Baie robot ABB S4Cplus : Guide d'intégration de l'application de base manutention

n° :	**GIPROMIA4.0-D**
	Guide d'intégration
Statut:	**Exécutoire**

Objet	Aide à la mise en œuvre des fonctions de l'application de base pour une baie ABB S4Cplus Automotive
Champ d'application	Groupe utilisant la manutention comme métier de base.
Emetteur	ABB ATRM - Département R&D - Support Produit
Confidentialité	Non confidentiel.

Approuvé par	*Fonction*	*Signature*	*Date d'application*
			Décembre 2004

Historique des versions

Version	*Mise à jour*	*Objet des principales modifications*	*Rédacteur*
A	Oct 2002	Création sur base du Document Réf : GI310149-B	M. BEKAERT
B	Fév 2003	Prise en compte des évolutions de la version 40.30	M. BEKAERT
C	Nov 2003	Diverses corrections	M. BEKAERT
D	Jnv2005	Prise en compte des évolutions de la version 40.10 Ajout des chronogrammes	S. HAMIDI B. COPITET M. BEKAERT

Mise à disposition

Autre Document

– NUPROMIA4.0 : Baie robot ABB S4Cplus Automotive : Notice Utilisateur de l'application de base PROMIA

Codification

Mots-clés

Sommaire

1. OBJET DU DOCUMENT.

Ce document est destiné avant tout aux intégrateurs afin de leur permettre de programmer les baies robot ABB S4Cplus équipées du logiciel d'application "PROMIA 4.0". Il est également destiné à toute personne, et notamment le personnel du client final, désirant avoir une connaissance de cet applicatif qui ne se limite pas à sa simple utilisation.

Il aborde les points suivants :

- description des fonctions apportées par le logiciel d'application
- configurations à effectuer
- écriture des programmes et trajectoires
- description des différentes actions et de leurs paramètres
- annexes (procédures de sauvegardes / restitutions, configuration des robots, limites de prestations, documentation à remettre...)
- règles de bonne programmation à respecter

L'application standard PROMIA version 4.0 doit être configurée et personnalisée par l'intégrateur. L'ensemble de ces opérations doit être réalisé conformément à la présente notice et sont sous l'entière responsabilité de l'intégrateur.

Les opérations sont décrites chronologiquement. Le chargement de l'application, décrit au chapitre 3 n'est pas nécessaire lorsque l'application a été pré-chargée par ABB.

Il est conseillé d'avoir préalablement lu la notice utilisateur et/ou les normes Renault disponibles sur le site *http://www.cnomo.com* afin de se familiariser avec les notions abordées dans ce guide, telles que séquences, contrôles dynamiques, points de repli, de rebouclage...

Ce document a pour complément la notice utilisateur PROMIA référence NUPROMIA4.0

Pour la procédure de re-calcul des référentiels outil, voir le document séparé : Procédure de correction d'un référentiel outil sur baie S4CPlus ABB-PROMIA.

2. CONVENTIONS

2.1 Icônes utilisées dans ce document :

: Comment faire ?

: Quelle utilisation ?

: A noter...

: Important !

: Pour plus d'information, voir aussi...

: Attention, danger !

: Trucs et astuces

2.2 Choix dans les menus

Mode opératoire général :

Les 10 premiers choix dans les menus de la baie comportent toujours un numéro associé.

✓ Pour effectuer un choix dans un menu, vous pouvez :

- soit utiliser les touches de navigation  et valider par 

- soit taper le numéro du choix dans la liste et valider par 

Dans ce document les numéros associés à un choix de menu seront donnés en gras et entre crochets, avec le symbole ✓.

Exemples :

Sélectionner le menu " **VOIR** " et " **MODULE** " **[✓ 6]**
sélectionner **Sujets** / **Signaux d'E/S [✓ 3]**

3. CHARGEMENT DE L'APPLICATION PROMIA

Ce chapitre détaille la procédure d'installation d'une application PROMIA vierge.

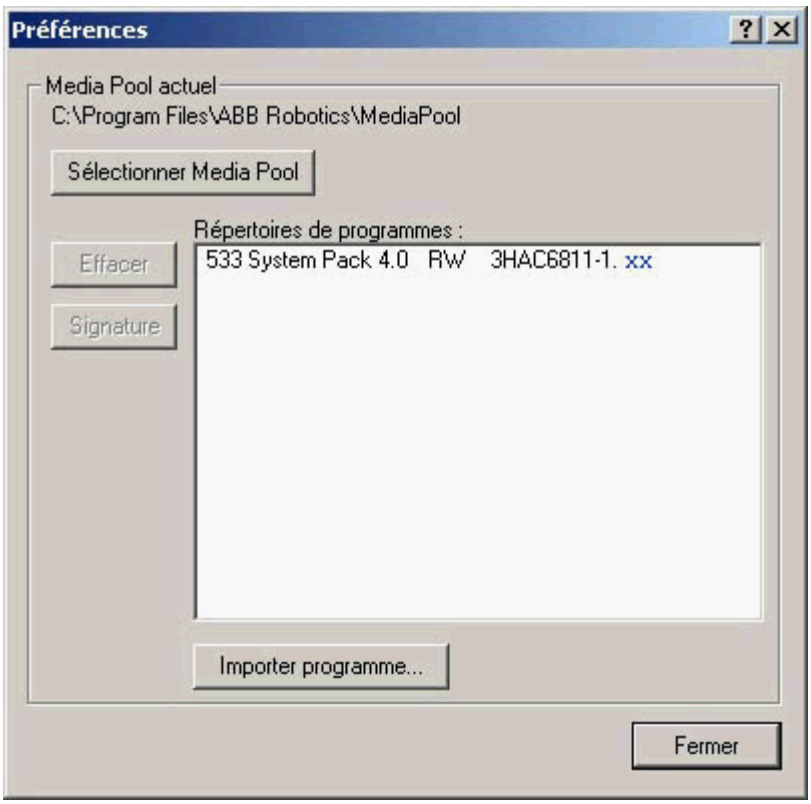
En règle générale, la dernière version validée de l'application aura été pré-chargée par ABB et cette procédure n'est à suivre qu'en cas de réinstallation ou de mise à jour.

Tout comme le logiciel de base, l'application se charge via le logiciel RobInstall. La partie de l'installation concernant le logiciel de base est expliquée dans le « Chapitre Système de commande / Installation du logiciel du système de commande » du « Manuel d'installation» présent dans le CDROM du référentiel.

3.1 Importation de l'application.

La première phase consistera à importer le répertoire contenant l'application, dans le répertoire « mediapool » (en général "C:\Program Files\ABB Robotics\MediaPool").

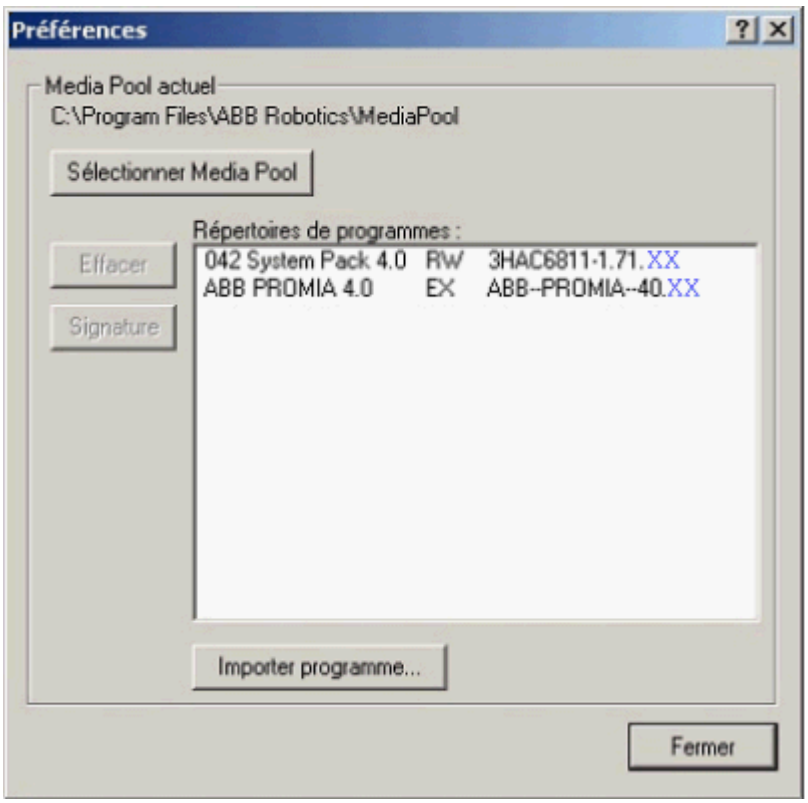
Lancer le logiciel Rob*Install* puis sélectionner « Préférences ».



Vérifier le Media Pool.

Sélectionner "Importer Programme"

Sélectionner le répertoire contenant la version d'application à installer,
(ABB--PROMIA--40.XX) puis tapez "OK".

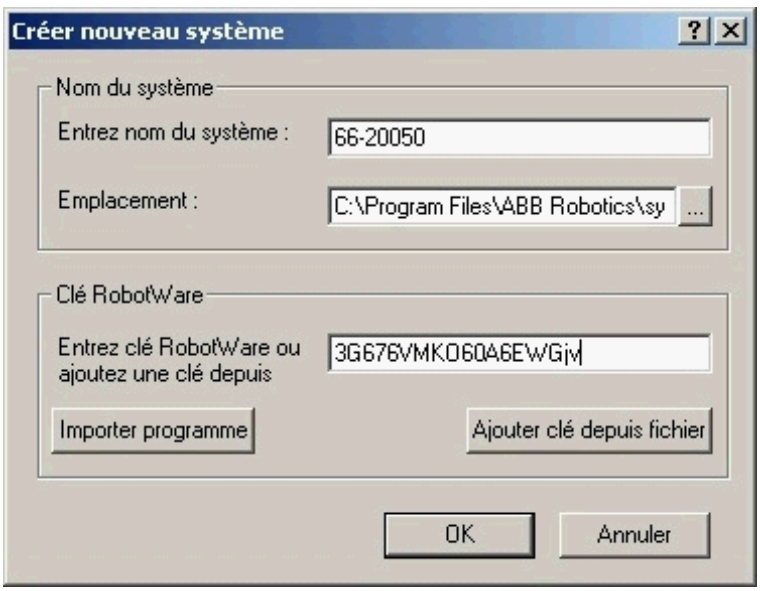


L'application est maintenant correctement importée, vous pouvez fermer la fenêtre et passer à la phase 2.

3.2 Création du système.

La phase 2 consistera à créer votre système.

Sélectionner "Nouveau".

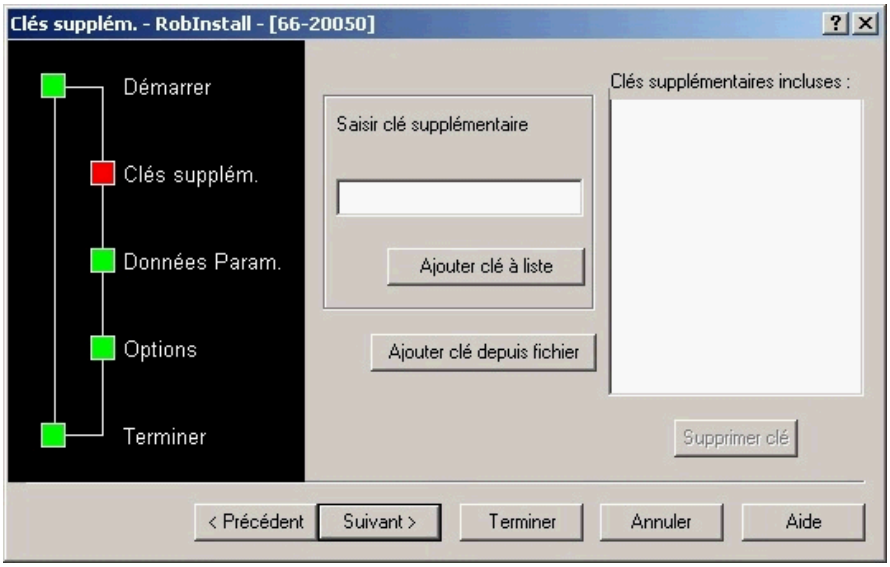


Entrer le nom du système

Vérifier qu'il sera bien créé sous le répertoire « xxxx\system », en général "C:\Program Files\ABB Robotics\System"

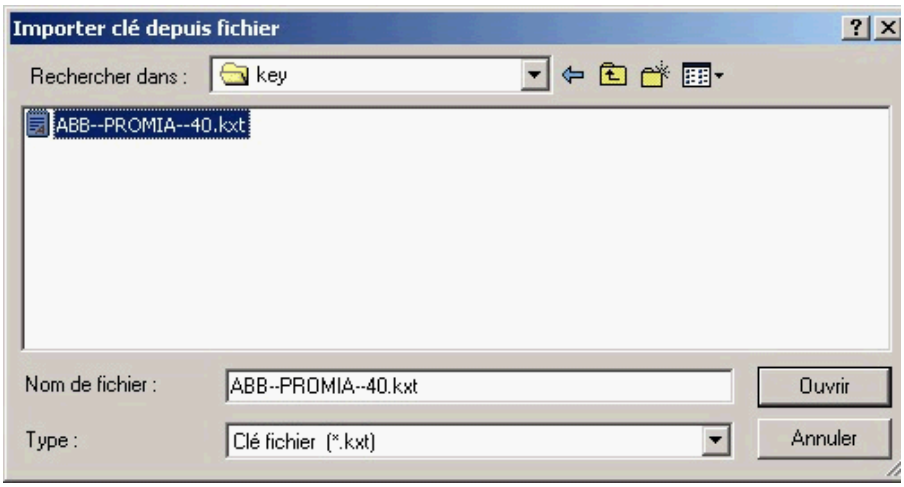
Faire "Ajouter clé depuis fichier" et sélectionner la clef du **robot** (KeyDisk) qui est jointe au robot.

Puis tapez "OK"



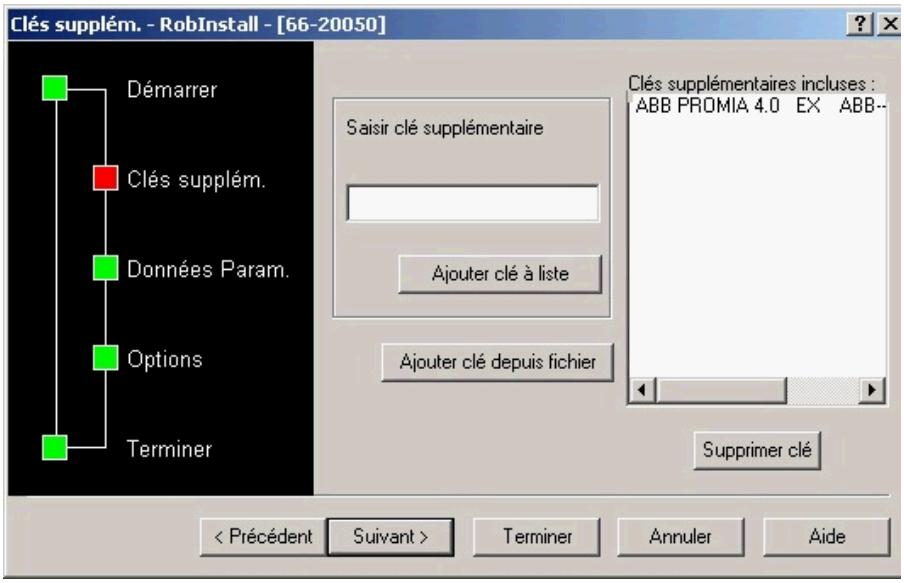
Il s'agit maintenant d'insérer la clef externe propre à l'application.

