mode de Marche manuel

**Hypothèse :** Le contrôleur Cs9 est en fonctionnement (voir **1.1**).

La mise sous puissance du bras du robot, nécessaire à sa mise en mouvement, avec un Mode de Marche manuel se fait à travers les deux étapes qui suivent :

a) Sur le Sélecteur de Mode de Marche (boitier appelé WMS9 situé près du contrôleur), veillez à ce que le commutateur (muni d'une clé) soit mis sur **manuel** comme cela est décrit dans la figure qui suit.


Figure 3 : Sélecteur de Mode de Marche (WMS9).

Il en résulte l'apparition de l'icône en bas à droite du *Teach Pendant* indiquant la sélection du Mode de Marche **manuel**. Ce mode est tel que la vitesse du robot est limitée à $250\ mm/s$, ce qui permet à l'opérateur de se tenir à proximité du bras du robot.

**Remarque :** Une alternative à l'utilisation du WMS9 consiste à sélectionner l'item ***manual slow*** , dans le menu déroulant situé en bas à droite du *Teach Pendant*, plutôt que l'item ***auto*** (donné par défaut), voir la figure qui suit :

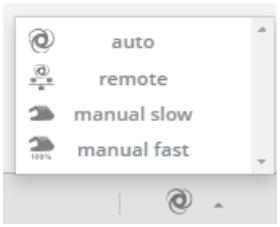



Figure 4 : Sélection du Mode de Marche à partir du *Teach Pendant*.

b) Appuyez sur la touche **Power** située en haut à droite du *Teach Pendant* pour mettre le bras sous puissance. Cette action n'est prise en compte que si le « corps mort » (*enabling device*), situé sur la face arrière du *Teach Pendant* (entouré en rouge dans la figure qui suit), a été amené dans sa position médiane (à travers un appui ni trop faible, ni trop fort, sur le bouton) au cours des 15 dernières secondes (notez qu'il faut relâcher le bouton puis appuyer à nouveau dessus si le bras n'a pas été mis sous puissance durant les 15 secondes). Un voyant entourant la touche **Power** apparaît (en clignotant pendant quelques secondes avant d'être fixe) pour indiquer que la puissance est mise sur le bras.

Figure 5 : Face arrière du *Teach Pendant*.

**N.B. :** La puissance sur le bras est coupée si le 'corps mort' est relâché alors que le bras est en mouvement manuel. Afin de remettre la puissance sur le bras, il faut appuyez sur le bouton bleu **Restart** du boitier WMS9 (voir Figure 3) afin d'acquitter le redémarrage avant d'appuyer sur la touche **Power**.

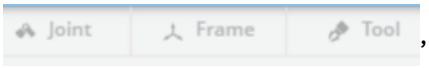
**N.B. :** Un bouton d'**Arrêt d'urgence**, situé en haut à droite du *Teach Pendant*, permet une coupure immédiate de la puissance sur le bras et donc un arrêt du mouvement du bras (s'il était en mouvement).

## 1.3) Mise en mouvement de manière manuelle du bras

**Hypothèse :** Le bras est sous puissance avec un Mode de Marche **manuel** (voir **1.2**).

**N.B. :** Il est possible de régler la vitesse du Tool Center Point (TCP) en appuyant sur la touche **Jog**  , située dans le bandeau vertical de droite du *Teach Pendant*. Sa valeur en pourcentage apparaît en bas à gauche, avec une vitesse maximale ($100\%$) égale à $250\ mm/s$.

Appuyez sur le bouton **JOG**  du menu principal du *Teach Pendant* (accessible - si vous n'y êtes pas déjà - en appuyant sur la touche **Home**  , située en haut à gauche) pour accéder à la fenêtre permettant un déplacement manuel du bras.

Une fois le menu **JOG** apparu, sélectionnez un des boutons représentés dans le bandeau horizontal  , situé en haut, pour indiquer l'espace dans lequel vous souhaitez effectuer le déplacement :

- le bouton **Joint** pour accéder à l'**espace articulaire** (*joint space*),
- le bouton **Frame** pour accéder à l'**espace cartésien** (*Cartesian space*) associé au repère $R_0$ de base du bras du robot,
- le bouton **Tool** pour accéder à l'**espace outil** (*tool space*) associé au repère $R_{Tool}$ associé à l'outil (rattaché à la bride du bras du robot).

### 1.3.a) Dans l'espace articulaire

Un appui sur le bouton **Joint** permet d'effectuer un mouvement du bras dans l'**espace articulaire** à travers les angles **J1**, **J2**, ..., **J6**, voir la figure qui suit.

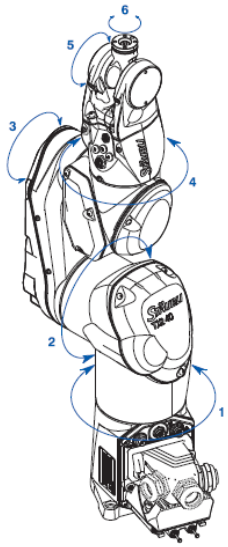



Figure 6 : Description des six articulations du bras TX2-40.

Appuyez sur la touche **Jog**  (située dans le bandeau vertical de droite du *Teach Pendant*), par exemple, relativement à l'articulation **J1** afin que le bras effectue une rotation autour de l'axe de **J1**, au choix dans le sens négatif ou positif.

**N.B. :** Relativement aux informations affichées, par défaut, dans la fenêtre principale :
- **flange** (équivalent à **flange[0]**) indique qu'aucun outil n'est sélectionné, ce qui fait que le repère outil $R_{Tool}$ coïncide avec le repère associé à la bride du bras du robot,
- **world** (équivalent à **world[0]**) indique que le repère de référence (notamment utilisé pour *situer* des *points*, des repères) est le repère world (lequel coïncide avec le repère $R_0$ de base du bras du robot).

## 1.3.b) Dans l'espace cartésien

Un appui sur le bouton **Frame** permet d'effectuer un mouvement du bras dans l'**espace cartésien** à travers les boutons **X**, **Y**, **Z** (en mm), **RX**, **RY**, **RZ** (en degré). Appuyez sur la touche **Jog**  (située dans le bandeau vertical de droite du *Teach Pendant*), par exemple, relativement à l'axe **X** afin que le bras effectue une translation du TCP le long de l'axe $x_0$ du repère $R_0$ de base du bras du robot. Lorsque la touche **Jog** est relative à **RX**, **RY** ou **RZ**, le TCP effectue une rotation autour des axes $x_0$, $y_0$ ou $z_0$.

## 1.3.c) Dans l'espace outil

Un appui sur le bouton **Tool** permet d'effectuer un mouvement du bras dans l'**espace outil** à travers les boutons **X**, **Y**, **Z** (en mm), **RX**, **RY**, **RZ** (en degré). Appuyez sur la touche **Jog**  (située dans le bandeau vertical de droite du *Teach Pendant*), par exemple, relativement à l'axe **Y** afin que le bras effectue une translation du TCP le long de l'axe $y_{Tool}$ du repère $R_{Tool}$ associé à l'outil (ou à la bride s'il n'y a pas d'outil). Lorsque la touche **Jog** est relative à **RX**, **RY** ou **RZ**, le TCP effectue une rotation autour des axes $x_{Tool}$, $y_{Tool}$ ou $z_{Tool}$.

## 1.4) Accès aux coordonnées du *Tool Center Point*

Le TCP correspond à l'origine du repère $R_{Tool}$ associé à l'outil. Notez que l'outil `flange` est utilisé par défaut (autrement dit, il n'y a pas d'outil rattaché à la bride), ce qui signifie que le repère outil est, par défaut, confondu avec le repère de la bride (*flange*).
Dans le menu **JOG** (accessible à travers le menu principal du *Teach Pendant*), il suffit de se rendre :

- dans l'**espace articulaire** (à l'aide du bouton **Joint**) pour accéder aux coordonnées angulaires du TCP (en degré) listées en face des boutons **J1**, ..., **J6**,
- dans l'**espace cartésien** (à l'aide du bouton **Frame**) pour accéder aux coordonnées cartésiennes du TCP dans le repère $R_0$ de base du bras du robot listées en face des boutons **X**, **Y**, **Z** (en mm), **RX**, **RY**, **RZ** (en degré),
- ou dans l'**espace outil** (à l'aide du bouton **Tool**), pour accéder aux coordonnées cartésiennes du TCP dans le repère $R_{Tool}$ associé à l'outil (dont l'origine correspond au TCP) listées en face des boutons **X**, **Y**, **Z** (en mm), **RX**, **RY**, **RZ** (en degré).