Faire "Ajouter clé depuis fichier".

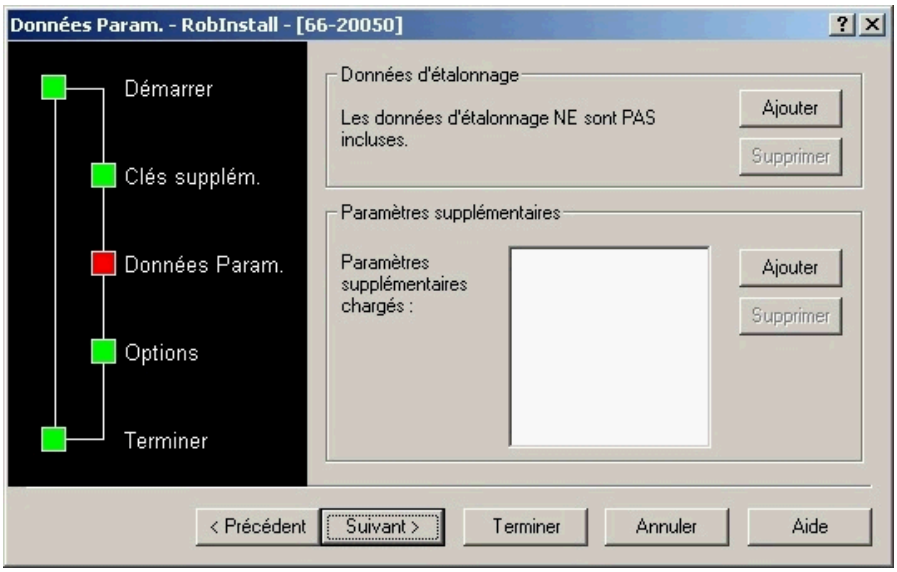


Sélectionner le fichier " ABB--PROMIA--40.kxt" correspondant à l'application (dans répertoire Key du CDROM)

Puis faire "Ouvrir".

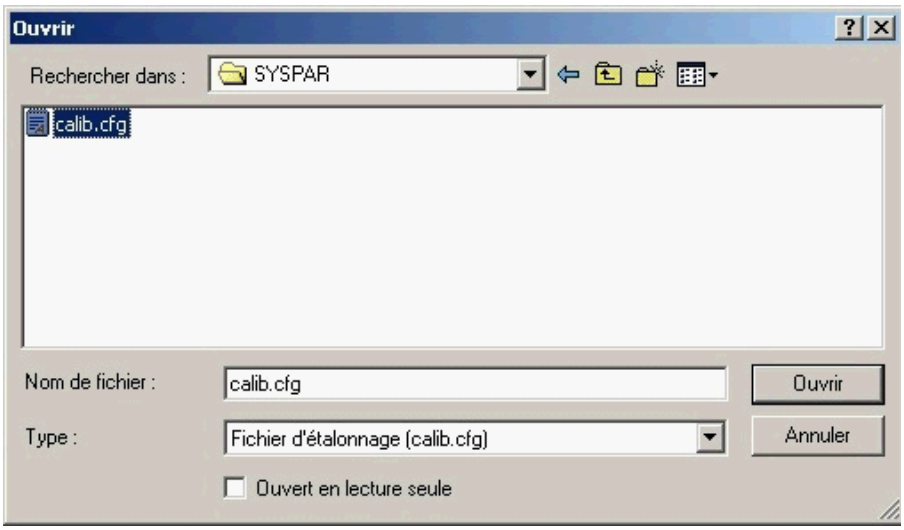


Une fois la clé présente, faire "Suivant".



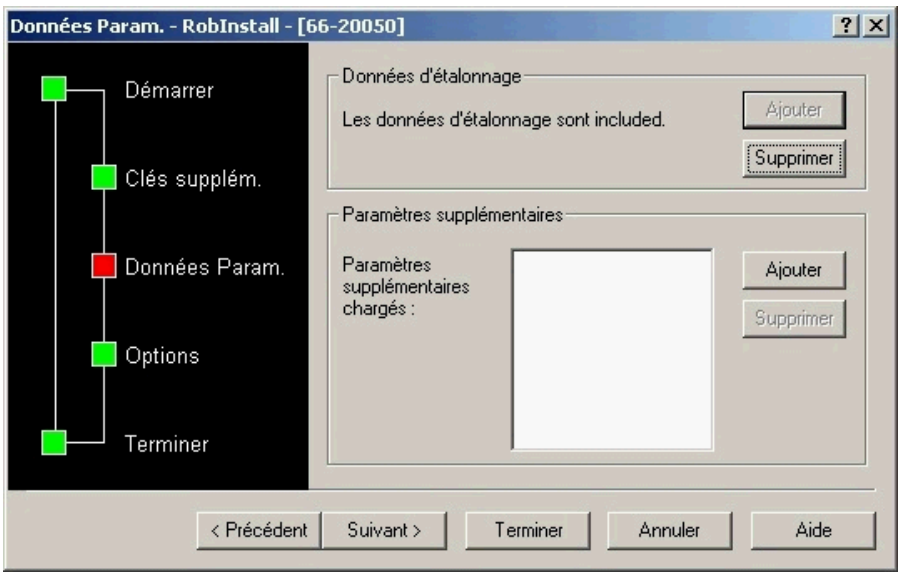
Si les données d'étalonnage du robot n'ont pas été incluses

Faire « Ajouter »

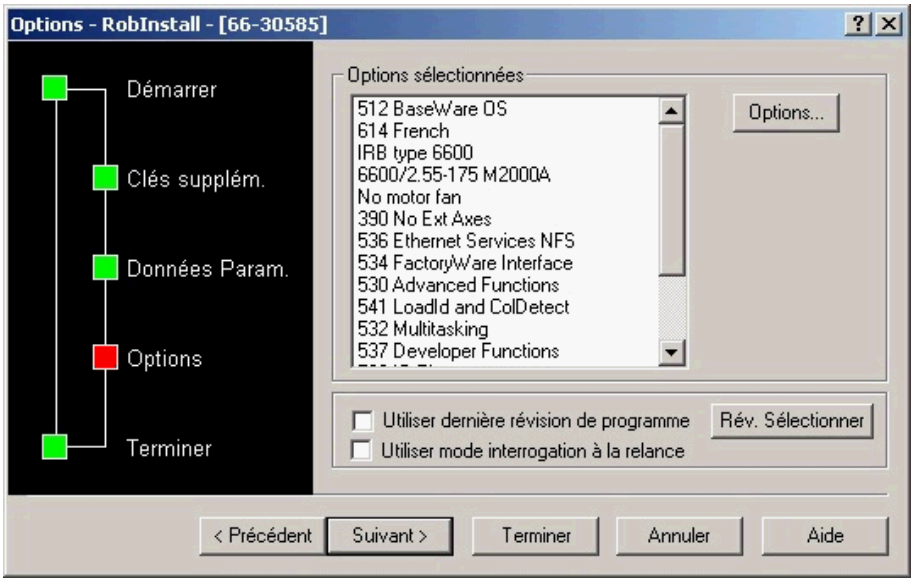


Sélectionner le fichier « calib.cfg » du robot.

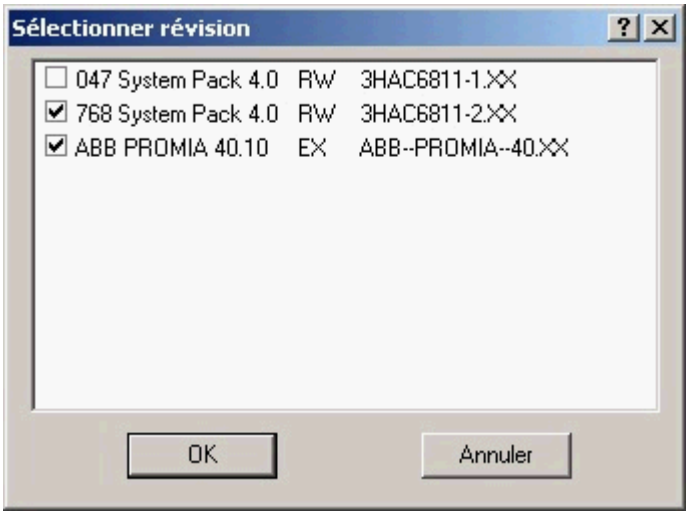
Faire « Ouvrir »



Une fois les données d'étalonnage incluses, faire "Suivant".

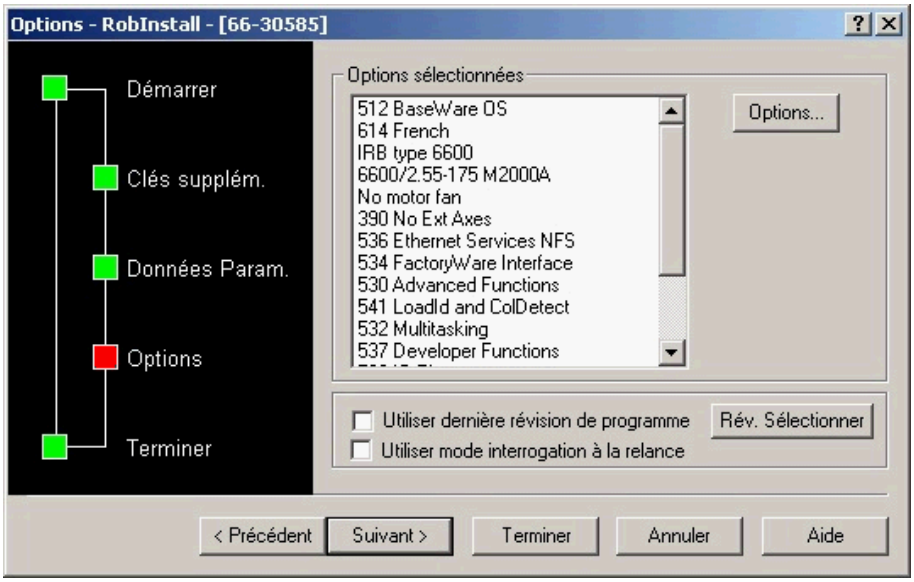


Si plusieurs versions de logiciel existent sur le PC, utiliser “Rev. Selectionner” pour spécifier celle qui doit être installée.



Sélectionner les révisions du logiciel de base et de l’application à installer.

Faire « OK »

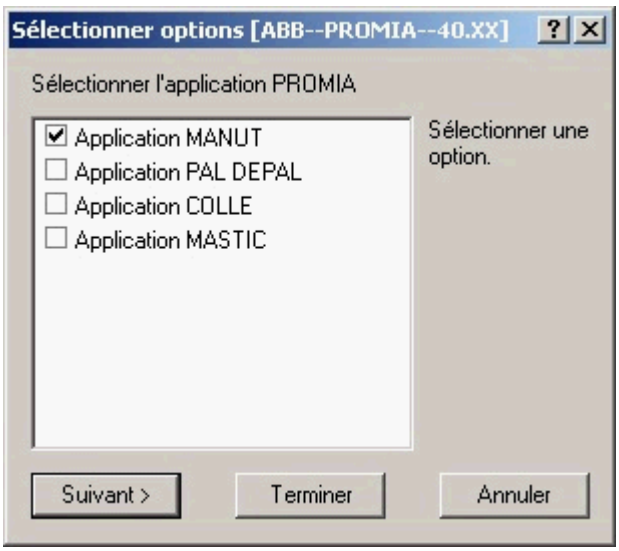


L’étape suivante est le choix du « métier », c.a.d le type d’application. Appuyer sur le bouton "Options".

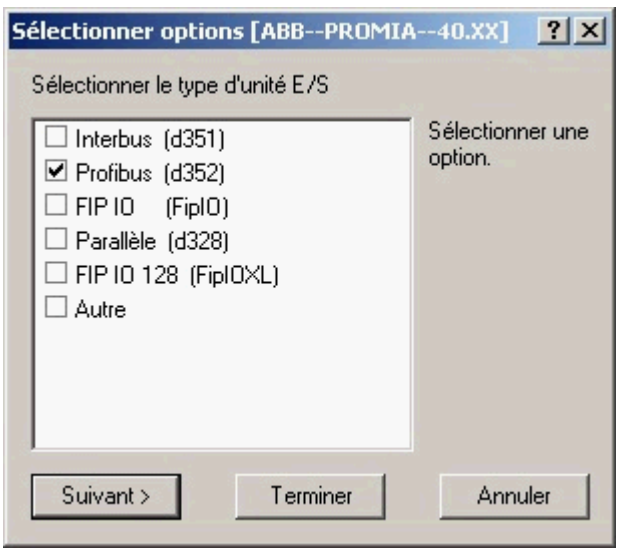
Ce bouton ‘Options’ permet également de sélectionner ou de désélectionner les options du logiciel de base.

<u>Attention aux éventuelles erreurs de manipulation</u>.

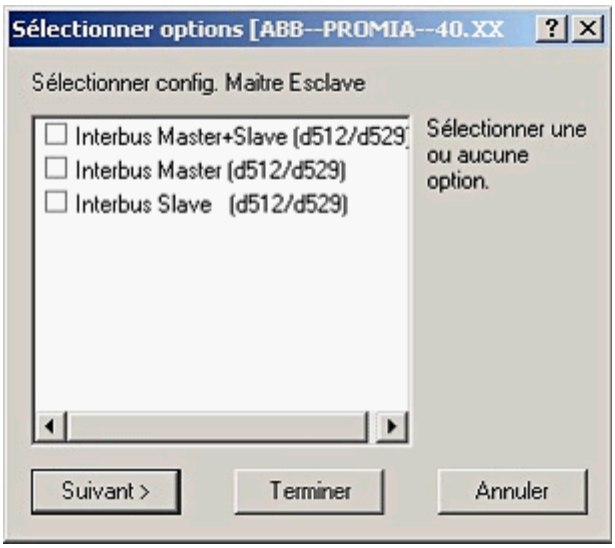
A priori il n’est pas besoin de modifier les options du logiciel de base, à part éventuellement la langue ou le type de manipulateur pour des essais ponctuels. En cas de doute, consulter ABB.



Dans cette fenêtre, sélectionner le type d'application. Tapez "Suivant" (ou « Next » dans une version anglaise de Robinstall)



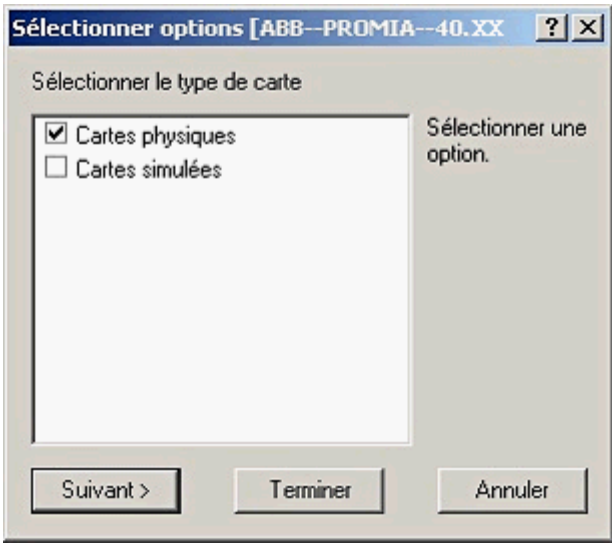
Choisir la configuration de la carte servant au dialogue avec l'automate. Taper "Suivant"



Le choix « Autre » permet de choisir la configuration Interbus Maître+Esclave si l'option 247/248 Interbus Master+Slave a été commandée pour ce robot.

Choisir le type de configuration de la carte DSQC512.

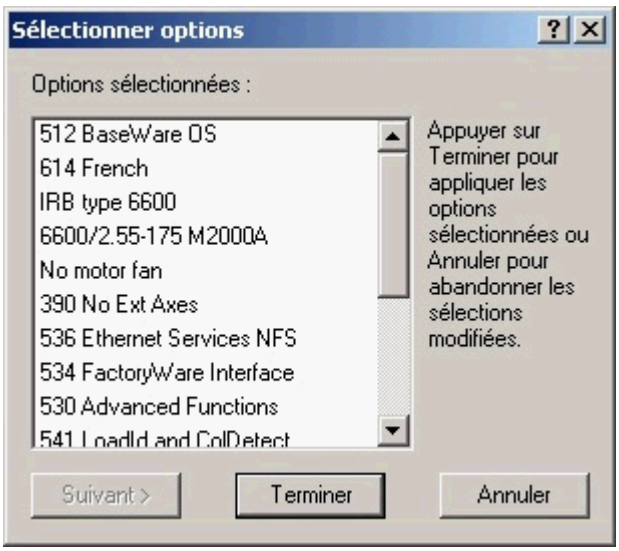
Taper "Suivant"



Le type de cartes doit être "physiques", le type simulé ne sert que pour le mode "DEBUG"

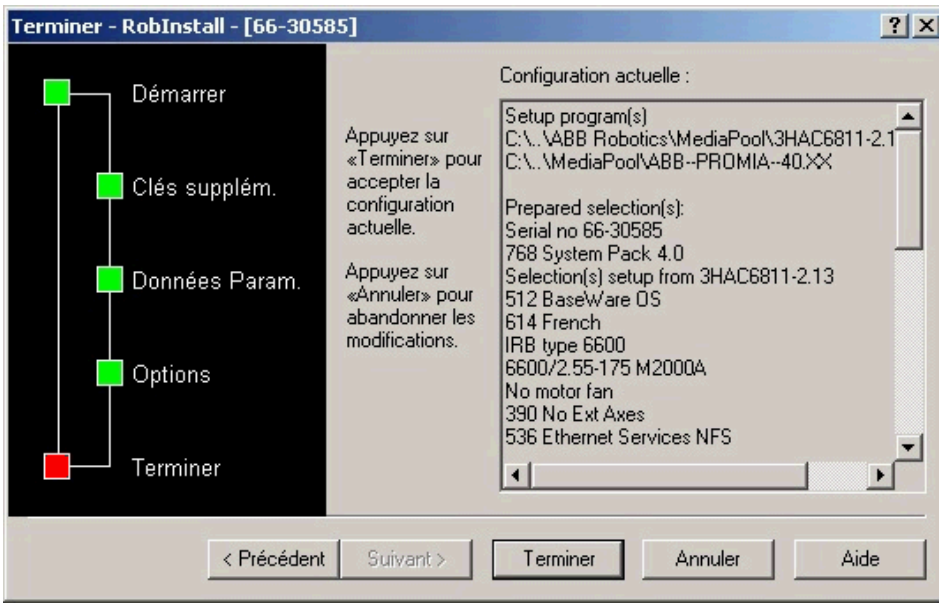
Attention : dans le type de configuration « cartes simulées » aucun dialogue avec les éléments externes à la baie n'est possible.

Taper "Terminer" puis "Suivant".



Vérifier les options choisies

Et taper "Terminer".



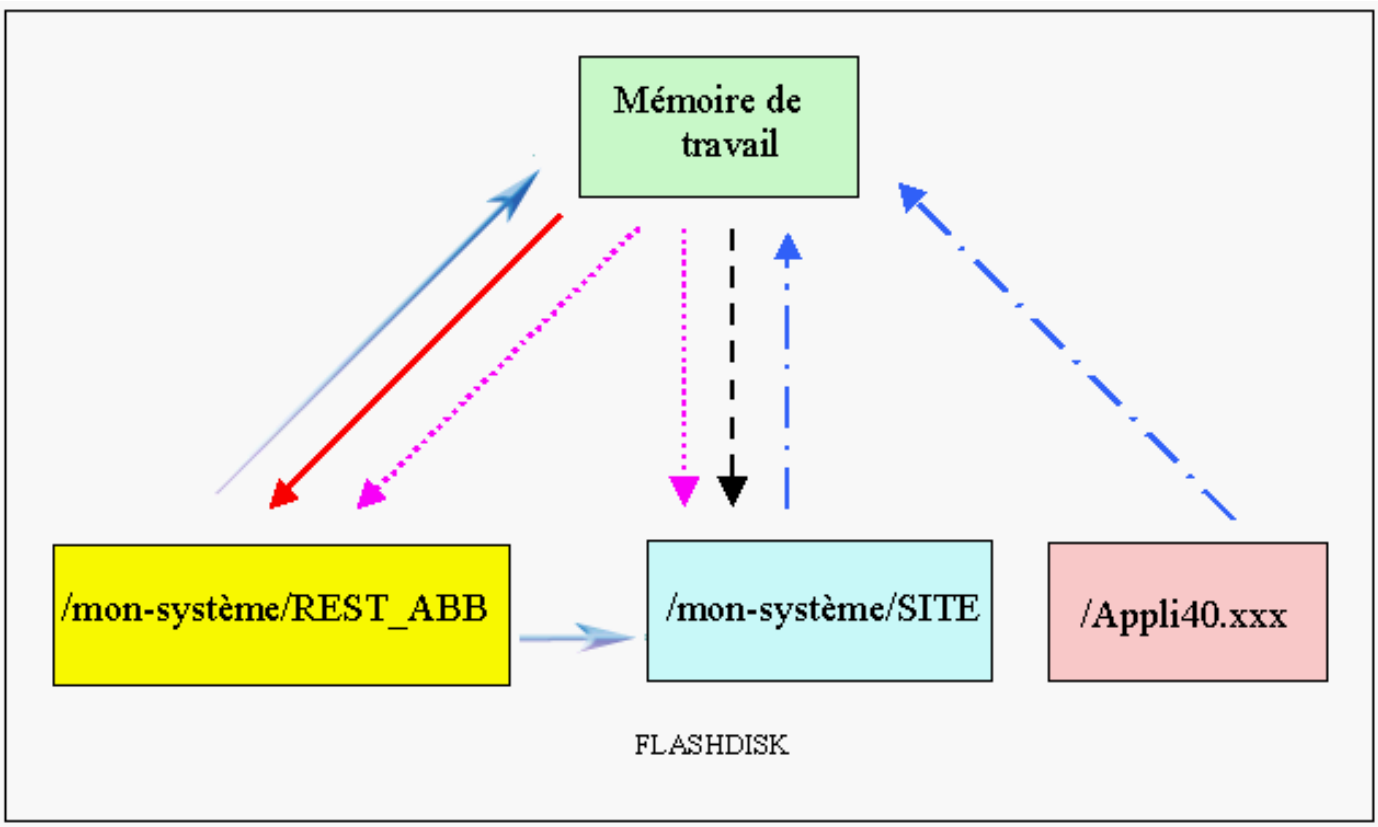
Une fois votre configuration terminée, taper "Terminer".

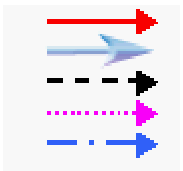
Votre système est maintenant prêt à être téléchargé dans votre Baie S4CPlus.

L'installation de l'application dans votre système est terminée.

4. SAUVEGARDE DE L'APPLICATION PROMIA

4.1 Synoptique de la gestion mémoire de la Baie





Backup : fonction "Sauvegarde" du menu Maintenance
Restauration : fonction "Restauration" du menu Maintenance
Sauvegarde module : sauvegarde d'un module, dans la fenêtre programmation.
Menu « Sauver » de l'écran de base : sauvegarde simplifiée modules programme (ou métier)
P-START : Après un chargement logiciel, ou un redémarrage avec saisie 258
Sur la notion de P-Start, se référer au Guide Utilisateur ABB, chapitre Maintenance -Redémarrer

Avant tout backup, il est fortement recommandé de réaliser une sauvegarde des modules PRG_MVT et MAINTENA via l'écran de base (cf.4.2.1), afin d'assurer la cohérence du répertoire avec le contenu du backup et éviter toute mauvaise surprise en cas de P-Start.

La sauvegarde par backup sur disquette est à éviter. Si l'utilisateur désire avoir une sauvegarde sur disquette, il est préférable de procéder comme indiqué au paragraphe 4.3 et de copier par la suite le répertoire REST_ABB sur la disquette. Il est également possible de transférer ce backup sur votre PC via FTP.

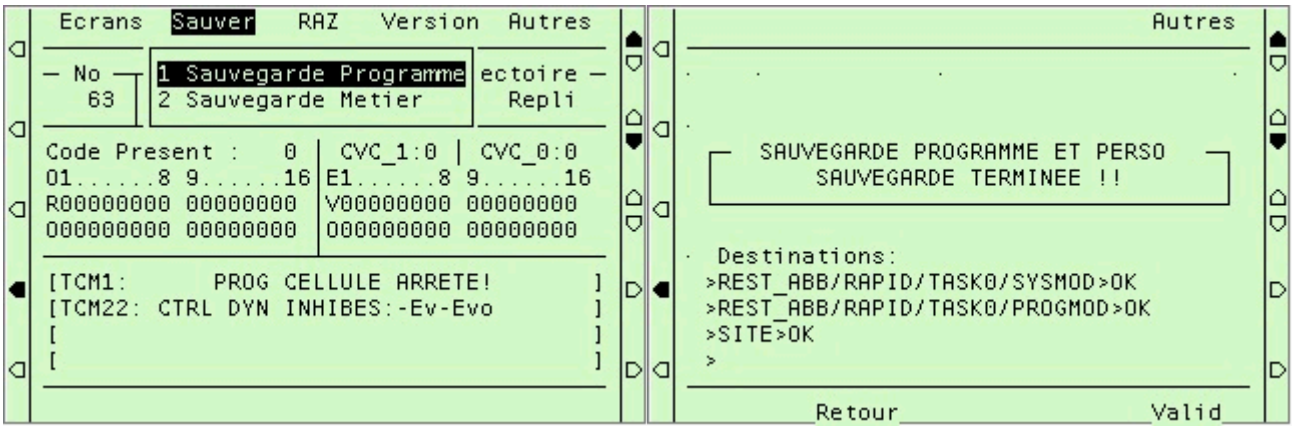
4.2 Sauvegarde des modules PRG_MVT.MOD et MAINTENA.SYS

La sauvegarde de ces modules est possible depuis l'écran de base de l'application. C'est l'opération la plus simple pour sauver les modules et celle qui garantit le mieux la cohérence des informations entre le répertoire SITE et le répertoire de backup.

Toutefois, il est toujours possible d'effectuer manuellement la sauvegarde des modules PRG_MVT.MOD et MAINTENA.SYS sur une disquette ou sur le flashdisk depuis la fenêtre de programmation.

4.2.1 Sauvegarde simplifiée

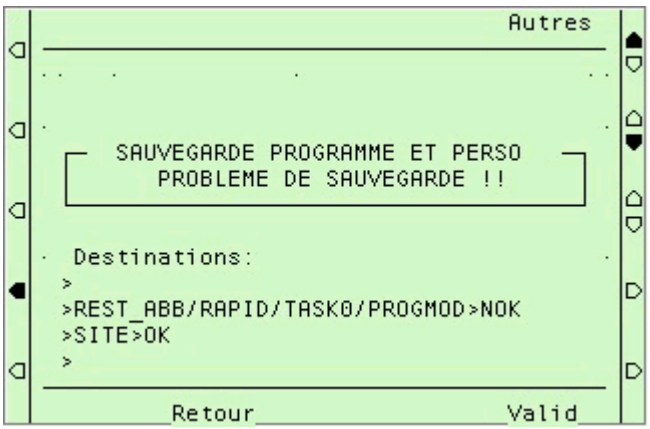
La sauvegarde des modules PRG_MVT et MAINTENA est possible depuis l'écran de base de l'application, dans le menu « **Sauver / Sauvegarde Programme** ». Dans ce cas, les modules sont sauvés à la fois dans le répertoire SITE et le répertoire de backup REST_ABB.
Les éventuels modules déclarés dans « Sauve_Perso » sont également sauvés selon la programmation réalisée (SITE, REST_ABB ou autres)



Le menu **« Sauver / Sauvegarde Metier »** sauvegarde les éventuels modules métier dans le système courant et REST_ABB

La sauvegarde simplifiée est accessible pendant l'exécution d'un cycle. Dans ce cas un message informe l'utilisateur que la demande est mémorisée : « SAUVE PROGRAM & PERSO MEMORISEE » ou « SAUVE METIER MEMORISEE ». Une fois le point de rebouclage atteint, la sauvegarde est réalisée automatiquement.

Si le backup REST_ABB n'est pas présent, la procédure de sauvegarde simplifiée va échouer sur un « Problème de sauvegarde »



4.2.2 Sauvegardes manuelles

4.2.2.1 Sauvegarde sur une disquette de travail.

- Se placer dans la fenêtre de programmation
- Sélectionner le menu " **VOIR** " et " **MODULE** " **[✓ 6]** pour faire apparaître tous les modules programmes.
- Sélectionner le module PRG_MVT ou MAINTENA avec 
- Sélectionner le menu " **FICHIER** " et " **SAUVER MODULE SOUS** " **[✓ 0]** puis  .
- Sélectionner le " **FLP1** " avec la touche " **UNITE** ".
- Valider par " **OK** ".

4.2.2.2 Sauvegarde manuelle dans le répertoire " SITE " dans le Flashdisk

- Se placer dans la fenêtre de programmation
- Sélectionner le menu " **VOIR** " et " **MODULE** " **[✓ 6]** pour faire apparaître tous les modules programmes.
- Sélectionner le module PRG_MVT ou MAINTENA avec 
- Sélectionner le menu " **FICHIER** " et " **SAUVER MODULE SOUS** " **[✓ 0]** puis  .
- Sélectionner le " **hd0a** " avec la touche " **UNITE** ".
- Appuyer sur la touche 
- Sélectionner le répertoire " **SITE** " puis 
- Valider par " **OK** ".