# 2) Programmation d'une application VAL3

Nous verrons au **2.1** comment créer une application VAL 3, intitulée `First_steps`, et visualiser le code (écrit en VAL3) de son programme `start()`.
Après une brève présentation des variables standards au **2.2**, une première application est réalisée au **2.3** afin que le bras du robot effectue un mouvement lui permettant d'être tendu à la verticale (cette posture étant définie à travers une *variable articulaire* intitulée `jDpt`) durant 2 secondes, puis le bras effectue un second mouvement afin que le TCP atteigne un point défini par une *variable cartésienne* intitulée `pExamplePoint` de valeurs égales à $X = 400, Y = 70, Z = 275\ (\text{mm}), RX = 25, RY = 100, RZ = -25\ (\text{degré})$.
La démarche permettant d'exécuter une application est décrite au **2.4** ; celle permettant de fermer une application, ce qui provoque son retrait de la mémoire vive du contrôleur, est décrite au **2.5**.

## 2.1) Création de l'application `First_steps` et édition de son programme `start()`

Une application est constituée de programmes, par défaut `start()` (appelé par le système lors du démarrage de l'application) et `stop()` (appelé par le système lors de l'arrêt de l'application). Dans ce qui suit, le code de l'application sera placée uniquement dans le programme `start()` (le programme `stop()`, initialement vide, ne sera pas modifié).

**Création de l'application `First_steps`**

A partir de la page d'accueil (accessible *via* la touche **Home**  , située en haut à gauche du *Teach Pendant*), la création d'une application, intitulée `First_steps`, nécessite de suivre les étapes suivantes :

- sélectionnez le menu **Val3** 

afin d'accéder à la fenêtre représentée dans la figure qui suit :

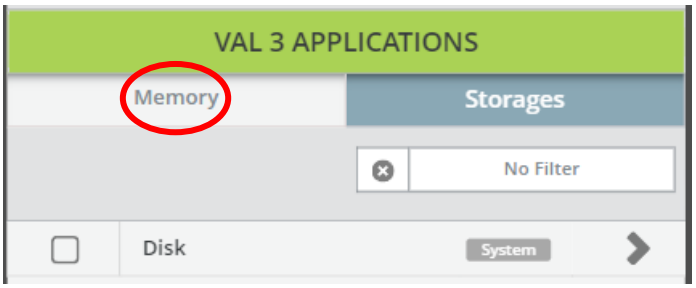



Figure 7 : Fenêtre des applications VAL3 avec la sélection, par défaut, de l'onglet `Storages`.

- sélectionnez l'onglet `Memory` (plutôt que `Storages`), entouré en rouge dans la figure précédente, afin d'accéder à la mémoire vive du contrôleur. Aucune application n'est présente dans la mémoire vive du contrôleur comme cela apparaît dans la figure qui suit :

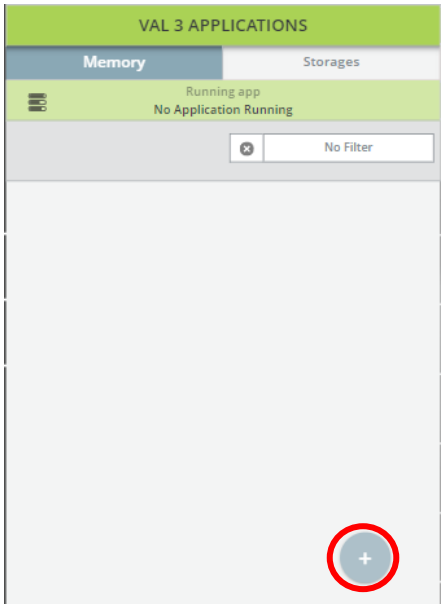



Figure 8 : Fenêtre des applications VAL3 quand l'onglet `Memory` est sélectionné.

- appuyez sur le bouton  , entouré en rouge dans la figure précédente, afin de permettre la création d'une application. Dans la fenêtre qui apparaît, voir la figure qui suit, tapez `First_steps` dans le champ **Name**, puis cliquez sur le bouton **OK** afin de valider.

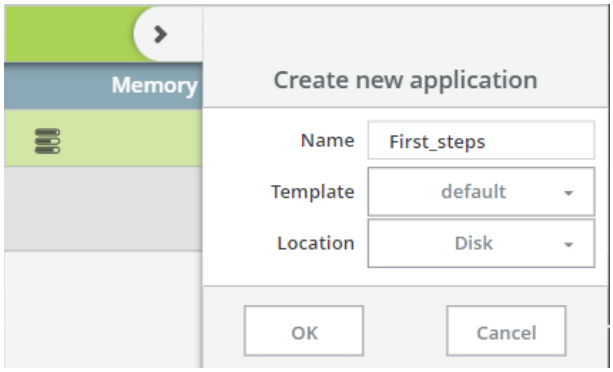



Figure 9 : Création de l'application `First_steps`.

Il en résulte la création de l'application `First_steps` dans la mémoire vive du contrôleur comme cela est montré dans la figure suivante où l'application `First_steps` apparaît dans l'onglet `Memory` de la fenêtre correspondant aux applications VAL3 :

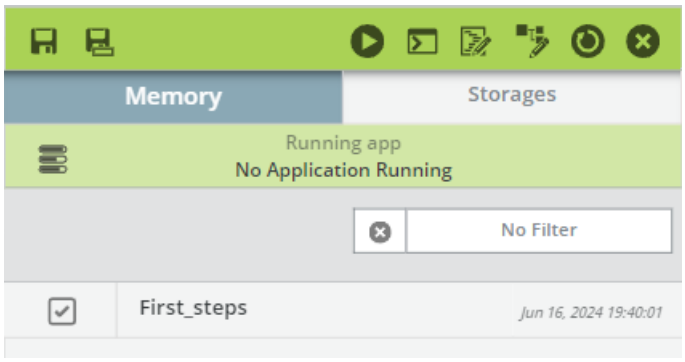



Figure 10 : Affichage de l'application `First_steps` dans l'onglet `Memory` de la fenêtre des applications VAL3.

Cette application est également sauvegardée dans le disque dur du contrôleur, comme cela peut être vérifié dans la fenêtre correspondant aux applications VAL3 avec la sélection de l'onglet `Storages`.

Par la suite, **veillez à ne travailler** que sur l'application `First_steps` afin de ne pas perturber le contenu du disque dur du contrôleur.

**Edition du programme `start()`**

Pour éditer le code du programme `start()` de l'application :

- appuyez sur le bouton  , situé en haut dans la barre du menu représentée dans la figure précédente. Les programmes de l'application, à savoir, `start()` et `stop()`, sont listés dans l'onglet **Programs** (sélectionné par défaut) de la fenêtre qui apparaît, voir la figure qui suit.

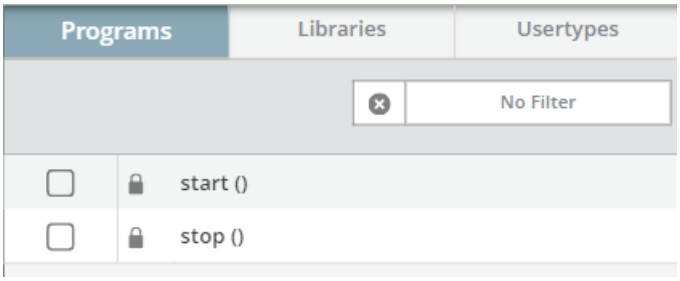



Figure 11 : Liste des programmes `start()` et `stop()` composant l'application `First_steps`.

- sélectionnez le programme `start()` pour faire apparaître son contenu, pour l'instant vide excepté les balises `begin` et `end` délimitant le code du programme :

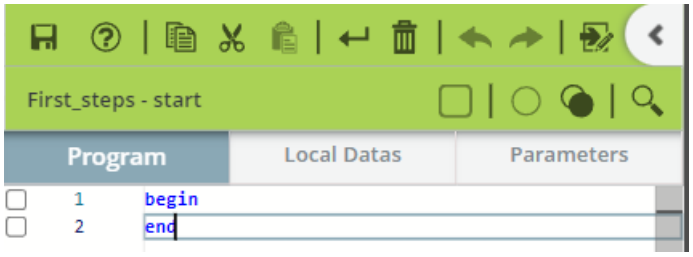



Figure 12 : Code du programme `start()`.

## 2.2) Les variables

Les principaux types de variables VAL3, dont ceux spécifiques à la robotique, sont rapidement décrits au **2.2.1**. La démarche permettant de visualiser les variables d'une application et de déclarer une nouvelle variable (avec des valeurs données par défaut) est décrite au **2.2.2**, sachant que la méthode permettant d'initialiser les valeurs d'une variable est décrite au **2.2.3**.

### 2.2.1) Types de variables

Plusieurs types de variables sont disponibles dans VAL3. Il y a celles classiques à un langage de programmation comme les variables booléennes (`bool`), numériques (`num`), de chaînes de caractères (`string`). Certaines variables sont spécifiques à la robotique, comme : les variables *points* définis dans l'*espace articulaire* (appelés par la suite, *points articulaires* (`jointRx`)) ou définis dans l'*espace cartésien* (appelés par la suite, *points cartésiens* (`pointRx`)) ; les variables *outils* (`tool`) ; les variables définissant des *repères cartésiens* (`frame`) ; les variables définissant des *changements de position* et/ou d'*orientation* (`trsf`).
Pour faciliter la reconnaissance du type d'une variable, il est supposé que les premières lettres de son nom indiquent son type, à savoir, concernant les types décrits précédemment :

`bVariable` pour une variable de type `bool`,
`nVariable` pour une variable de type `num`,
`sVariable` pour une variable de type `string`,
`jVariable` pour une variable de type `jointRx`,
`pVariable` pour une variable de type `pointRx`,
`tVariable` pour une variable de type `tool`,
`fVariable` pour une variable de type `frame`,
`trVariable` pour une variable de type `trsf`.

### 2.2.2) Affichage des variables de l'application `First_steps`, déclaration d'une nouvelle variable

**Hypothèse :** L'application `First_steps` est chargée dans la mémoire vive du contrôleur (voir la figure 10), ce qui donne accès au menu décrit dans la figure qui suit.

**Affichage des variables de l'application `First_steps`**

Appuyez sur le bouton  , entouré en rouge dans la barre du menu représenté dans la figure qui suit, pour visualiser les variables de l'application.

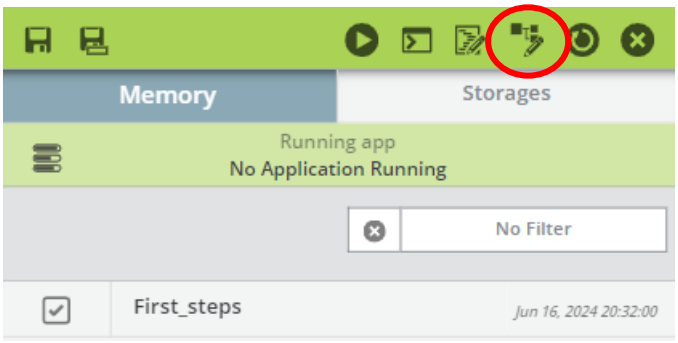



Figure 13 : Bouton permettant l'accès aux variables de l'application `First_steps`.

Il en résulte la fenêtre, représentée dans la figure qui suit, listant les variables de tous types par ordre alphabétique lorsque l'onglet **Data** est sélectionné (ce qui est le cas par défaut). Les variables sont listées de manière hiérarchisée lorsque l'onglet **Geometry** est sélectionné.
Par défaut, la variable `mNomSpeed`, de type `mdesc`, est définie afin d'indiquer la vitesse du TCP lors des déplacements du bras du robot.

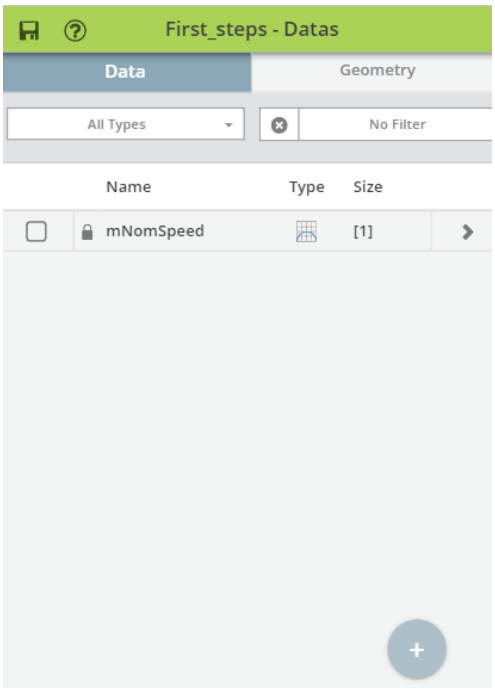



Figure 14 : Affichage des variables (*via* l'onglet **Data**) de l'application `First_steps`.

Notez que la présence d'une icône en forme de cadenas en face d'une variable indique qu'elle est privée (ce qui est le cas de la variable `mNomSpeed`), elle est publique dans le cas contraire.

**Déclaration d'une nouvelle variable**

A titre d'exemples, définissons deux variables : une variable articulaire intitulée `jDpt` (de type `jointRX`), puis une variable cartésienne intitulée `pExamplePoint` (de type `pointRx`).

Le bouton  , situé en bas à droite dans la figure précédente, permet la création de nouvelles variables dont le type, le nom, le conteneur, etc. sont à déclarer dans la fenêtre qui apparaît suite à l'appui sur le bouton  .

✓ **Création de la variable articulaire `jDpt`**

Suite à un appui sur le bouton  , la variable `jDpt` est créée à travers les contenus des champs décrits dans la figure qui suit :

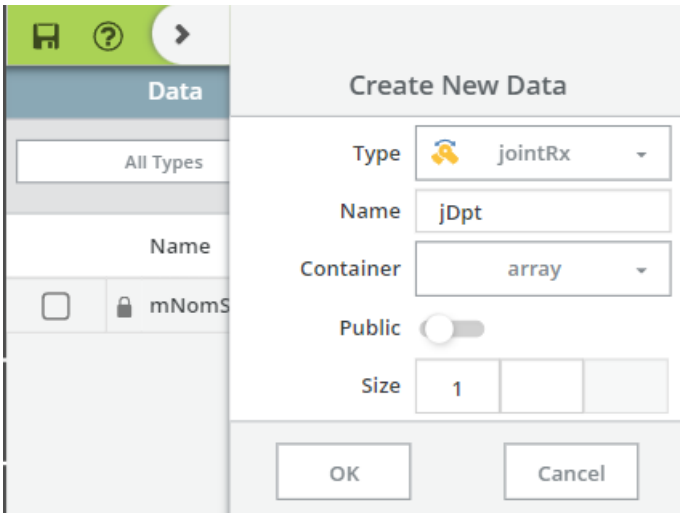



Figure 15 : Création de la variable `jDpt`.

où il est indiqué que la variable `jDpt` (champ `Name`) est de type `jointRX` (champ `Type`), qu'elle correspond à un tableau (`array`) (champ `Container`) de taille unitaire (champ `Size`) (ce qui explique la dénomination `jDpt[0]`) et que sa portée est privée (bouton `Public` sur off). Pensez à valider vos données en cliquant sur le bouton **OK**. Notez que la variable est initialisée avec des valeurs par défaut, l'accès à ces valeurs étant décrit à l'annexe **A.2**.

L'initialisation de cette variable est faite au **2.2.3** dans le programme `start()` ; sachez qu'il est également possible de lire, ou d'initialiser, une variable articulaire ou cartésienne en utilisant le menu VAL3, voir pour cela l'annexe **A.2**.