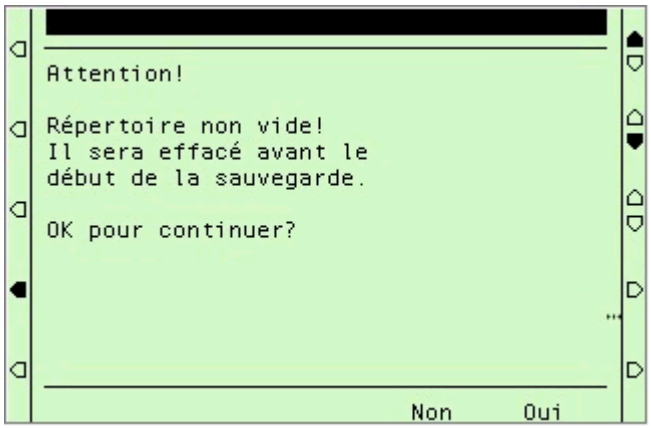
4.3 Sauvegarde (Backup)

La sauvegarde complète (« Backup ») permet de sauvegarder en une fois et dans un seul répertoire la totalité de l'application soit :

- Le contenu de toutes les tâches RAPID
- Les paramètres systèmes

Procédure de Sauvegarde

- aller en « **AUTRES FENETRES** »
- placer le curseur sur « **Maintenance** » – puis
- sélection « **fichier / Sauvegarde** » **[✓ 3]** – puis
- modifier éventuellement le répertoire de destination proposé par : , puis navigation et positionnement sur **REST_ABB** et valider par **« OK »**
- Valider par « **OK** »
- Si le répertoire est non vide, valider l'écrasement par **« OUI »**



- Attendre la fin de la sauvegarde

La sauvegarde par backup pendant l'exécution du programme mouvement est à éviter. L'utilisateur doit arrêter l'exécution du cycle avant de lancer la sauvegarde et attendre la fin du backup avant de relancer en exécution.

4.4 Procédure de restauration :

Elle recopie dans la mémoire de travail de la baie tout le contenu de la sauvegarde et met à jour le Flashdisk.

Procédure de Restauration

- Appuyer sur la touche « **AUTRES FENETRES** » 
- placer le curseur sur « **Maintenance** » – puis 
- sélectionner « **fichier / Restauration [✓ 4]** » – puis 
- Appuyer sur  pour choisir le répertoire source (en général REST_ABB)
- placer le curseur sur le répertoire à restaurer - valider par **« OK » (2 fois)**
- taper **« 0 0 7** » (mot de passe programmeur) et valider par **« OK » (2 fois)**
- attendre la fin du chargement
- la baie redémarre automatiquement

5. CONFIGURATION DE L'APPLICATION

5.1 Configuration des E/S

5.1.1 Principe de base

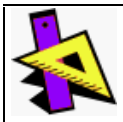
Lors du chargement initial de l'application Manutention, le fichier EIO.cfg (paramètres d'entrées-sorties) est chargé en fonction de la configuration réseau du site (Interbus S, Profibus, FIPIO ou Parallèle). Les cartes d'entrées sorties sont déclarées de la façon suivante :

- Réseau INTERBUS S :
  - "INTERBUS" : carte DSQC351 à l'adresse 10 sur le bus CAN (dialogue robot - API pour les applications Interbus)
- Réseau PROFIBUS :
  - "PROFIBUS" : carte DSQC352 à l'adresse 10 sur le bus CAN (dialogue robot - API pour les applications Profibus)
- Réseau FIPIO :
  - "FIP" : carte Fip Io (FSDC8) à l'adresse 10 sur le bus CAN (dialogue robot - API pour les applications FIP I/O)
- Liaison parallèle :
  - " Cart_Gest_Std" : carte DSQC328 à l'adresse 10 sur le bus CAN (dialogue robot - API pour les applications sans réseau)
  - " Cart_Dial_Autom" : carte DSQC328 à l'adresse 12 sur le bus CAN (dialogue robot - API pour les applications sans réseau)
- TOUS RESEAUX :
  - " MANUT" : carte DSQC328 (16E/16S) à l'adresse 13 sur le bus CAN
  - " Cart_Dial_Outil" : carte simulée (16E/16S) sur le bus simulé dédié au dialogue avec l'outil
  - "SIMBRD" : carte simulée (65E/64S/1SAna) sur le bus simulé qui contient les signaux des fonctions optionnelles utilisées dans l'application
- Autres (avec l'option 247/248 Interbus Master+Slave »:
  - "Interbus Master+Slave " : carte DSQC512 sur le BUS IBS
  - " Interbus Master"
  - "Interbus Slave"

Pour plus d'information sur la configuration des entrées / sorties par défaut,
voir le chapitre 12 : CARTES D'ENTREES SORTIES

5.1.2 Déclaration des cartes


Cette opération est à effectuer pour ajouter des cartes d'E/S supplémentaires par rapport à la configuration chargée à l'origine.



- Appuyer sur la touche « **AUTRE FENETRES** » 
- placer le curseur sur **Paramètres Systèmes** puis 
- sélectionner **Sujets** / **Signaux d'E/S [✓ 3]** puis 
- sélectionner **Types** / **Unités [✓ 1]** puis 
- valider **Ajouter** en bas de l'écran
- remplir les renseignements relatifs à la carte ajoutée
- sélectionner Fichier / **Redémarrer** puis 
- Faire OK et attendre la fin du chargement.

5.1.3 Contraintes sur les entrées/sorties

Il est possible de déplacer les entrées/sorties, mais certaines règles doivent être respectées.


Les ordres API, OR1 à OR16, doivent avoir des adresses contiguës et former le groupe « xOrdres ».

Les événements API, EV1 à EV16, doivent avoir des adresses contiguës et former le groupe « xEvents ».

Les ordres outil, OROU1 à OROU16 doivent être « contigus par morceau ». Ils doivent pouvoir être regroupés dans au maximum 4 groupes, G1OrdOut à G4OrdOut, de longueur variable.

Le mapping des entrées/sorties dédiées au dialogue automate ne doit pas être modifié. Cela concerne les 32 premiers bits d'entrées et de sortie et les 16 bits du mot commentaire suivi (MOT_COM).


S'il s'avère impossible de respecter physiquement ces contraintes, une astuce consiste à déclarer ces groupes de sorties sur une carte simulée et à cross connecter les signaux individuels correspondants sur les sorties physiques réellement utilisées.

5.1.4 Déplacement des entrées et sorties au pupitre


Cette procédure est utilisée pour déplacer les E/S utilisées des cartes simulées vers les cartes réelles.

- Appuyer sur la touche « **AUTRE FENETRES** » 
- placer le curseur sur **Paramètres Systèmes** puis 
- sélectionner **Sujets** / **Signaux d'E/S [✓ 3]** puis 
- sélectionner **Types** / **Signaux Utilisateurs [✓ 2]** puis 
- placer le curseur sur le signal à modifier puis 
- placer le curseur sur TYPE SIMBRD puis 
- placer le curseur sur Nom d'Unité - valider par OK
- placer le curseur sur n° Signal puis 
- modifier le n° valider par **OK**
- sélectionner Fichier / **Redémarrer** puis 
- faire OK et attendre la fin du chargement

5.1.5 Déplacement des entrées et sorties hors ligne

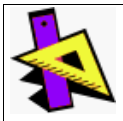

Les fichiers de configuration des entrées sorties (les fichiers EIO.cfg) peuvent être modifiés hors ligne par un éditeur de texte. Dans ce cas, faire attention à conserver l'attribut « -store » pour les sorties suivantes :

- xOrdres (Ordres API)
- G1OrdOut à G4OrdOut (Ordres outil)
- PIL1 à PIL10 (pilotages séquences)
- Repli (robot à la position de repli)
- Reb_prg (robot à la position de rebouclage)
- WREPLI

5.2 Configuration des modules système

L'intégrateur dispose de 3 modules système pour configurer l'application. Ces 3 modules, "MAN_CONF.SYS", "DEF_SITE.SYS" et "PERSO.SYS" , se trouvent dans le répertoire « SITE » et sont chargés aussi bien dans la tâche d'avant plan que dans la tâche de surveillance. Ils ne peuvent être modifiés que par un éditeur, hors ligne.

5.2.1 Réalisation de la configuration



Pour réaliser la configuration des modules système, il faut procéder comme suit :

- Editer le fichier "MAN_CONF.SYS", "DEF_SITE.SYS" ou "PERSO.SYS" sur un PC à l'aide d'un éditeur de texte, type Wordpad (ABB préconise UltraEdit32).
- Effectuer la personnalisation du fichier : modifier les routines et / ou variables (voir les paragraphes 5.2.2 à 5.2.4)
- Sauvegarder le fichier sur une disquette de travail
- Vérifier et charger le module : voir le paragraphe 5.2.5

5.2.2 Configuration du module MAN_CONF.SYS

5.2.2.1 Configuration des codes programme réservés

La gestion des codes programme mouvement correspond à la sélection du travail à faire exécuter par le robot en fonction de l'état machine et de l'état pièce de l'environnement du robot.
Cette sélection est réalisée par un système externe à la baie robot qui génère une information appelée "code programme" qui sera transmise à la baie robot. (la baie ne réalisant que le départ sur le programme demandé).

Cependant certains numéros de programmes sont réservés et gérés par l'application. Ils ne doivent donc pas être utilisés pour les programmes de travail. Ces codes correspondent à la demande de repli et aux programmes de service 1 à 4.
Les valeurs fournies par défaut sont celles utilisées dans les lignes de carrosserie - tôlerie; toutefois il est possible de modifier ces valeurs, notamment pour les installations où le nombre de codes des programmes de travail est supérieur à 64. Dans ce cas, contacter le client final afin de définir de concert les codes à utiliser. Il suffira alors de modifier les valeurs des constantes suivantes :

```
CONST num CP_REPLI:=63;
CONST num CP_DemServ1:=62;
CONST num CP_DemServ2:=61;
CONST num CP_DemServ3:=60;
CONST num CP_DemServ4:=59;
```

5.2.2.2 Configuration de la manutention - Généralités

La configuration de la manutention se fait dans le fichier "**MAN_CONF.SYS**" : l'intégrateur peut y paramétrer la gestion des préhenseurs au moyen des séquences et la gestion des contrôles de présence pièce.
Les E/S dédiées à cette fonction sont par défaut déclarées sur la carte " **MANUT** " :

- 14 entrées banalisées utilisables au choix pour les contrôles des actionneurs ou les contrôles des présences pièce (CTRL1 à CTRL14). CTRL14 n'est pas disponible avec l'option climatiseur.
- 10 sorties de pilotage / dé-pilotage des actionneurs (PIL1 à PIL10)

- Ces entrées/Sorties peuvent être renommées au gré de l'intégrateur, mais les noms choisis doivent être limités à 8 caractères pour un affichage correct sur les écrans de l'application.

- Cette configuration par défaut correspond à la manutention de base 5 séquences. Il est cependant possible d'ajouter autant de cartes physiques et de signaux d'entrées / sorties que nécessaire pour une application particulière, et ce dans la limite de 8 contrôles présence pièce et 16 séquences (16 actionneurs et 16 capteurs).

5.2.2.3 Configuration générale des séquences (nombre, mode déverminage, tolérance)


Cette partie de la configuration décrit le comportement général des séquences:

- le nombre de séquences : NB_SEQ
- le comportement, réel ou simulé, des séquences en mode déverminage : Tab_Inhib_Seq, un tableau de variables booléennes qui contient pour chacune des séquences TRUE si la séquence est simulée, ou FALSE si la séquence est réellement activée.
- la temporisation qui remplace la séquence en mode déverminage : TEMPO_DEVERM.
  
  cette temporisation sert également à simuler le temps d'attente des contrôles présence pièce en mode déverminage
- le temps de latence des séquences, TEMPO_SEQ_ATT. Il s'agit du temps de réponse de l'équipement en régime transitoire : pendant cette durée, l'absence de positionnement correct des entrées correspondantes n'entraîne pas de remontée de défaut.

La configuration est livrée par défaut avec les paramètres suivants :

```
!#####################################
! Definition du nombre de prehenseurs
!#####################################
! NB_SEQ PAR DEFAUT 16!
CONST num NB_SEQ:=16;
! Tempo techno en deverminage sur SEQ et CPP (en secondes)
CONST num TEMPO_DEVERM:=2;
! Tempo avant info suivi attente sequence (en secondes)
CONST num TEMPO_SEQ_ATT:=2;
! Tableau d'inhibition des séquences en déverminage ECRAN 8
PERS bool Tab_Inhib_Seq{16}:=[TRUE,TRUE,TRUE,TRUE,TRUE,TRUE,TRUE,TRUE,
     TRUE,TRUE,TRUE,TRUE,TRUE,TRUE,TRUE,TRUE];
```

5.2.2.4 Définition des séquences.

Dans le fichier MAN_CONF.SYS, l'intégrateur doit renseigner les routines Act_Seq ..1 à 16 pour définir comment activer ou désactiver une séquence et les routines Tst_Seq ..1 à 16 pour définir comment tester son état.

- **Pilotage des séquences**

Les routines Act_Seq..1 à 16 sont appelées pour la gestion de l'activation ou désactivation des actionneurs des séquences 1 à 16. Ces routines ont comme argument l'état souhaité de la séquence : « **DEACTIVATE** » (par convention, mettre l'actionneur en position « repos ») et « **ACTIVATE** » (par convention, mettre l'actionneur en position « travail »).

L'intégrateur doit programmer pour chacun de ces deux états d'une séquence des instructions spécifiques permettant de piloter à 1 ou à 0 les actionneurs mis en œuvre, et ce en utilisant respectivement les instructions « **ActiveAct** » et « **DesactiveAct** » dont l'argument est le nom de la sortie à actionner.

Exemple :

Fourniture de base :

```
!######################
! Active la sequence 1
!######################
PROC Act_Seq1(
  num etat)

  TEST etat
  CASE DEACTIVATE:
    ! a definir par l'utilisateur
  ...
  CASE ACTIVATE:
    ! a definir par l'utilisateur
       ...
  DEFAULT:
  ENDTEST
ENDPROC
```

Personnalisé :

```
!######################
! Active la sequence 1
!######################
PROC Act_Seq1(
  num etat)

  TEST etat
  CASE DEACTIVATE:
    ! a definir par l'utilisateur
...
      DesactiveAct  PIL1;
      ActiveAct  PIL2;
      DesactiveAct  PIL3;
      ActiveAct  PIL4;
  CASE ACTIVATE:
    ! a definir par l'utilisateur
...
      ActiveAct  PIL1;
      DesactiveAct  PIL2;
      ActiveAct  PIL3;
      DesactiveAct  PIL4;
  DEFAULT:
  ENDTEST
ENDPROC
```

➢ Contrôle des séquences

Les routines Tst_Seq..1 à 16 sont appelées pour le contrôle des capteurs programmés dans les deux états possibles de chaque séquence .

Principe : Chaque capteur peut être contrôlé à l'état « 1 » ou « 0 ». Pour le contrôle d'une séquence un 'ET' logique est opéré par le système entre les contrôles de chaque capteur concerné par cette séquence. Si les équations logiques des séquences sont plus complexes que ce simple 'et' logique, il sera nécessaire de définir en plus des cross-connexions dans les paramètres d'entrées / sorties, afin de se ramener à des 'ET' logiques dans les routines de contrôle.

Pour ce faire, l'intégrateur doit programmer pour les deux états de pilotage, « **DEACTIVATE** » (repos) et « **ACTIVATE** » (actif), des instructions spécifiques permettant de contrôler les capteurs : il s'agit de l'instruction « CtrlSeq» dont les arguments sont le nom de l'entrée à tester (CTRL1, DETECT1) et l'état attendu (REPOS = « 0 » ou ACTIF = « 1 »).