✓ **Création de la variable cartésienne `pExamplePoint`**

Suite à un appui sur le bouton  , la variable `pExamplePoint` est créée à travers les contenus des champs décrits dans la figure qui suit :

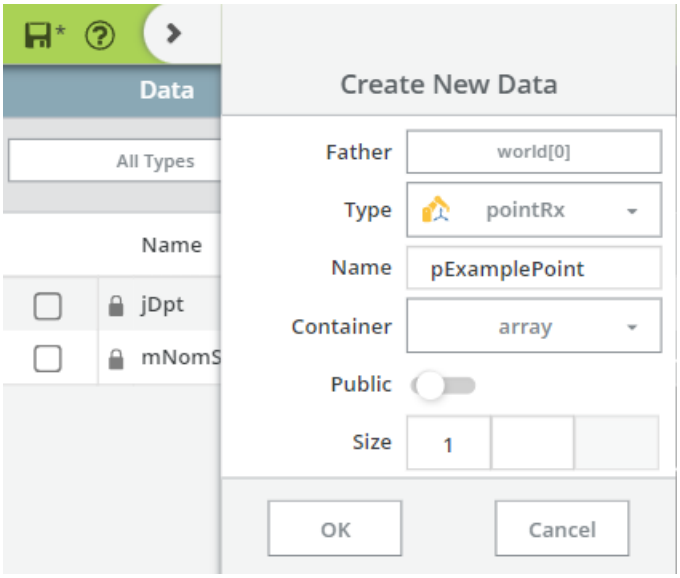



Figure 16 : Création de la variable `pExamplePoint`.

où il est indiqué que la variable `pExamplePoint` (champ `Name`) est de type `pointRX` (champ `Type`), qu'elle correspond à un tableau (`array`) (champ `Container`) de taille unitaire (champ `Size`) et que sa portée est privée (bouton `Public` sur off). La variable est définie par rapport au

repère `world` (champ `Father`), lequel correspond au repère $R_0$ de base du robot. Sachez qu'il est possible de définir des points par rapport à un repère (préalablement défini) autre que `world`. Pensez à valider vos données en appuyant sur le bouton **OK**. Notez que la variable est initialisée avec des valeurs par défaut, l'accès à ces valeurs étant décrit à l'annexe **A.2**.

**N.B :** Les variables ne sont pas enregistrées si un astérisque apparait dans l'icône représentant une disquette (  ), située en haut à gauche dans la figure précédente. Cliquez sur cette icône pour effectuer l'enregistrement, l'icône ne comporte alors plus d'astérisque.

### 2.2.3) Initialisation d'une variable

Il est possible d'initialiser dans le programme `start()` une variable comme, par exemple, `jDpt` (de type `jointRX`) ou `pExamplePoint` (de type `pointRX`). Ceci se fait simplement à travers les instructions suivantes (bien sûr, à placer avant leur utilisation dans une instruction) :

```
jDpt={0,0,0,0,0,0}
pExamplePoint={{400,70,275,25,100,-25},{ssame,esame,wsame}}
```

où :

- les valeurs `{400,70,275,25,100,-25}` correspondent aux coordonnées `x,y,z,Rx,Ry,Rz` indiquant la *situation* (*i.e.*, la *position* et l'*orientation*) du *point* `pExamplePoint`,
- les paramètres `{ssame,esame,wsame}` (de type configuration) interdisent un changement de configuration du bras au cours du mouvement vers le point `pExamplePoint`. Dans le programme décrit au **2.3**, la configuration du bras est celle prise pour atteindre le point articulaire `jDpt`, à savoir, épaule à gauche, coude et poignet positifs).**Remarque :** Deux autres méthodes existent afin d'initialiser une variable. Contrairement à la méthode précédente, qui initialise une variable directement dans le code du programme, ces méthodes nécessitent de faire des manipulations sur le *Teach Pendant*.

La méthode, décrite dans l'annexe **A.2**, est applicable *via* le menu **VAL3** et permet d'initialiser une variable de tous types, notamment des variables : *point* de type `jointRx` ou `pointRx` ; *repère* de type `frame` ; *outil* de type `tool`.
La méthode, décrite dans l'annexe **A.3**, est applicable *via* le menu **JOG** (utilisé habituellement pour la mise en mouvement du bras du robot de manière manuelle vers un point donné, voir **3**) et permet d'initialiser des variables *point*, *repère* et *outil*.

## 2.3) Codage du programme `start()`

**Hypothèse :** Les variables articulaire `jDpt` et cartésienne `pExamplePoint` sont créées, voir la démarche décrite au **2.2.2**.

La première étape consiste à éditer le code du programme `start()`, voir pour cela la démarche décrite au **2.1**. Complétez le programme `start()`, initialement constitué des balises `begin` et `end`, comme suit :

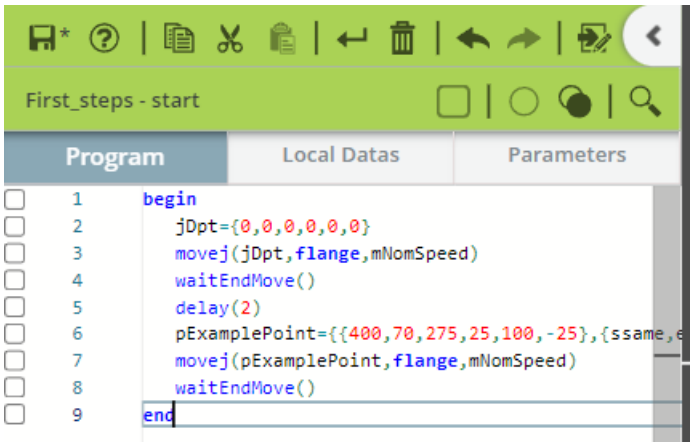



Figure 17 : Code contenu dans le programme `start()`.

La variable articulaire `jDpt`, de type `JointRx` (voir **2.2.1** pour plus de détails), est telle que $J1 = \cdots = J6 = 0$ (voir **2.2.3** pour l'initialisation de la variable).
L'instruction `movej(jDpt,flange,mNomSpeed)` permet de faire un mouvement du *point courant* (celui atteint juste avant l'exécution de l'instruction) vers le *point* `jDpt` (où le bras du robot est tendu à la verticale). Le paramètre `flange` indique qu'il n'y a pas d'outil fixé au bras du robot, autrement dit, le repère associé au TCP correspond à celui associé à la bride (« *flange* » en anglais) du robot. L'utilisation de `movej` fait que la trajectoire réalisée optimise la vitesse du mouvement (dit de *point-à-point*) ; notez qu'il existe d'autres instructions de mouvement permettant d'effectuer une trajectoire rectiligne ou circulaire, mais sans permettre de garantir une vitesse optimale du mouvement.
L'instruction `waitEndMove()`, située après `movej(jDpt,flange,mNomSpeed)` permet de finaliser le mouvement vers le *point* `jDpt` avant de poursuivre l'exécution du programme, ainsi il n'y a pas de phénomène de lissage du *point* `jDpt` avec le *point* suivant (`pExamplePoint`).
L'instruction `Delay(2)` provoque une attente de 2 secondes au point courant, à savoir, `jDpt`.
Voir **2.2.3** concernant l'initialisation de la variable cartésienne `pExamplePoint`.
L'instruction `movej(pExamplePoint,flange,mNomSpeed)` réalise un mouvement vers le *point* `pExamplePoint` avec un mode de fonctionnement similaire au mouvement vers le *point* `jDpt`.
L'instruction `waitEndMove()`, située juste avant l'instruction `End`, permet de finaliser le mouvement vers le *point* `pExamplePoint` avant que le programme s'arrête. Aussi, veillez à toujours placer cette instruction juste avant l'instruction `End` de votre programme.

Ce programme est tel qu'une fois le bras tendu à la verticale, 2 secondes s'écoulent, puis le TCP rejoint le *point* `pExamplePoint` de coordonnées $X = 400, Y = 70, Z = 275, RX = 25, RY = 100, RZ = -25$.

**N.B :** Un programme n'est pas enregistré si un astérisque apparait dans l'icône représentant une disquette ( ), située en haut à gauche dans la figure précédente. Cliquez sur cette icône pour effectuer l'enregistrement, l'icône ne comporte alors plus d'astérisque.

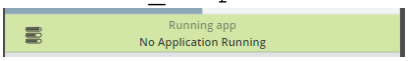
## 2.4) Exécution de l'application

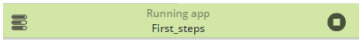
**Hypothèses :**
- Le mode **Off** est celui sélectionné dans le menu **JOG** (et non **Joint**, **Frame** ou **Tool**) (voir **1.3**),
- Le bras est sous puissance avec un Mode de Marche **manuel** (voir **1.2**),
- L'application `First_steps` est chargée dans la mémoire vive du contrôleur (voir la figure 10).