Les instructions standards « InitCtrlSeq » et « CtrlSeqOK » qui figurent dans les routines Tst_Seq… ne doivent en aucun cas être supprimées. L'ordre des contrôles utilisé dans DEACTIVATE doit être réutilisé dans ACTIVATE (ex : CTRL1 puis CTRL2)

Exemple :

Fourniture de base :

```
!####################
  ! Test de la sequence 1
  !####################
  PROC Tst_Seq1(
    num etat,
    INOUT bool Controle)

    InitCtrlSeq;
    TEST etat
    CASE DEACTIVATE:
      ! Test de la sequence 1
      ! a definir par l'utilisateur
    ...
      Controle:=CtrlSeqOK();
      RETURN;
    CASE ACTIVATE:
      ! Test de la sequence 1
      ! a definir par l'utilisateur
    ...
      Controle:=CtrlSeqOK();
      RETURN;
    DEFAULT:
      Controle:=TRUE;
    ENDTEST
    Controle:=FALSE;
  ENDPROC
```

Personnalisé :

```
!####################
! Test de la sequence 1
!####################
PROC Tst_Seq1(
  num etat,
  INOUT bool Controle)

  InitCtrlSeq;
  TEST etat
  CASE DEACTIVATE:
    ! Test de la sequence 1
    ! a definir par l'utilisateur ...
    CtrlSeq CTRL1,REPOS;
    CtrlSeq CTRL2,ACTIF;
    Controle:=CtrlSeqOK();
    RETURN;
  CASE ACTIVATE:
    ! Test de la sequence 1
    ! a definir par l'utilisateur ...
    CtrlSeq CTRL1,ACTIF;
    CtrlSeq CTRL2,REPOS;
    Controle:=CtrlSeqOK();
    RETURN;
  DEFAULT:
    Controle:=TRUE;
  ENDTEST
  Controle:=FALSE;
ENDPROC
```

5.2.2.5 Définition des contrôles présence pièce.

La configuration des signaux de présence pièce se fait par l'utilisation de l'instruction DefineCPP dans la routine Enreg_CPP du module MAN_CONF.SYS.

Par défaut aucun contrôle présence pièce n'est défini, et la routine Enreg_CPP fournie en standard est la suivante :

```
!#######################
! Initialisation des CPP
!#######################
PROC Enreg_Cpp()
  ! Noms des signaux jusqu'à 8 différents
  ! a définir par l'utilisateur...
  !DefineCPP 1,CPP1;
  !DefineCPP 2,CPP2;
ENDPROC
```

Pour personnaliser cette routine à votre site, il suffit de créer autant d'appels à l'instruction DefineCPP que nécessaire.

Programmation d'une instruction DefineCPP

Syntaxe :

DefineCPP nCPP signal

***Exemple* : DefineCPP 1,ctrl_v1 ;**

Paramètres obligatoires :

nCPP	Numéro du contrôle pièce à définir.
signal	Signal d'entrée logique associé à ce contrôle présence pièce

5.2.3 Personnalisation du module DEF_SITE.SYS

5.2.3.1 Personnalisation des écrans

Les messages présents dans les écrans utilisateur sont personnalisables dans le module Def_Site (par PC).

Il est impératif de respecter le nombre exact de caractères.

```
! Texte des sequences: 15 caracteres max
!                   "123456789012345"
CONST string SEQ1:="SEQUENCE1        ";
….
CONST string SEQ8:="SEQUENCE8        ";

! Texte des demandes pilotage sequence: 15 caracteres max
!                     "123456789012345"
CONST string REPOS_1:="RECULE           ";
CONST string ACTIF_1:="AVANCE           ";
….
CONST string REPOS_16:="RECULE          ";
CONST string ACTIF_16:="AVANCE          ";

!========
! Ordres
!========
! Texte des ordres: 15 caracteres max
!                   "123456789012345"
CONST string ORD01:="ORDRE1           ";
 ….
CONST string ORD16:="ORDRE16          ";
!===============
! Ordres Outils
!===============
! Texte des ordres outils: 15 caracteres max
!                     "123456789012345"
CONST string ORDOU01:="ORDREOUTIL1     ";
…
CONST string ORDOU16:="ORDREOUTIL16    ";
!============
! Evenements
!============
! Texte des evenements: 15 caracteres max
!                   "123456789012345"
CONST string EVE01:="EVENT1           ";
….
CONST string EVE16:="EVENT16          ";

!==================
! Evenements Outils
!==================
! Texte des evenements outils: 15 caracteres max
!                     "123456789012345"
CONST string EVEOU01:="EVENTOUTIL1     ";
….
CONST string EVEOU03:="EVENTOUTIL16    ";
!============================
! Contrôles Présences Pièces
!============================
! Texte des CPP: 15 caractères max
!                   "123456789012345 "
```

```
CONST string CPP_1:="PRESENCE PIECE1";
…..
CONST string CPP_8:="PRESENCE PIECE8";
```

5.2.3.2 Ajout éventuel de messages d'information, de défauts et de défauts fatals

Les messages d'information et les messages de défaut apparaissent dans la zone " **MESSAGES ET DEFAUTS** " de l'écran principal, ou sur des pages spécifiques.

Ils sont classés suivant trois types importance :

- Message.(simple information)
- Défaut non fatal : le robot est arrêté, mais repartira dès disparition du défaut et du message associé, sans validation de l'opérateur.
- Défaut fatal exigeant une validation par l'opérateur une fois le défaut corrigé.

Le principe de programmation est identique pour les 3 types de messages :

- un numéro 'No' de message correspond à un texte de message PuyyyNo (Exemple : le message No 2 correspond forcément au texte PUMSG2)
- une instruction **Aff_xxxx_PU No** permet d'afficher le message numéro 'No'
- une instruction **Raz_xxxx_PU No** permet d'effacer le message numéro 'No'

Pour une meilleure lisibilité des programmes, il est conseillé d'utiliser des constantes ayant un nom approprié plutôt que les valeurs de ces constantes.

- **Messages**

Par défaut le module DEF_SITE est livré avec 5 messages utilisateur, et contient la séquence de déclaration suivante :

```
!*********************************************************************
!*              MESSAGES PERSO UTILISATEUR                           *
!*********************************************************************
!********************Déclarations des Messages**********************
!**********************"12345678901234567890123456789012345678"
CONST message PUMSG1:="PUM1: reserve ";
CONST message PUMSG2:="PUM2: reserve ";
CONST message PUMSG3:="PUM3: reserve ";
CONST message PUMSG4:="PUM4: reserve ";
CONST message PUMSG5:="PUM5: reserve ";
! Nom des messages
CONST num MESS_reserve1:=1;
CONST num MESS_reserve2:=2;
CONST num MESS_reserve3:=3;
CONST num MESS_reserve4:=4;
CONST num MESS_reserve5:=5;
! Dimension tableau des messages
CONST num MAX_MESS_PU:=5;
```

Personnalisation :

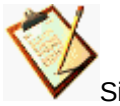
- Il est impératif de respecter le nombre maximum de caractères.
- Il est conseillé de conserver "PUMx:" en début de message, pour une identification rapide

⇨ Définition du message :

```
!*********************Déclarations des Messages*********************
!**********************"12345678901234567890123456789012345678"
CONST message PUMSG4:="PUM4: Mise en rotation outil";
```

⇨ Création d'une variable numérique qui permettra d'utiliser le message :

```
! ** Perso utilisateur **
CONST num MesRotOutil:=4;
```



Si les 5 messages pré-configurés par défaut ne suffisent pas et si l'on est donc amené à créer de nouveaux messages, il faut aussi modifier la taille du tableau de messages (c'est à dire le nombre maximal de messages utilisés) :

```
! Dimension tableau des messages
CONST num MAX_MESS_PU:=X;
```

Utilisation de ces messages (par exemple dans le module PRG_MVT):

- Affichage du message :

```
Aff_MessPU MesRotOutil;
```

- suppression du message affiché précédemment

```
Raz_MessPU MesRotOutil;
```

➢ Défauts non fatals

Par défaut le module DEF_SITE est livré avec 2 messages de défaut non fatals utilisateur, et contient la séquence de déclaration suivante :

```
!**********Déclarations des Defauts sans validation*****************
!**********************"12345678901234567890123456789012345678"
CONST message PUDEF1:="PUS1: reserve ";
CONST message PUDEF2:="PUS2: reserve ";
! Nom des defauts sans validation
CONST num DEF_reserve1:=1;
CONST num DEF_reserve2:=2;
! Dimension tableau des defauts sans validation
CONST num MAX_DEF_PU:=2;
```

Personnalisation :

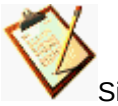
- Il est impératif de respecter le nombre maximum de caractères.
- Il est conseillé de conserver "PUSx:" en début de message, pour une identification rapide

⇨ Définition du message :

```
!********************Déclarations des Messages*********************
!**********************"12345678901234567890123456789012345678"
CONST message PUDEF2:="PUS2: Attente rotation outil";
```

⇨ Création d'une variable numérique qui permettra d'utiliser le message :

```
! ** Perso utilisateur **
CONST num DefRotOutil:=2;
```


Si les 2 messages de défaut sans validation pré-configurés par défaut ne suffisent pas et si l'on est donc amené à créer de nouveaux messages, il faut aussi modifier la taille du tableau de messages (c'est à dire le nombre maximal de messages utilisés) :

```
! Dimension tableau des messages
CONST num MAX_DEF_PU:=X;
```


Utilisation de ces messages (par exemple dans le module PRG_MVT):

- Affichage du message :

```
Aff_DefPU DefRotOutil;
```

- suppression du message affiché précédemment

```
Raz_DefPU DefRotOutil;
```

➢ Défauts fatals

Par défaut le module DEF_SITE est livré avec 3 messages de défaut fatals utilisateur, et contient la séquence de déclaration suivante :

```
!**********Déclarations des Defauts avec validation*****************
!**********************"12345678901234567890123456789012345678"
CONST message PUDVAL1:="PUV1: reserve  ";
CONST message PUDVAL2:="PUV2: reserve  ";
CONST message PUDVAL3:="PUV3: reserve  ";
! Nom des defauts avec validation
CONST num DVAL_reserve1:=1;
CONST num DVAL_reserve2:=2;
CONST num DVAL_reserve3:=3;
! Dimension tableau des defauts avec validation
CONST num MAX_DVAL_PU:=3;
```

Personnalisation :

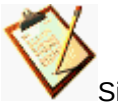

- Il est impératif de respecter le nombre maximum de caractères.
- Il est conseillé de conserver "PUVx:" en début de message, pour une identification rapide

⇨ Définition du message :

```
!********************Déclarations des Messages**********************
!**********************"12345678901234567890123456789012345678"
CONST message PUDVAL1:="PUV1: Erreur rotation outil";
```

⇨ Création d'une variable numérique qui permettra d'utiliser le message :

```
! ** Perso utilisateur **
CONST num ErrRotOutil:=1;
```



Si les 3 messages de défaut fatal pré-configurés par défaut ne suffisent pas et si l'on est donc amené à créer de nouveaux messages, il faut aussi modifier la taille du tableau de messages (c'est à dire le nombre maximal de messages utilisés) :

```
! Dimension tableau des messages
CONST num MAX_DVAL_PU:=X;
```

Utilisation de ces messages (par exemple dans le module PRG_MVT):

- Affichage du message :

```
Aff_DefValPU DefRotOutil;
```

- suppression du message affiché précédemment

```
Raz_DefValPU DefRotOutil;
```

5.2.3.3 Paramètres généraux

Dans le module DEF_SITE sont également définis des paramètres généraux, précisant les contrôles dynamiques par défaut, l'utilisation d'une deuxième entrée de contrôle air, l'utilisation d'un contrôle climatiseur ou la nécessité de prise en compte des codes cycles au point de rebouclage,.

- **Contrôles dynamiques par défaut**


Le contrôle dynamique permet d'arrêter le robot en cas de perte de l'une des informations surveillées, et de le redémarrer automatiquement dès que la condition revient.

Les contrôles dynamiques par défaut sont les contrôles dynamiques activés après un démarrage du programme ou lors de l'utilisation de l'action CTRL_DYN \OUI.

Dans la fourniture ABB, les contrôles dynamiques activés par défaut sont ceux des séquences et des contrôles présence pièce. Les événements et les événements outils ne sont par défaut pas contrôlés dynamiquement.
Il est possible de modifier ce comportement par défaut en affectant des valeurs TRUE ou FALSE à des variables booléennes prédéfinies: Active_ctl_cpp (présences pièce), Active_ctl_seq (séquences), Active_ctl_ev (événements) et Active_ctl_evo, (événements outil). Une valeur TRUE signifie actif par défaut, FALSE, inactif par défaut.

La fourniture de base ABB est la suivante :

```
!Flags d inhibition des controles dynamiques
PERS bool Active_ctl_cpp:=TRUE;
PERS bool Active_ctl_evo:=FALSE;
PERS bool Active_ctl_ev:=FALSE;
PERS bool Active_ctl_seq:=TRUE;
```

- **Prise en compte d'une seconde entrée présence air**

Le contrôle présence air se fait classiquement via l'entrée CPAIR. Pour des installations complexes, il est possible de surveiller un deuxième signal, CPAIR2.
Pour ce faire il faudra déplacer sur une carte physique cette entrée qui est par défaut définie sur une carte simulée et passer à TRUE la variable ctl_air12.

Fourniture de base ABB :

```
!Flag pour 2ème controles débit air CPAIR2
PERS bool ctl_air12:=FALSE;
```

- **Prise en compte d'une entrée climatiseur (optionnel)**

Le contrôle climatiseur se fait via l'entrée di_ClimOk.
Pour ce faire il faudra passer à TRUE la variable ctl_clim.

Fourniture de base ABB :

```
! Flag pour controle climatiseur
PERS bool ctl_clim:=FALSE;
```

Si le contrôle climatiseur n'est pas utilisé, l'entrée di_ClimOk qui est par défaut définie sur la carte MANUT peut être passée sur une carte simulée.

➢ Lecture des codes cycle sur le point de rebouclage

Les codes cycles ne sont pris en compte, par défaut, que sur le point de rebouclage. Les différents cycles devront alors commencer et se terminer sur ce point de rebouclage, selon le schéma dit “de la marguerite” appliqué en tôlerie.
Il est possible de modifier ce comportement afin de pouvoir terminer un cycle en dehors du point de rebouclage, en passant à FALSE la variable Code_Au_Reb.
Fourniture de base ABB :

```
! Flag d'autorisation de lecture des codes cycles en dehors
! du point de rebouclage programme
PERS bool Code_Au_Reb:=TRUE;
```

.

5.2.3.4 Carte d'identité robot

Bien que ces données ne soient pas directement utilisées par l'application et apparaissent sous forme de commentaire, l'intégrateur se doit de renseigner dans le fichier DEF_SITE.SYS la carte d'identité du robot. Ces renseignements seront utilisés par des utilitaires hors ligne spécifiques.
Ces données se trouvent sous la forme de commentaires au début du fichier.

```
!====================================
!=========================
! Identification du robot
!=========================
! Texte identification robot : 30 caractères maximum
!            "123456789012345678901234567890"
! ROBOT    :=" BAIE ROBOT / PORTEUR         ";
! LIGNE    :=" SOUBASSEMENT                 ";
! UNIT     :=" PLANCHER CENTRAL             ";
! ZONE     :=" ZONE 3                       ";
! DMES     :=" date de mise en service      ";
! NOM      :=" nom intégrateur              ";
! NUMROBOT :=" numéro du robot              ";
! NUMBAIE  :=" numéro de la baie            ";
! MATRICULE:=" matricule de la baie ";
!            "123456789012345678901234567890"
! NUMOUTIL1:=" numéro outil                ";
! NUMOUTIL2:="                             ";
! NUMOUTIL3:="                             ";
! NUMOUTIL4:="                             ";
! CHGOUTIL :=" OUI ou NON ?                ";
!            "123456789012345678901234567890"
! BUTMECA1N:="valeur axe sur butée méca  - ?";
! BUTMECA1P:="valeur axe sur butée méca  + ?";
! BUTELEC1N:="valeur axe sur butée élec. - ?";
! BUTELEC1P:="valeur axe sur butée élec. + ?";
! BUTMECA2N:="                             ";
! BUTMECA2P:="                             ";
! BUTELEC2N:="                             ";
! BUTELEC2P:="                             ";
! BUTMECA3N:="                             ";
! BUTMECA3P:="                             ";
! BUTELEC3N:="                             ";
! BUTELEC3P:="                             ";
!===========
```

5.2.4 Autres personnalisations : configuration du module PERSO.SYS

5.2.4.1 Principe

L'application offre à l'intégrateur une ouverture sur la tache asynchrone pour y ajouter certaines fonctions, telles que des surveillances spécifiques.
Cette ouverture se trouve dans le fichier "**Perso.sys**" qui se trouve dans le répertoire "**SITE**".