**N.B. :** La touche **Move/Hold**  , située en haut à droite du *Teach Pendant*, permet un arrêt soft et immédiat du mouvement du bras du robot (ce qui peut être utile lors de la mise au point d'un programme), ce type d'arrêt n'enclenche pas une coupure de la puissance du bras.

L'application `First_steps` étant chargé dans la mémoire vive du contrôleur, allez dans la page décrite à la figure 10 (après 2 appuis successifs sur la touche **Back**  si vous venez du menu décrit dans la figure précédente). La case correspondant à l'application étant cochée (ce qui place l'application « en haut » de la pile d'exécution si plusieurs applications sont chargées en mémoire vive (ce qui n'est pas le cas ici)), le lancement de l'application se fait en appuyant sur le bouton **Run**  (situé dans la barre du menu), puis en appuyant : sur la touche **Move/Hold**  du *Teach Pendant* une première fois, puis une seconde fois de manière constante, ainsi que sur le corps mort (le mouvement du bras s'arrêtant dès que la touche Move/Hold ou le corps mort est relâché)**.**

## 2.5) Fermeture de l'application

Assurez-vous que l'application `First_steps` est bien arrêtée comme indiqué par le message **No Application Running** 

affiché sur le *Teach Pendant* ; si ce n'est pas le cas, l'application peut être arrêtée en appuyant sur le bouton **stop** 

, situé en haut à droite du *Teach Pendant*. La case correspondant à l'application `First_steps` étant cochée (voir la figure 10), sa fermeture se fait en appuyant sur le bouton **Close**  , situé en haut à droite du *Teach Pendant*, ce qui provoque son retrait de la mémoire vive (Memory).

# 3) Mouvement de manière manuelle vers un point donné

Durant la mise au point d'une trajectoire, il peut être intéressant de bouger manuellement le bras du robot pour tester l'accès à certains points. De tels mouvements se font à partir du menu **JOG**, accessible *via* le menu principal du *Teach Pendant* (appuyez sur la touche **Home**  pour s'y rendre).

**Hypothèses :**
- Le bras est sous puissance avec un Mode de Marche **manuel** (voir **1.2**),
- L'application `First_steps` est chargée dans la mémoire vive du contrôleur (voir la figure 10).

A partir du menu **JOG** décrit dans la figure suivante :

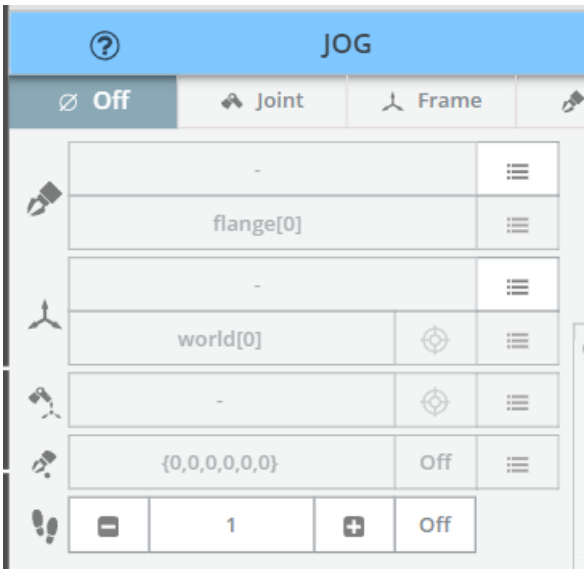



Figure 18 : Menu **JOG**.

- Appuyez sur le bouton entouré en rouge dans la figure qui suit pour accéder à l'application `First_steps` (située en bas dans la figure qui suit) :

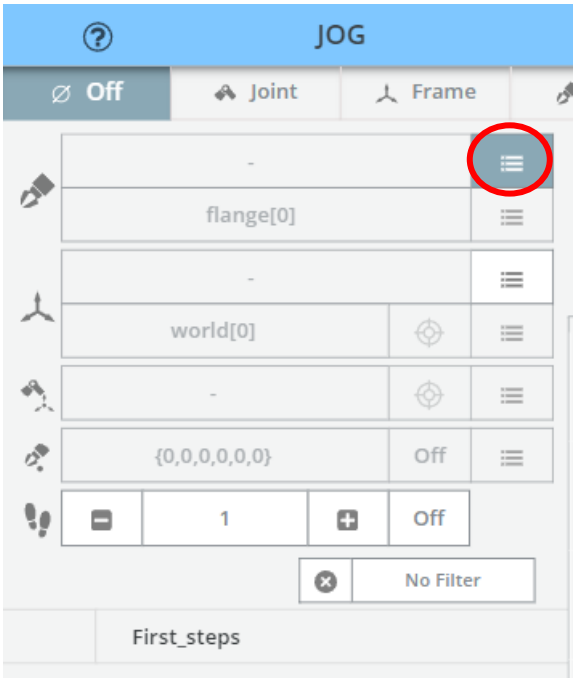



Figure 19 : Accès à l'application `First_steps` dans le menu **JOG**.

- Sélectionnez l'application `First_steps` pour faire apparaître la fenêtre décrite dans la figure qui suit où l'outil et le repère de base utilisés dans l'application sont ceux proposés par défaut, à savoir : `flange` (correspondant au cas où il n'y a pas d'outil attaché au bras du robot) et `world` (ce qui signifie que les *points* considérés par la suite sont définis dans le repère $R_0$ de base du robot (et non un autre repère qui aurait été défini préalablement)).

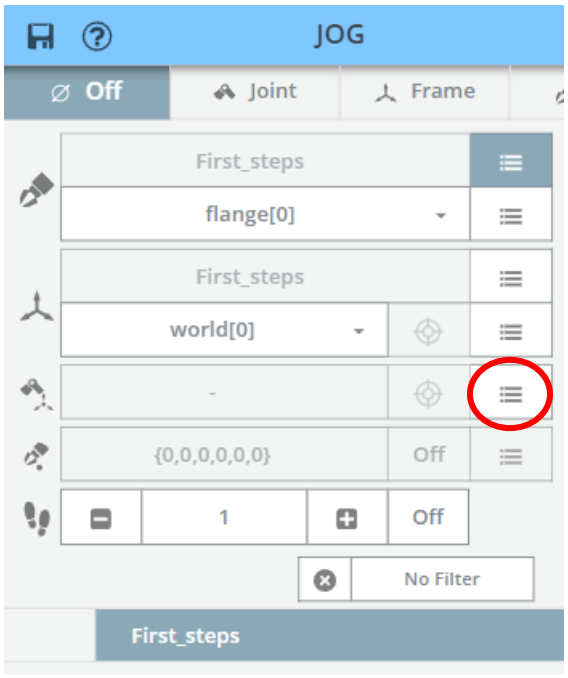



Figure 20 : Sélection de l'application `First_steps` dans le menu **JOG**.

- Appuyez sur le bouton, entouré en rouge dans la figure précédente, pour visualiser les *variables articulaires* (du fait de la sélection, par défaut, de **Joint** dans le champ entouré en rouge dans la figure qui suit) associées à l'application : la variable `jDpt`, dans le cas présent, comme indiqué dans la figure qui suit :

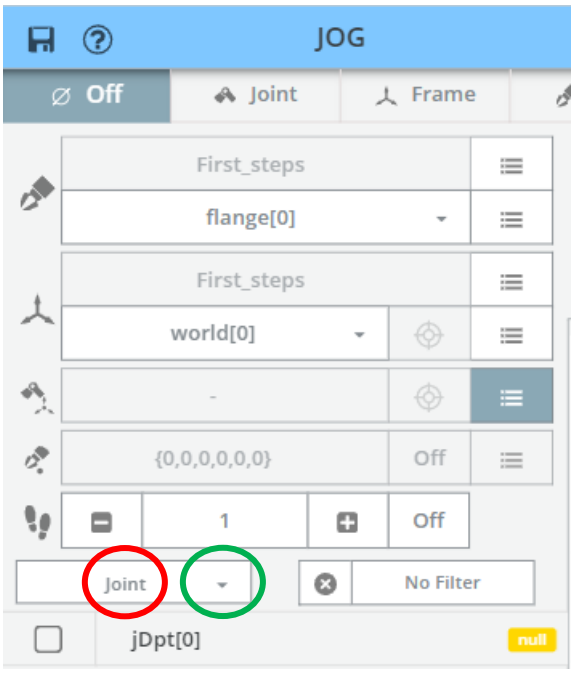



Figure 21 : Affichage des variables de type **joint** (dans le cas présent, `jDpt`) dans le menu **JOG**.