Ce module est découpé en plusieurs parties :

- Une zone de déclaration : déclaration des données applicatives de l'intégrateur.
- Une routine "**Init_perso**" : cette routine est appelée lors de la mise sous tension de la baie. Elle permet l'initialisation de cette personnalisation intégrateur : initialiser des données, positionner des ordres, des sorties, connecter une trap routine...
- Une routine "**Init_Repli_Perso**" : cette routine est appelée lorsque le robot est à la position de repli. Cela permet d'initialiser certaines variables sur cette position particulière, comme ré émettre les ordres qui sont normalement remis à zéro en fin de trajectoire de repli.
- Une routine "**Surv_Perso**" : cette routine (vide dans la fourniture de base) est appelée périodiquement par la tache asynchrone du robot. Elle permet de tester périodiquement les conditions surveillées.
- Une routine "**Raz_Perso**" : cette routine (vide dans la fourniture de base) est appelée quand la RAZ du contexte d'exécution est demandée (Touche RAZ des écrans utilisateur). Elle permet de réinitialiser les variables et les contrôles utilisés.
- Une routine "**Sauve_Perso**" : cette routine (vide dans la fourniture de base) est appelée quand la « sauvegarde programme » est demandée (Touche Sauve des écrans utilisateur). Elle permet de sauvegarder certains modules utilisateurs d'une manière simple.

ATTENTION :

LES ROUTINES DU MODULE PERSO.SYS SONT APPELEES PAR LA TACHE DE FOND !!!

⇨ **IL NE FAUT EN AUCUN CAS METTRE D'INSTRUCTIONS BLOQUANTES (WaitDi, STOP, While sans de petite tempo …) SOUS PEINE DE BLOQUER LA TACHE DE FOND.**

⇨ **TOUS LES AJOUTS D'INSTRUCTIONS DANS LA ROUTINE " SURV_PERSO " ENTRAINERONT UNE DEGRADATION DE LA PERIODE DE SCRUTATION DE LA TACHE DE FOND.**

5.2.5 Vérification et chargement des modules " RAPID"

Les procédures données dans ce paragraphe sont données pour le module MAN_CONF.SYS, mais valent également pour les modules DEF_SITE.SYS et PERSO.SYS.
Les modules PRG_MVT.MOD et MAINTENA.SYS doivent être préalablement sauvés dans SITE avant de charger des personnalisations par P-Start

5.2.5.1 Vérification du module modifié

Cette opération permet de vérifier la syntaxe du fichier personnalisé sur la disquette de travail.

sélectionner GESTION DES PROGRAMMES par la touche 

sélectionner Fichier / **Ouvrir [✓ 1]** – puis 

sélectionner " flp1 " par la touche " **Unité** "

parcourir éventuellement la disquette pour se placer dans le répertoire contenant le module – puis 

placer le curseur sur le module MAN_CONF.SYS – puis 
A la question « voulez-vous remplacer ? » => confirmer en répondant " **OUI** "

S'il n'y a plus d'erreur de syntaxe, vous pouvez passer à l'étape suivante.
Sinon un message indique les coordonnées du ou des blocs erronés.
Dans ce cas, faites les modifications nécessaires à la correction du problème.

5.2.5.2 Chargement des personnalisations sur le flashdisk

Cette opération permet de copier le fichier modifié sur le support de sauvegarde de la baie (FLASHDISK).

Appuyer sur la touche AUTRES FENETRES 

placer le curseur sur Gestionnaire **Fichiers** – puis 

sélectionner Voir " **flp1 "** - puis 

parcourir éventuellement la disquette pour se placer dans le répertoire contenant le module – puis 

placer le curseur sur le fichier " **MAN_CONF.SYS** "

sélectionner Fichier / Copier

sélectionner " **hd0a** " comme support de destination par la touche " **Unité "**

placer le curseur sur le répertoire " **SITE** " – puis 

valider copier (répondre **OK** pour la réécriture du fichier)

attendre la fin de la copie

5.2.5.3 Chargement des personnalisations en mémoire

Cette opération permet de charger le fichier modifié en mémoire de travail.

sélectionner GESTION DES PROGRAMMES par la touche

sélectionner **Fichier** / Ouvrir **[✓ 1]** – puis

sélectionner “ **hd0a** ” par la touche “ **Unité** ”

placer le curseur sur le répertoire "SITE" – puis

placer le curseur sur le module MAN_CONF.SYS – puis

A la question « Voulez-vous remplacer ? » => Confirmer en répondant “ **OUI** ”.

Appuyer sur la touche AUTRE FENETRES

Placer le curseur sur Maintenance – puis

Sélectionner Fichier / Redémarrage –puis

Appuyer successivement sur les touches **2, 5, 8**. (du pavé numérique) pour afficher le choix “ **P-START** ”.

Valider P-START**.**

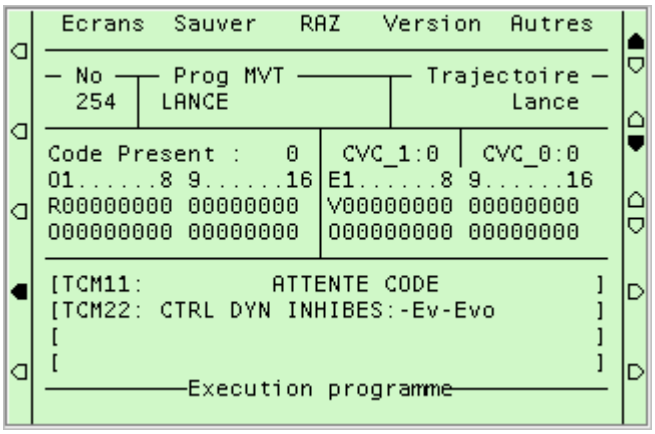
Attendre la fin du redémarrage.

Le chargement des personnalisations de la manutention est terminé.

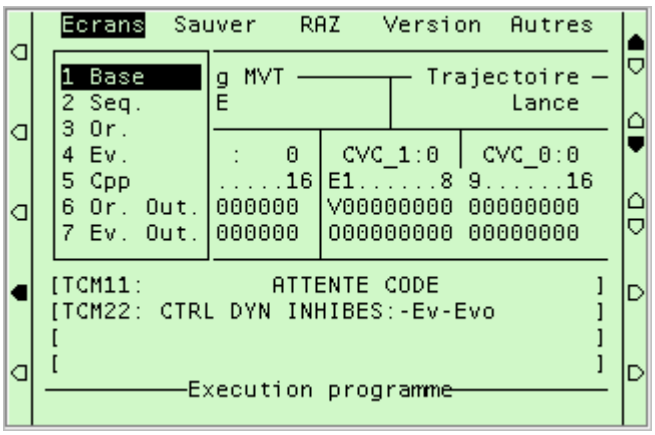
Après ce chargement, il est conseillé de faire un backup pour mettre à jour le répertoire REST_ABB.

6. ECRAN DE BASE.

Un écran commun à toutes les applications est accessible sur le pupitre robot.
Il est composé de 7 pages qui regroupent les informations sur le fonctionnement du site, les informations sur les pilotages avec la possibilité de piloter manuellement les préhenseurs, les messages de défaut ou d'alarme, la visualisation de la version de l'application et la fonction RAZ APPLI.



Ces 7 pages sont accessibles en appuyant sur la touche qui se trouve au-dessus de “ **Ecrans** ”.
Pour les ouvrir, il suffit de taper sur le numéro (pavé numérique) de l’écran concerné.



Pour la description de ces écrans référez vous à la Notice Utilisateur.

6.1 Raz du contexte d'exécution.

Cette fonction est accessible par la touche RAZ dans la partie supérieure de l'écran de base, elle ne peut être activée qu'en marche manuelle et demande une seconde validation pour être effective.

Elle effectue:
- Une initialisation des fonctions d'affichage.
- La mise à zéro des ordres.
- La mise à zéro des ordres outil.

Dans le cas d'une personnalisation de l'application par l'utilisateur elle exécute les fonctions définies dans la routine **Raz_Perso** du module **Perso.sys.**

7. ECRITURE DES PROGRAMMES ET TRAJECTOIRES

Le module PRG_MVT contient les programmes et les trajectoires du robot. Un “programme”, qui correspond à l’ensemble des tâches à réaliser pour un code programme, est constitué d’un enchaînement de trajectoires. Les trajectoires sont des routines de type "PROC".

Toutes les trajectoires et toutes les données de mouvement doivent a priori se trouver dans ce module PRG_MVT. La création d’un nouveau module ne peut se faire qu’avec l’accord du client final.

Le travail à faire par l’intégrateur est à effectuer dans le module "PRG_MVT".

Renseignement des programmes de travail.

Création et apprentissage des trajectoires des programmes de travail.

Apprentissage des trajectoires de service :

⇒ T_Lance,
⇒ T_Repli,
⇒ T_Serv1
⇒ T_Serv2
⇒ T_Serv3
⇒ T_Serv4

Tout ce travail peut se faire directement sur la baie sans passer par un PC.

Après chaque modification de trajectoire ou de programme, ne pas oublier de sauvegarder le module PRG_MVT.MOD sur une disquette de travail mais aussi sous la racine “ SITE ” dans le “ Flashdisk ” cf. § 4.2,Sauvegarde des modules PRG_MVT.MOD et MAINTENA.SYS.

7.1 Structure du module PRG_MVT.MOD

7.1.1 Routine « Main »

La routine “Main” du module PRG_MVT contient une boucle sans fin : attente d’un numéro de programme et appel de la routine “Programme”, qui contient pour chaque programme l’enchaînement des trajectoires à exécuter.

Le numéro du programme reçu prend en compte un ordre de priorités, tel que défini au paragraphe 8.1 Priorité des échanges codes et la gestion des demandes de service décrite au paragraphe 7.5 Programmes de service .

Module PRG_MVT	Commentaires

`!*****************************`
`!*  ATTENTION !!!!!  *`
`!*  ROUTINE A NE PAS MODIFIER *`
`!*  ET A NE PAS SUPPRIMER !!  *`
`!*****************************`
`PROC MAIN()`
`  InitProg ;`
`  WHILE TRUE DO`
`    Att_Num_Prog;`
`    Programme(Num_Prog);`
`  ENDWHILE`
`ENDPROC`
	NE PAS MODIFIER
	
	Initialisations lors d'un spécial '3'
	Boucle principale sans fin du programme PRG_MVT
	Attente n° programme
	Tâches pour ce programme robot

7.1.2 Routine « Programme »

Cette routine est appelée par la routine « Main » pour chaque nouveau travail à effectuer. Elle a la forme d'un débranchement en fonction du numéro de programme reçu. Les enchaînements des trajectoires constituant les programmes de travail seront renseignés dans cette routine.

Pour des détails sur la personnalisation de cette routine, voir 7.3 Programmes de travail.

Module PRG_MVT (fourniture ABB)	Commentaires
`PROC Programme(` `  num prog)`	
`  TEST prog`	
`  CASE 1:`	Programme de travail n°1
`    Aff_Prog prog,"Prog 1";`	Affichage du nom du programme dans les écrans utilisateurs
`    ! pulse sur la sortie Top cycle`	
`    PulseDO\PLength:=1,TC;`	Top cycle pour l'automate
`    autorise_traj:=TRUE;`	
`    Traj1;`	Lancement de la trajectoire Traj1.
`  CASE 2:`	
`    Aff_Prog prog,"Prog 2";`	Programme de travail n°2
`    ! pulse sur la sortie Top cycle`	
`    PulseDO\PLength:=1,TC;`	
`    autorise_traj:=TRUE;`	
`    Traj2;`	
`  CASE 3:`	
`    Aff_Prog prog,"Prog 3";`	Programme de travail n°3
`    ! pulse sur la sortie Top cycle`	
`    PulseDO\PLength:=1,TC;`	
`    autorise_traj:=TRUE;`	
`    Traj1;`	Une trajectoire peut être partagée par plusieurs programmes
`    autorise_traj:=TRUE;`	
`    Traj2;`	
`  DEFAULT:`	
`    Dem_Services prog;`	Traitement des demandes de service
`  ENDTEST`	
`ENDPROC`	

7.1.3 Routine « Dem_Services »

Cette routine est appelée par la routine « Programme » dans le cas où le numéro de programme reçu ne correspond pas à un programme de travail. Il s'agit alors d'un code de demande de service (au sens large). Les travaux à exécuter pour les demandes de service 1 à 4 sont à renseigner dans cette routine. Le traitement des demandes de repli et de l'exécution de la trajectoire de lancement sont traités par la routine « Services », non modifiable.

Pour des détails sur la personnalisation de cette routine, voir 7.5 Programmes de service.

```
PROC Dem_Services(
    num sprog)
    TEST sprog
    CASE CP_DemServ1:
      Aff_Prog sprog,"Service 1";
      autorise_traj:=TRUE;
      T_Serv1;
      Etat_Serv1:=SV_PAS_DEMANDE;
    CASE CP_DemServ2:
      Aff_Prog sprog,"Service 2";
      autorise_traj:=TRUE;
      T_Serv2;
      Etat_Serv2:=SV_PAS_DEMANDE;
    CASE CP_DemServ3:
      Aff_Prog sprog,"Service 3";
      autorise_traj:=TRUE;
      T_Serv3;
      Etat_Serv3:=SV_PAS_DEMANDE;
     CASE CP_DemServ4:
      Aff_Prog sprog,"Service 4";
      autorise_traj:=TRUE;
      T_Serv4;
      Etat_Serv4:=SV_PAS_DEMANDE;
    DEFAULT:
      Services sprog;
    ENDTEST
  ENDPROC
```

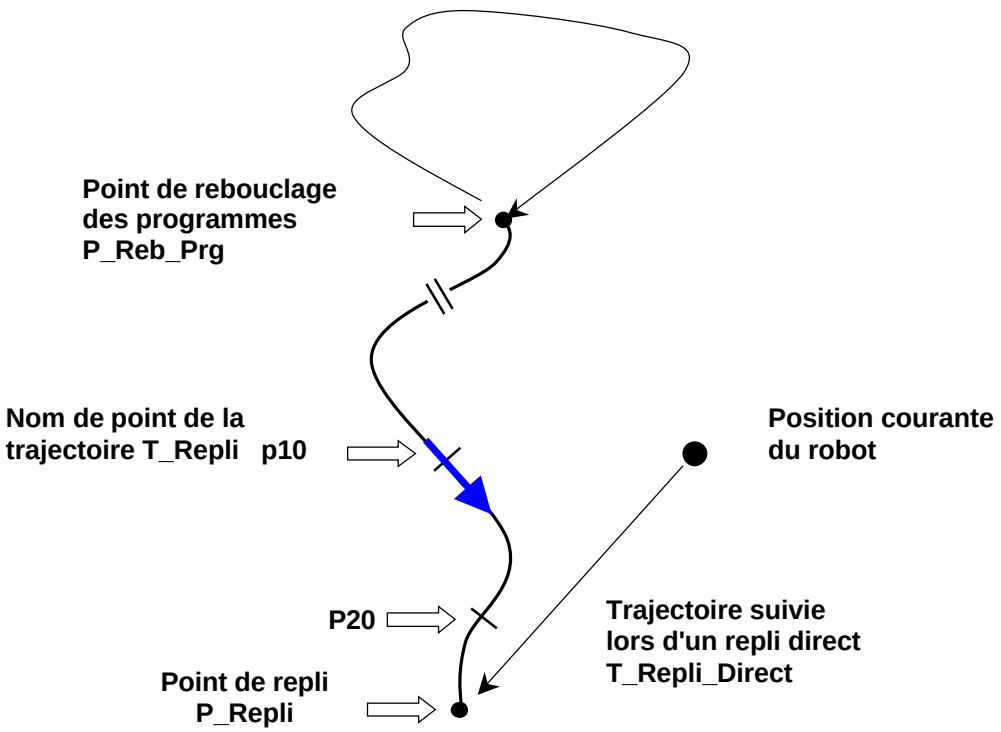
Affichage du nom du programme dans les écrans utilisateurs

Lancement de la trajectoire T_Serv1.
Information « demande de service effectuée ».