## 3.1) Mouvement manuel vers le point articulaire `jDpt`

Dans la fenêtre décrite dans la figure précédente, cochez la case correspondant à la variable `jDpt` pour indiquer que vous souhaitez que le TCP se dirige vers le *point* `jDpt`. Appuyez sur le bouton **Joint**  , qui apparaît en haut à droite dans la fenêtre décrite dans la figure

précédente, pour indiquer que le mouvement va être calculé dans l'espace articulaire, ce qui fait apparaître la fenêtre décrite dans la figure qui suit :

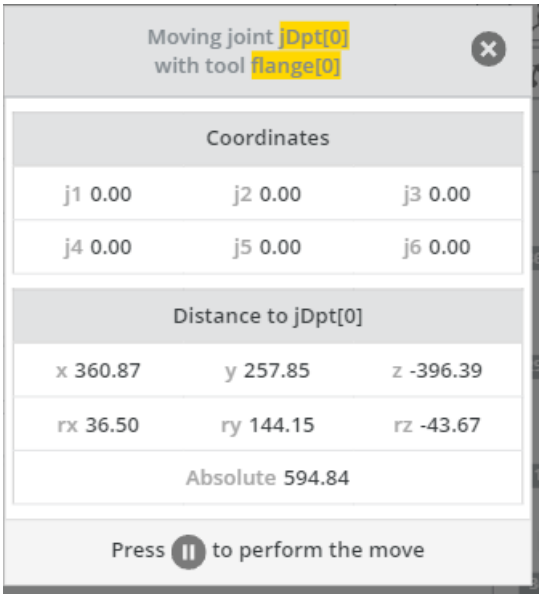



Figure 22 : Fenêtre affichant les données de la variable `jDpt` avant le mouvement du bras du robot.

où est indiquée la distance entre le point courant et le point `jDpt` et mentionne que la mise en mouvement du bras se fait en appuyant sur la touche **Move/Hold**  .

## 3.2) Mouvement manuel vers le point cartésien `pExamplePoint`

Dans la fenêtre décrite dans la figure 21, sélectionnez **Point** (au lieu de **Joint** sélectionné par défaut) dans le menu déroulant entouré en vert dans la figure 21 pour disposer des *variables cartésiennes* associées à l'application : la variable `pExamplePoint`, dans le cas présent, comme indiqué dans la figure qui suit :

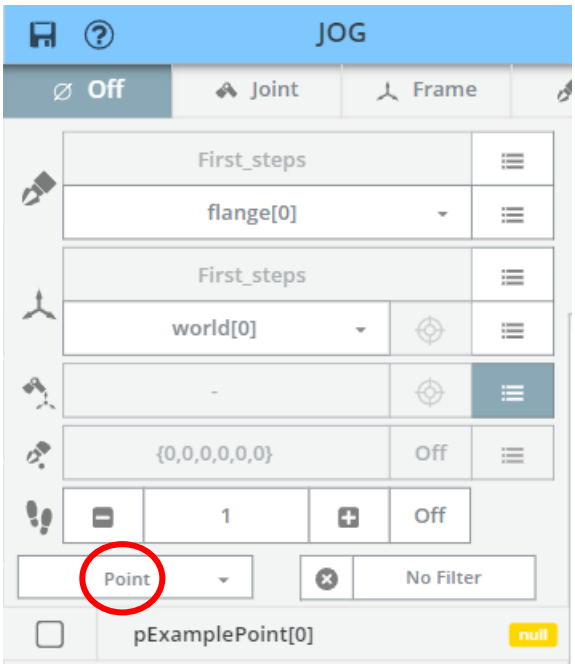



Figure 23 : Affichage des variables de type **point** (dans le cas présent, `pExamplePoint`) dans le menu **JOG**.

Cochez la case correspondant à la variable `pExamplePoint` pour indiquer que vous souhaitez que le TCP se dirige vers le point `pExamplePoint`. Deux façons de mettre en mouvement le bras du robot sont proposées :

- soit en appuyant sur le bouton **Joint**  , qui apparaît en haut à droite dans la fenêtre décrite dans la figure précédente, pour indiquer que le calcul de ce mouvement se fait dans l'espace articulaire, ce qui fait apparaître la fenêtre décrite dans la figure qui suit :

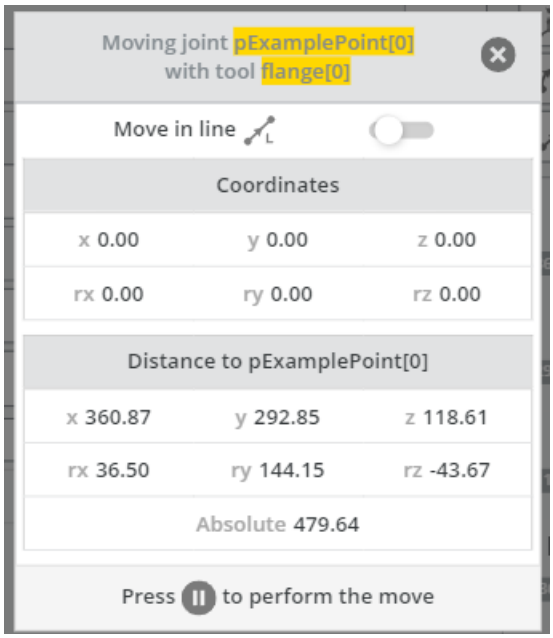



Figure 24 : Fenêtre affichant les données de la variable `pExamplePoint` avant le mouvement du bras du robot.

laquelle indique la distance entre le *point courant* et le point `pExamplePoint` et mentionne que la mise en mouvement du bras se fait en appuyant sur la touche **Move/Hold**  .

Vous noterez que le bouton correspondant au champ  'Move in line' est sur **off** ce qui fait que le mode actif est bien celui articulaire (et non linéaire).

- soit en appuyant sur le bouton **Line**  , qui apparaît en haut à droite dans la fenêtre décrite dans la figure 23, afin que le TCP rejoigne le point `pExamplePoint` **en ligne droite**. Notez que le calcul du mouvement se fait dans l'espace cartésien, aussi le mouvement n'est pas toujours possible !

# ANNEXE

## A.1) Chargement en mémoire vive d'une application sauvegardée dans le contrôleur

La démarche qui suit permet de charger en mémoire vive une application située sur le disque dur du contrôleur, par exemple, en vue de l'exécuter.

A partir de la page d'accueil (accessible *via* la touche **Home**  , située en haut à gauche du *Teach Pendant*) :

- accédez à la liste des applications situées dans le disque dur du contrôleur en sélectionnant l'onglet **Storages** du menu **VAL3**  , puis en appuyant sur le bouton  situé sur la ligne correspondant à **Disk**, entouré en rouge dans la figure qui suit ;

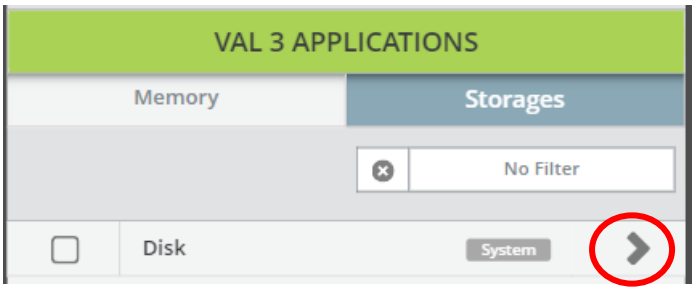



Figure 25 : Sélection du disque dur du contrôleur.

- dans la fenêtre qui apparaît, cochez la case correspondant à l'application que vous souhaitez charger en mémoire vive, ce qui provoque son apparition dans l'onglet **Memory**, à l'image de la fenêtre décrite dans la figure 10.

## A.2) Lecture, initialisation d'une variable en utilisant le menu VAL3

Comme cela a été vu au **2.2.2**, le menu **VAL3** permet d'afficher l'ensemble des variables (de tous *types*) déclarées dans une application. Nous allons voir que le menu **VAL3** peut également être utilisé pour initialiser le contenu d'une variable, par exemple, la variable articulaire `jDpt` utilisée dans l'application `First_steps`.