NE PAS MODIFIER

7.2 Trajectoires de repli et de lancement

7.2.1 Définition des trajectoires de repli



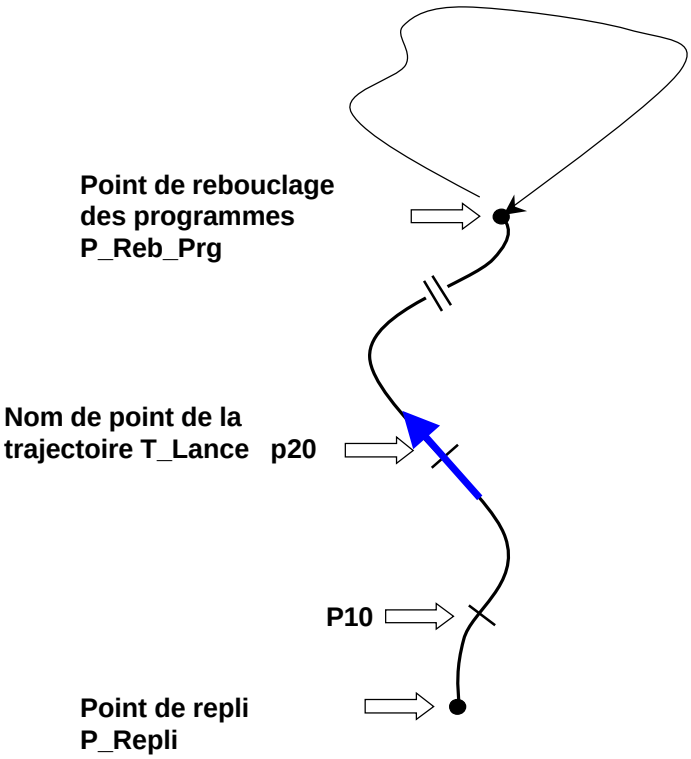
- La trajectoire T_Repli part de la position de rebouclage pour aller au point de repli.
- La trajectoire T_Repli_Direct ne comporte qu'un seul point : le point de repli.

- Ce point correspond à une position physique du robot permettant une maintenance aisée. Il est le plus éloigné possible des parties dangereuses.
- Le nom du point est défini : "p_repli"
- Le mouvement appris est articulaire : "MoveJ"
- Le référentiel du point est "wobj0"
- La nature du point est "fine" : Z_arret (point d'arrêt)
- Le repli direct ne peut être lancé qu'en mode manuel.
- Le test automatique robot au point de repli (p_repli) impose un redémarrage du programme (Spécial 3) quand cette position est modifiée. Aucune action ne doit être programmée sur le dernier point.

Exemple :

```
MoveJ p_repli,V_moyen,Z_arret,Tool_Defaut \Wobj:=wobj0;
```

7.2.2 Définition de la trajectoire de lancement



- La trajectoire T_Lance part de la position de repli pour aller au point de rebouclage des programmes. C'est à partir de ce point que débuteront et finiront tous les autres programmes. Il est donc positionné de façon à optimiser les temps et les déplacements pour l'enchaînement des autres programmes.
- Le nom du point est défini : "p_reb_prg"
- Le mouvement appris est articulaire : "MoveJ"
- Le référentiel du point est "wobj0"
- Le référentiel outil est "Tool_defaut"
- La nature du point est "fine" : Z_arret (point d'arrêt)
- Le test automatique robot au point de rebouclage (p_reb_prg) impose un redémarrage du programme (Spécial 3) quand cette position est modifiée.
-
- Les actions (ordres, événements, ouverture/fermeture préhenseur, etc..) sont à programmer à partir du deuxième point en fonction des besoins du site.

Exemple :

```
MoveJ p_reb_prg,V_rapide,Z_arret,Tool_defaut\Wobj:=wobj0;
```

7.2.3 Apprentissage des points de repli et de rebouclage

Les trajectoires **T_Lance**, **T_Repli** et **T_Repli_Direct** sont pré-renseignées dans la fourniture de base ABB. Elles mettent en œuvre deux points particuliers, le point de repli et le point de rebouclage.
Ces deux points particuliers sont pré-déclarés dans l'application de base mais leurs coordonnées par défaut ne correspondent pas aux besoins du site. Comme les trajectoires **T_Repli** ou **T_Repli_Direct**, puis **T_Lance** sont appelées dès le lancement du programme robot, on commencera donc par renseigner ces deux points particuliers.

Ces points **doivent impérativement** être appris avec le référentiel objet de base **wobj0** et l'outil **Tool_defaut**. Ils doivent également être éloignés d'au moins 20 mm.

L'outil **Tool_defaut** est une variable de type ***tooldata***, définie avec le CDO à 0 (sur le flasque) et avec la charge nominale de l'outil embarqué. Il aura fait l'objet d'une identification de charge particulière. Dans le cas de l'utilisation d'un changeur d'outil, il appartient à l'intégrateur de créer autant d'outils que nécessaire et de mettre à jour cette variable en fonction de l'outil effectivement porté.

Pour mémoriser ces positions, il faut procéder de la manière suivante :

- Mettre le robot dans sa position physique de repli en pilotage manuel après avoir sélectionné l'outil **Tool_defaut** et l'objet **wobj0**
- sélectionner GESTION DES PROGRAMMES par la touche 
- sélectionner **Voir** / **Routine [✓ 2]** puis 
- placer le curseur sur " **T_Lance** " - puis 
- positionner le curseur sur la ligne "**MoveJ p_repli, ...**"
- valider **Modpos** en bas de l'écran
- valider " **OK** "
- mettre le robot sur sa position physique du point de rebouclage des programmes
- positionner le curseur sur la ligne "**MoveJ p_reb_prg** ..."
- valider **Modpos** en bas de l'écran
- valider " **OK** "

Après avoir modifié une de ces positions, il est nécessaire de redémarrer le programme à son point de départ (**Spécial [✓ 3]**)

7.3 Programmes de travail

Pour chaque programme de travail il convient de renseigner dans la routine « Programme », en fonction du numéro de programme qui correspond au code envoyé par l'automate, l'enchaînement des trajectoires à exécuter. L'intégrateur doit également renseigner le nom du programme, tel qu'il apparaîtra dans l'écran de base de l'application.

Une trajectoire peut être commune à plusieurs programmes. Il est possible de choisir, au sein d'un même programme, entre deux trajectoires, au moyen de test sur des événements (voir exemple ci-dessous).

ATTENTION :

Toute trajectoire devra comporter l'appel de la procédure **« Anti_Reboucle** » et tout appel de trajectoire dans la routine « Programme » doit être précédé de la ligne **" Autorise_Traj :=TRUE ; "**

Ces deux instructions permettent de gérer l'Anti-rebouclage en cas de Spécial 4.

Si a priori rien n'empêche l'utilisation d'instructions logiques (telles que IF, FOR, WHILE) dans la routine « Programme », cette utilisation est fortement déconseillée. En cas de déroutement (choix entre deux trajectoires) on préfèrera l'aiguillage via le test d'événements.

Voir 9.2.4 Evénements utilisés en aiguillage.

Exemple de personnalisation (les instructions modifiées ou insérées sont sur un fond coloré) :

Module PRG_MVT	Commentaires

`!*********************************`
`! A personnaliser par intégrateur`
`!*********************************`
`PROC Programme(Num_Prog)`
	Détail de la routine PROGRAMME

`TEST prog`	

`  CASE 1`
`    Aff_Prog  prog," AILE D74 ";`
`    ! pulse sur la sortie Top cycle`
`    Pulse DO\Plength:=1,TC;`
`    Autorise_traj:=true;`
	Programme de travail n°1
	Affichage du nom du programme dans les écrans utilisateurs
	Top cycle pour l'automate

`    TpriseB;`
`    Autorise_traj :=true ;`
`    TColle;`
	NB Une trajectoire peut être utilisée dans plusieurs programmes.

`    Autorise_traj :=true ;` `    TdeposeB;`	

`  CASE 2`
`    Aff_Prog  prog," AILE X9+ ";`
`    ! pulse sur la sortie Top cycle`
`    Pulse DO\Plength:=1,TC;`
`    Autorise_traj:=true;`
	Programme de travail n°2

`    TpriseB;`
`    Autorise_traj :=true ;`
`    TcolleB;`	

`    ! attente aiguillage`
`    Flag_Aff_mess:=TRUE;`
`    Tab_Mess_Aff{MESS_ATT_AIGUILL}:=TRUE;`
	Exemple d'aiguillage (par exemple dépose sur convoyeur ou sur table)
	Affichage « Attente aiguillage »

`    WaitUntil Dinput (EV3) <>Dinput (EV4);` `    Tab_Mess_Aff{MESS_ATT_AIGUILL}:=FALSE;`	Attente de l' événement 3 ou 4

`    IF Dinput (EV3)=1 THEN`
`      Autorise_traj :=true ;`
`      TdeposeC;`
	Effacement message
	Dépose sur convoyeur

`    ELSE`
`      Autorise_traj :=true ;`
`      TDeposeT;`
`    ENDIF`
	Dépose sur table

`DEFAULT:`
`    Dem_Services prog;`
`ENDTEST`
`ENDPROC`	

7.4 Trajectoires de travail

Convention : Le nom des trajectoires de travail doit commencer par la lettre T et ne doit pas dépasser 8 caractères.

Il est recommandé de nommer les variables de manière homogène, notamment en ce qui concerne les référentiels objet et outil. Une suggestion est d'utiliser Obj_xxx pour les référentiels objet, et Out_xxx pour les référentiels outil.

Pour la description des actions sur trajectoire utilisables pour tout métier, voir le chapitre 9, ACTIONS DE BASE PROGRAMMABLES SUR TRAJECTOIRES. Pour les éventuelles actions spécifiques à un métier se référer au guide d'intégration correspondant à ce métier.

Voir également la documentation ABB « Vue d'ensemble de Rapid »

7.4.1 Contraintes d'apprentissage

En principe, la première trajectoire d'un programme doit partir du point de rebouclage, et la dernière trajectoire d'un programme se terminer sur ce point.


Sur ce point, voir 5.2.3.3, Lecture des codes cycle sur le point de rebouclage

Chaque routine de trajectoire devra débuter par les instructions " **Aff_Traj " T_xxxxx"** " qui permet d'afficher le nom de la trajectoire en cours sur l'écran utilisateur et " **Anti_Reboucle** " (qui permet de gérer l'anti-Rebouclage en cas de " spécial 4 ")

A la suite de ces instructions, la routine de trajectoire contiendra les instructions de mouvement et les actions de la trajectoire.

7.4.1.1 Utilisation des référentiels objet

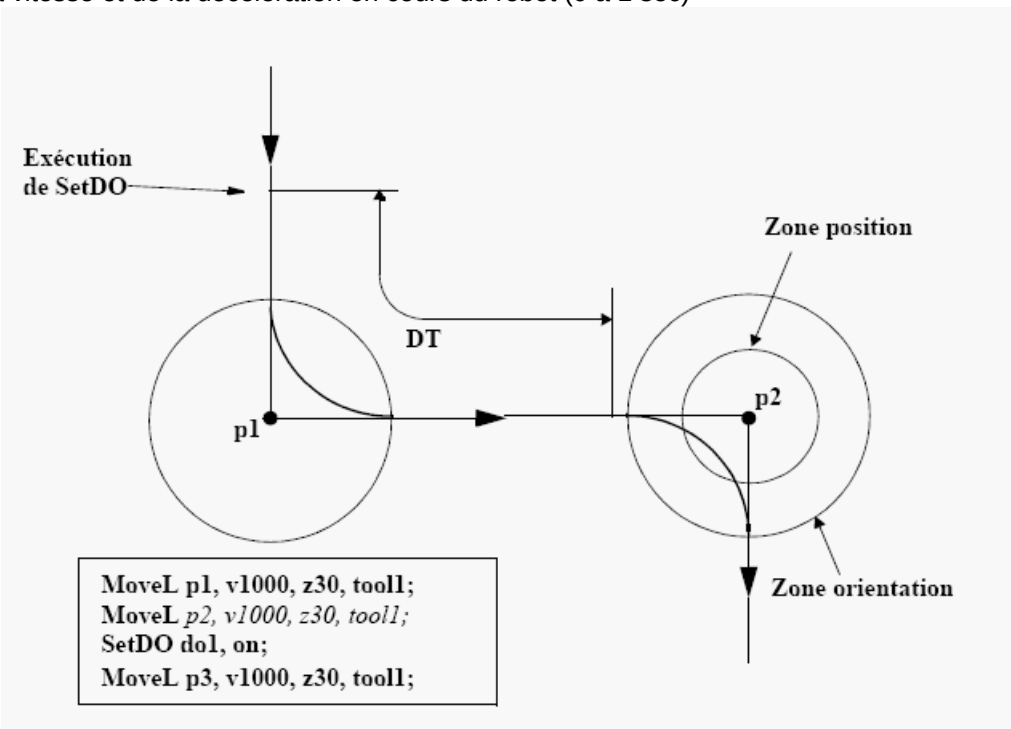
Il est impératif que les trajectoires liées à une pièce de travail soient apprises dans un référentiel objet qui corresponde aux éléments mécaniques qui supportent cette pièce (montage, convoyeur, machine outil, etc.). Ainsi, en cas de modification de l'implantation de ce support, il suffira de redéfinir ce référentiel objet sans avoir à retoucher les trajectoires.

L'utilisation de wobj0 est réservée aux positions de repli et de rebouclage. Elle est néanmoins également possible pour les points non liés à une pièce (points de service, par exemple).

7.4.1.2 Notion d'anticipation

Le système de commande planifie à l'avance les mouvements du robot de façon à pouvoir générer à temps les points de passage. La programmation de ceux-ci permet donc d'anticiper l'exécution d'instructions logiques.