**Hypothèse :** L'application `First_steps` est chargée dans la mémoire vive du contrôleur (voir la figure 10).

Allez à la page permettant de visualiser les variables de l'application `First_steps` (pour cela voir la démarche décrite au **2.2.2**), ce qui donne lieu à la page décrite dans la figure qui suit :

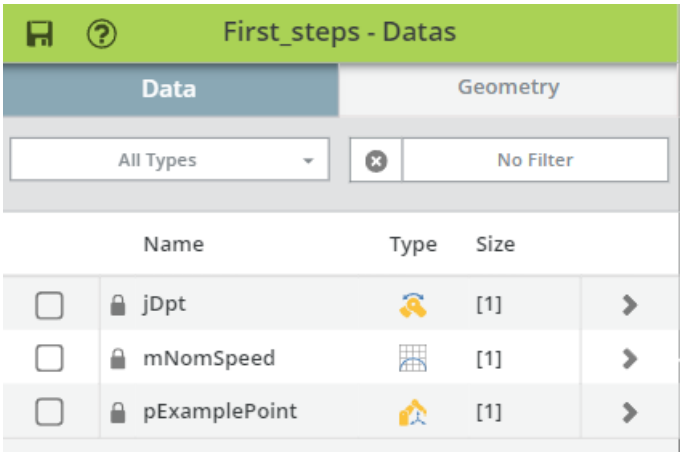



Figure 26 : Affichage des variables *via* l'onglet **Data**.

Le fait d'appuyer sur le bouton > correspondant à la variable `jDpt`, puis de cocher la case lui correspondant dans la fenêtre qui apparaît, permet de lire le contenu de la variable, mais également d'initialiser ses valeurs en appuyant sur le bouton **Edit** (situé en haut dans la barre du menu), voir la figure qui suit :

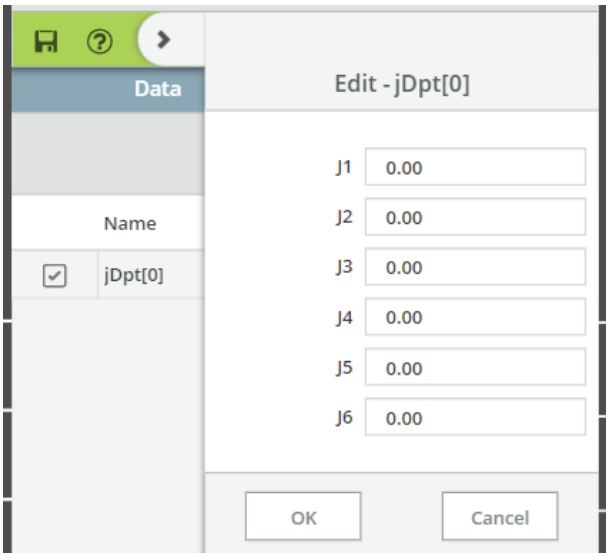


Figure 27 : Lecture, initialisation du contenu de la variable `jDpt`.

Pensez bien sûr à valider avec le bouton **OK** en cas de modification du contenu de la variable !

## A.3) Lecture, initialisation d'une variable en utilisant le menu JOG

Le menu **JOG** est habituellement utilisé pour mettre en mouvement le bras du robot de manière manuelle vers un point donné (voir **3**), nous allons voir qu'il est également possible d'utiliser ce menu pour créer une variable de type : point (`jointRX`, `pointRX`), repère (`frame`) ou outil (`tool`), ainsi que de lire ou initialiser son contenu.

A titre d'exemple, lisons le contenu de la variable articulaire `jDpt` utilisée dans l'application `First_steps`. Puis, créons une nouvelle variable cartésienne intitulée `pOtherPoint`.

**Hypothèse :** L'application à éditer, dans le cas présent `First_steps`, est en mémoire vive (voir la figure 10).

✓ **Lecture, initialisation de la variable articulaire `jDpt` (de type `jointRx`)**

Allez dans le menu **JOG** afin de visualiser les variables articulaires de l'application `First_steps`, pour cela voir la démarche décrite au **3** et **3.1**, ce qui donne lieu à la page décrite dans la figure qui suit (identique à la figure 21) :

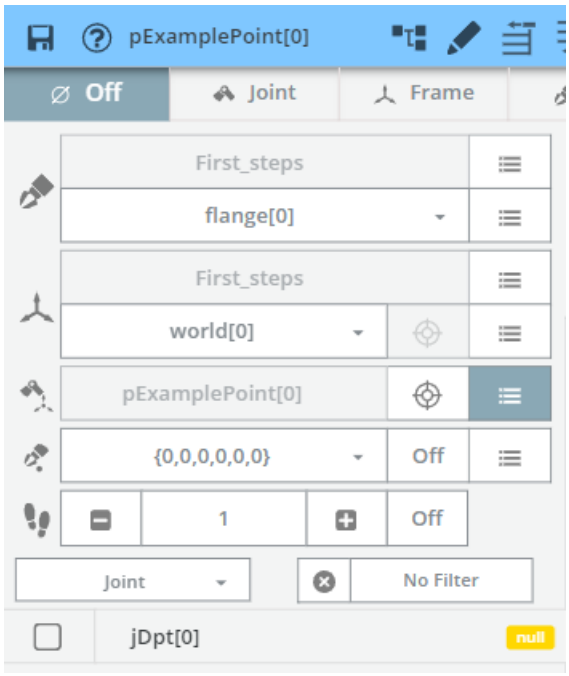



Figure 28 : Affichage des variables de type **joint**, ici `jDpt`, dans le menu **JOG**.

- Cochez la case correspondant à la variable `jDpt`, puis appuyez sur le bouton **Edit**  , situé en haut dans la barre du menu, afin d'accéder à la fenêtre décrite dans la figure qui suit laquelle permet la lecture ou l'initialisation du contenu de la variable `jDpt`.


Figure 29 : Lecture, initialisation du contenu de la variable `jDpt`.

✓ **Création d'une variable cartésienne `pOtherPoint` (de type `pointRx`)**

Allez dans le menu **JOG** afin de visualiser les variables cartésiennes de l'application `First_steps`, pour cela voir la démarche décrite au **3** et **3.2**, ce qui donne lieu à la page décrite dans la figure qui suit (identique à la figure 23) :

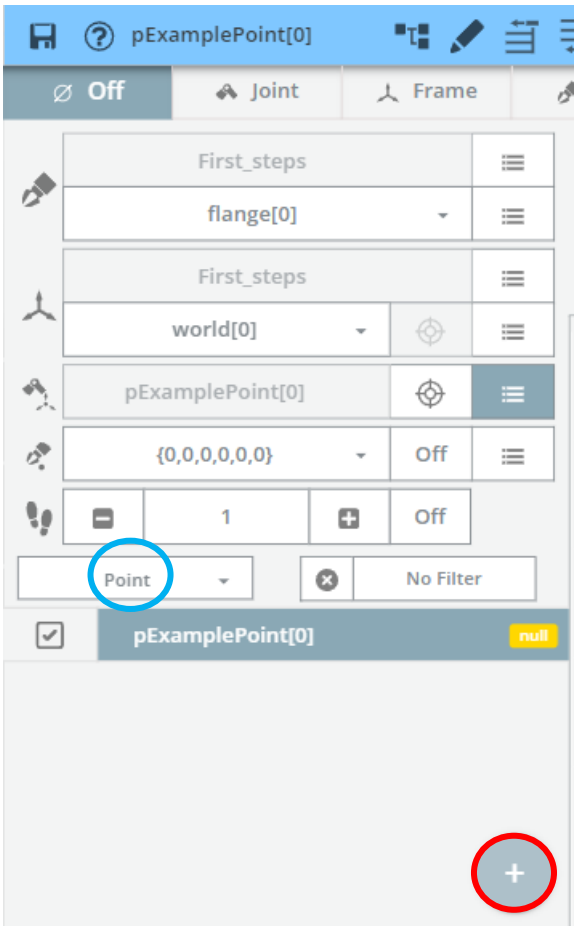



Figure 30 : Affichage des variables cartésiennes (de type `pointRX`), ici `pExamplePoint`, dans le menu **JOG**.

✓ Notez bien que le point qui va être créé est défini par rapport au repère `world` (*i.e.*, le repère $R_0$ de base du robot), comme indiqué dans la figure qui suit.

- Appuyez alors le bouton  entouré en rouge (en bas à droite) dans la figure précédente.

Sélectionnez dans la fenêtre qui apparaît le contenu des champs :

- **Name** afin de déclarer une variable, intitulée `pOtherPoint`, cartésienne (**Type** `pointRx`),
- **Container** afin d'indiquer que la variable correspond à un tableau (`array`) de dimension 1 (champ **Size**),

indiquez que sa portée est privée (champ **Public** sur off, sélectionné par défaut), comme indiqué dans la figure qui suit.

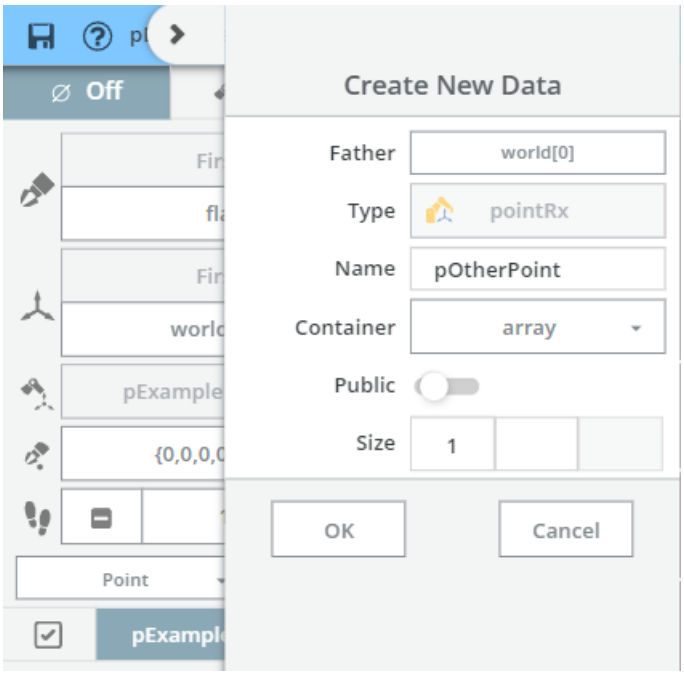



Figure 31 : Création de la variable `pOtherPoint`.

ce qui donne lieu (après validation *via* le bouton **OK**) au résultat décrit dans la figure qui suit où apparaît la variable `pOtherPoint`.

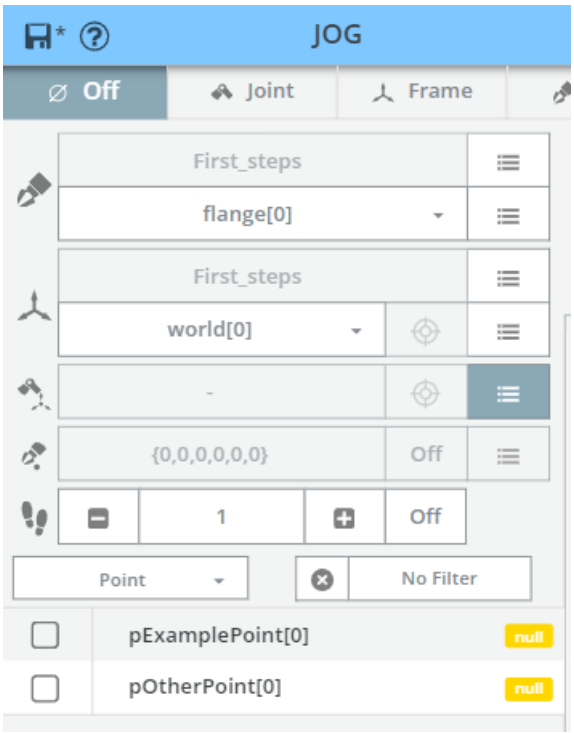



Figure 32 : Affichage des variables Cartésiennes (de type `pointRX`), ici `pExamplePoint` et `pOtherPoint`, dans le menu **JOG**.

- Cochez la case correspondant à la variable `pOtherPoint`, puis appuyez sur le bouton **Edit**  , situé en haut dans la barre du menu, afin d'accéder à la fenêtre décrite dans la figure qui suit laquelle permet la lecture ou l'initialisation du contenu de la variable `pOtherPoint`, à travers les valeurs des champs $X, Y, Z, RX, RY, RZ$.

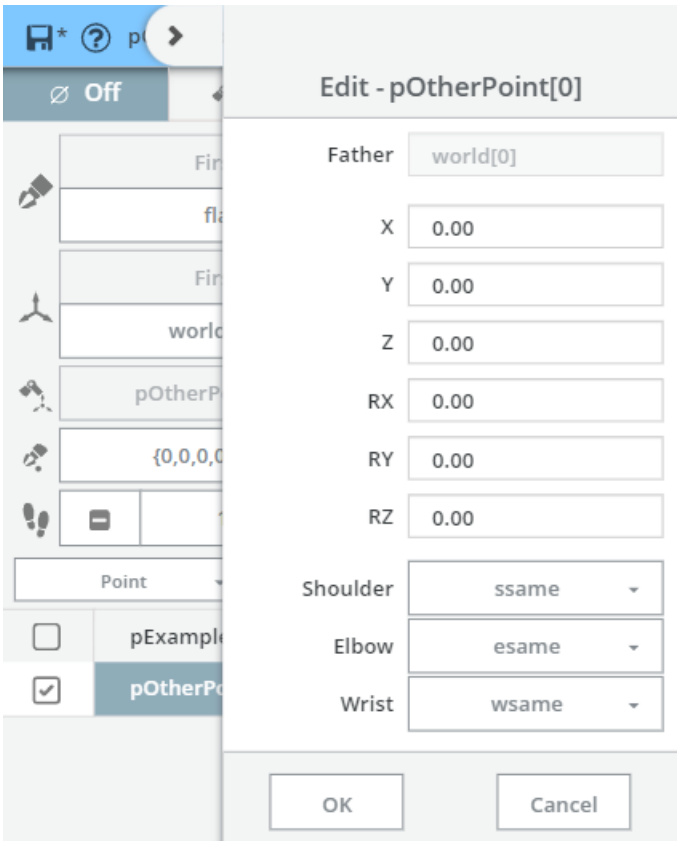



Figure 33 : Lecture, écriture/initialisation du contenu de la variable `pOtherPoint`.

**N.B. :** Il est possible « d'apprendre » un point avec le *Teach Pendant*, c'est-à-dire, dans notre exemple, d'initialiser le contenu de la variable `pOtherPoint` avec les coordonnées cartésiennes du *point courant* du TCP (correspondant au point atteint par le TCP suite au dernier mouvement effectué sur le bras du robot). Pour cela, cochez la case correspondant à la variable `pOtherPoint` (comme précédemment), puis appuyez sur la mire, relative à cette variable, entourée en rouge dans la figure qui suit.

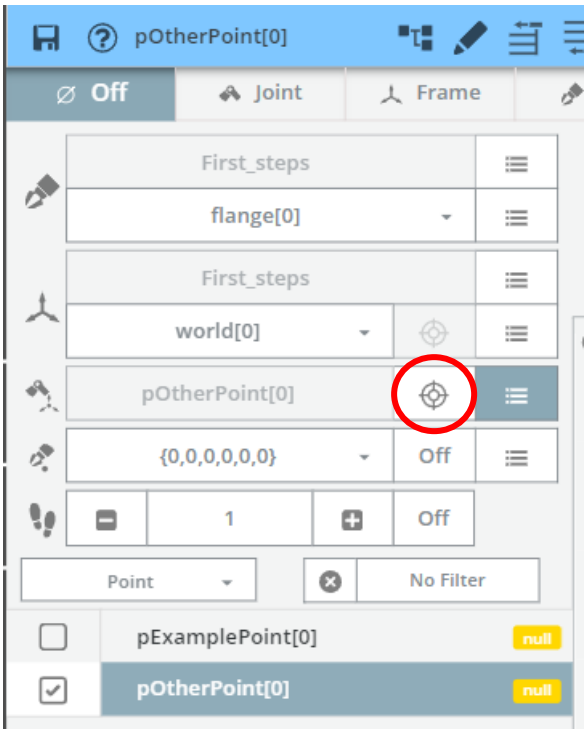



Figure 34 : Initialisation du contenu de la variable `pOtherPoint` avec les coordonnées cartésiennes du *point courant*.

Il s'ensuit l'apparition d'une fenêtre permettant de valider l'assignation de la variable `pOtherPoint` aux valeurs correspondant au *point courant*.