Le temps DT avant lequel l'instruction est exécutée dépend :
- du temps de calcul du prochain mouvement (environ 0,1 sec)
- de la vitesse et de la décélération en cours du robot (0 à 1 sec)



Dans le cas d'une succession de point de passage rapprochés, l'instruction logique peut ainsi être exécutée bien avant que le point ne soit atteint.

Lorsqu'on souhaite qu'une information soit envoyée précisément par rapport à une position physique du robot, par exemple une sécurité arrière ou une sortie de zone d'interférence il faut utiliser une autre méthode :

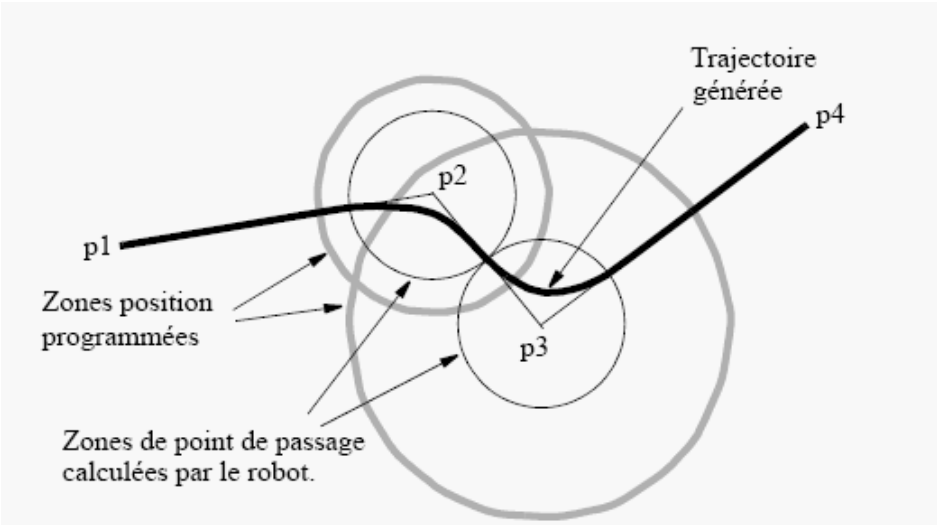
- S'il est possible d'arrêter le robot, l'instruction ORDRE ou ORD_OUTIL sera écrite derrière une instruction de mouvement programmée en "fine". C'est à dire avec arrêt.
- Sinon, il faudra utiliser l'instruction MOVE_ORDRE ou MOVE_ ORD_OUTIL qui permettent de définir une action en fonction d'une position précise du robot sur sa trajectoire.

Nota : Il est aussi possible, a contrario, de différer l'émission de l'ordre par l'emploi de l'argument \Delai dans l'instruction ORDRE ou ORD_OUTIL

7.4.1.3 Notions de lissage de trajectoire et de vitesse

Si des positions sont très rapprochées, il est possible que le robot calcule des zones plus petites que les zones programmées. Afin de réaliser un trajet correct et d'atteindre la vitesse optimum à tout moment, le robot recalcule et réduit si nécessaire la taille des zones de chacun des deux points.

Il faut donc éviter de programmer des zones importantes sur des points proches pour ne pas avoir de recouvrement de zones.



Un point de passage peut être transformé en point d'arrêt dans le cas où le mouvement suivant n'a pu être calculé à temps. Les causes les plus courantes sont :

- un nombre d'instructions important avec de longs programmes à exécuter entre deux mouvements
- les points sont très rapprochés et la vitesse élevée

Règles générales:

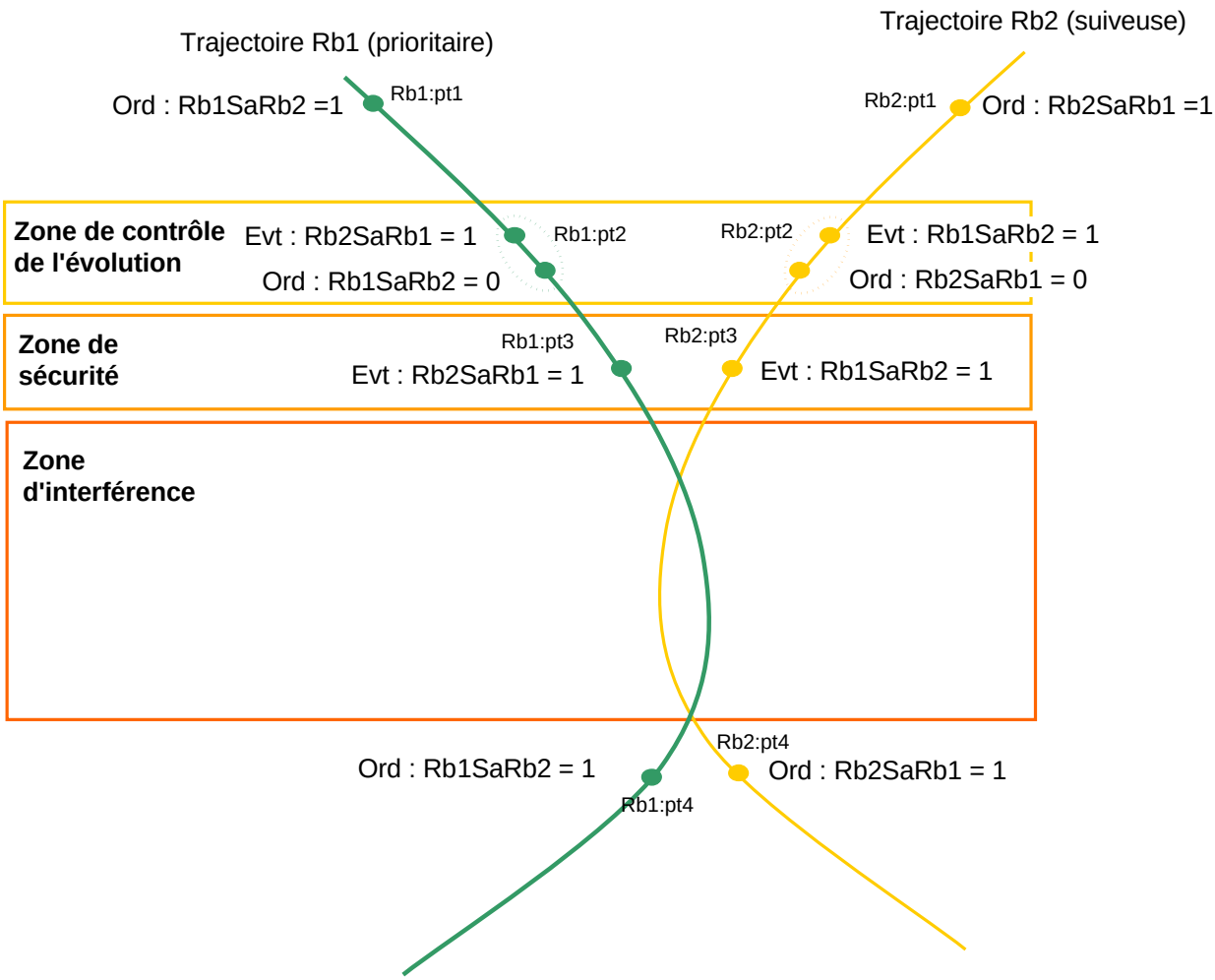
S zones < distance entre points.

Pas d'utilisation systématique de « vmax »

7.4.1.4 En résumé

On ne doit pas …	Mais il faut …
Définir systématiquement la vitesse à "vmax"	Adapter la vitesse au mouvement à réaliser
Définir systématiquement la zone à "z200"	Choisir des zones sans recouvrement
Utiliser sans raison l'arg **\conc** du Move	L'utiliser avec précaution et le moins possible
Mettre des instructions bloquantes dans une Trap	Positionner des flags ou exécuter des instructions simples
Programmer des boucles vides	Utiliser l'instruction WaitTime
Utiliser l'instruction Goto	Structurer ses programmes
Programmer une attente sans message	Créer un message dans Def_Site
Créer des routines sans traitement d'erreur	Traiter ErrNo dans Error
Utiliser l'instruction **ORDRE** sur point de passage	Utiliser **MOVE_ORDRE** ou traiter sur point d'arrêt
Utiliser l'instruction **ORD_OUTIL** sur point de passage	Utiliser **MOVE_ORD_OUTIL** ou traiter sur point d'arrêt
Utiliser l'instruction **SEQ** sur point de passage	Traiter uniquement sur point d'arrêt
Utiliser l'instruction **CPP** sur point de passage	Traiter uniquement sur point d'arrêt
Utiliser l'instruction **EVENT** sur point de passage	Utiliser **MOVE_EVENT** ou traiter sur point d'arrêt
Utiliser l'instruction **EVT_OUTIL** sur point de passage	Traiter uniquement sur point d'arrêt
Déclarer des données dans le module **User**	Les déclarer dans un des modules utilisateurs
Utiliser systématiquement **wobj0**	Créer un référentiel objet chaque fois que le point de travail est situé sur une pièce liée à l'atelier par un montage mécanique.

7.4.2 Gestion des interférences robot



Les trajectoires des deux robots se contrôlent mutuellement au moyen de la paire ordre/événement Rb1SaRb2/Rb2SaRb1. La trajectoire qui passe son ordre à 0 la première bloque l'autre robot soit en zone de contrôle ou si les deux robots étaient presque synchrones en zone de sécurité. Le robot arrêté ne repartira qu'après que le robot en zone d'interférence soit sorti et ai réactivé l'ordre.

7.5 Programmes de service

Les programmes de repli et de lancement sont gérés en interne et ne sont visibles qu'à travers les trajectoires **T_Lance**, **T_Repli** et **T_Repli_Direct.**

Les programmes de service suivants sont pré-déclarés et gérés par l'application, mais modifiables :

- Programme de service 1
- Programme de service 2
- Programme de service 3
- Programme de service 4

7.5.1 Principes de la gestion des demandes de service


Les principes exposés ici sont valables pour tout type d'application, manutention et autres métiers. Cependant, si l'application tronc commun manutention n'utilise pas dans sa fourniture de base de demande de service dédiée, certains programmes de service peuvent par contre être réservés par les applications métier. Se reporter alors au guide d'intégration spécifique de ces applications.


Il existe 2 façons d'appeler ces programmes de services :

- L'API envoie le code correspondant au programme de service a exécuté (comme pour un programme de travail)

- Une demande de service a été mémorisée dans une des variables « **Etat_Serv1** » (demande de service 1) à « **Etat_Serv4** » (demande de service 4). Le robot exécutera le programme de service correspondant avant de prendre en compte un nouveau code programme.

Si l'application particulière développée par l'intégrateur nécessite la gestion de programmes de service, celui-ci devra utiliser ces variables Etat_ServX et aura tout intérêt à lui affecter les valeurs prédéfinies suivantes :

- SV_PAS_DEMANDE : pas de demande de service ou demande de service correctement acquittée
- SV_ABANDONNE : problème lors de la trajectoire de service; la demande de service n'a pas été correctement terminée.
- SV_DDE_INTERNE : demande de service provenant d'une demande interne à l'application, suite par exemple au dépassement d'un seuil du compteur de cycle
- SV_DDE_EXTERNE : demande externe, par exemple par BP.
- SV_EN_COURS : demande de service en cours (trajectoire de service en cours)
- SV_EFFECTIF : fonction de service réalisée mais programme de service non terminé

Une demande de service positionnée par l'intermédiaire d'une variable Etat_ServX restera active tant que le programme de service correspondant n'aura pas remis cette variable à l'état SV_PAS_DEMANDE.
Par défaut dans la fourniture ABB, cet acquittement est fait systématiquement dans la routine « Dem_Services » après l'appel de la trajectoire de service correspondante. Si l'intégrateur souhaite gérer des exceptions au cours de la trajectoire de service et forcer un nouvel appel de la routine de service en cas d'échec, il devra supprimer cet acquittement de la routine « Dem_Services » et le placer judicieusement dans la trajectoire de service.

Pour le système, toutes les valeurs autres que SV_PAS_DEMANDE sont traitées de la même manière. La différenciation de ces valeurs permet uniquement de faciliter la mise au point de l'application.

7.5.2 Renseignement des trajectoires de service

T_Lance : appelée par le programme Lance. Cette trajectoire va du point de repli vers le point de rebouclage des programmes. La trajectoire " ***T_Lance*** " est déjà définie dans l'application, il reste à insérer les points de passage entre **MoveJ p_repli;** et **MoveJ p_reb_prg;**

T_Repli : trajectoire appelée par le programme Repli. Cette trajectoire part du point de rebouclage des programmes et rejoint le point de repli selon une trajectoire apprise.

T_Repli_Direct : trajectoire appelée par le programme de repli direct. Cette trajectoire ne comporte que le point de destination (p_repli).

T_SERV1, T_SERV2, T_SERV3, T_SERV4 : si les demandes de service 1 à 4 sont utilisées, il suffit d'insérer les points et les actions dans les trajectoires correspondantes.

Le pas à pas ne permet pas d'entrer dans les routines de service. Pour effectuer des modifications, utiliser « Voir Routines » ou « Spécial 4 » si un test est nécessaire.

8. ECHANGES CODES

8.1 Priorité des échanges codes

Il existe plusieurs types de demandes de travail. Ces différents types de code programme sont hiérarchisés au niveau de la priorité de leur prise en compte.

Voici dans l'ordre de ces priorités (du PLUS prioritaire au moins prioritaire) :

Demande de repli
Les demandes de services.
Code interne (généré par la routine " Code_Interne ")
Les codes externes.

8.2 Principe du code interne

Le code retourné par la fonction Code_Interne est prioritaire sur tous les codes programmes externes présents. En fonction du nombre retourné par l'instruction " **Return x** ; " le travail défini dans la routine " **Programme** " au " **Case x** " sera exécuté.

Par défaut, la routine retourne " **0** " laissant ainsi la priorité aux codes externes.

En fonction de conditions spécifiques à l'installation, la routine retournera le code du programme à exécuter de façon prioritaire. Si ces conditions ne sont pas remplies, il est **IMPERATIF** de retourner la valeur « 0 »(comme dans l'exemple qui suit) afin de permettre la prise en compte des demandes de code externe.

Exemple d'utilisation:

```
FUNC num Code_Interne()
    ! Calculer le code a executer
    ! en fonction des conditions
    ! internes et retourner
    ! le code calcule
    !
    ! Ex: return XXX
    ! Si pas de code interne
    IF EV1=1 THEN
      RETURN 2;
    ELSE
      RETURN 0;
    ENDIF
  ENDFUNC
```

8.3 Principe des programmes de services.

Les demandes de service sont faites par bouton poussoir, par l'intermédiaire de la routine " Code_Interne ", ou par code externe. Elles sont gérées (mémorisation et priorité à l'exécution) dans la tache d'arrière plan.
La demande d'exécution d'un programme de service de priorité supérieure annule la demande d'exécution d'un programme de service de priorité inférieure.
Chaque exécution d'un programme de service s'effectue lorsque le robot est à son point de rebouclage.

Valeur des 4 trajectoires de service.

Programme de service 1
Programme de service 2
Programme de service 3
Programme de service 4

Code programme à retourner dans la routine “ Code-interne ”

```
CONST num CP_DemServ1 := 62;
CONST num CP_DemServ2 := 61;
CONST num CP_DemServ3 := 60;
CONST num CP_DemServ4 := 59;
```

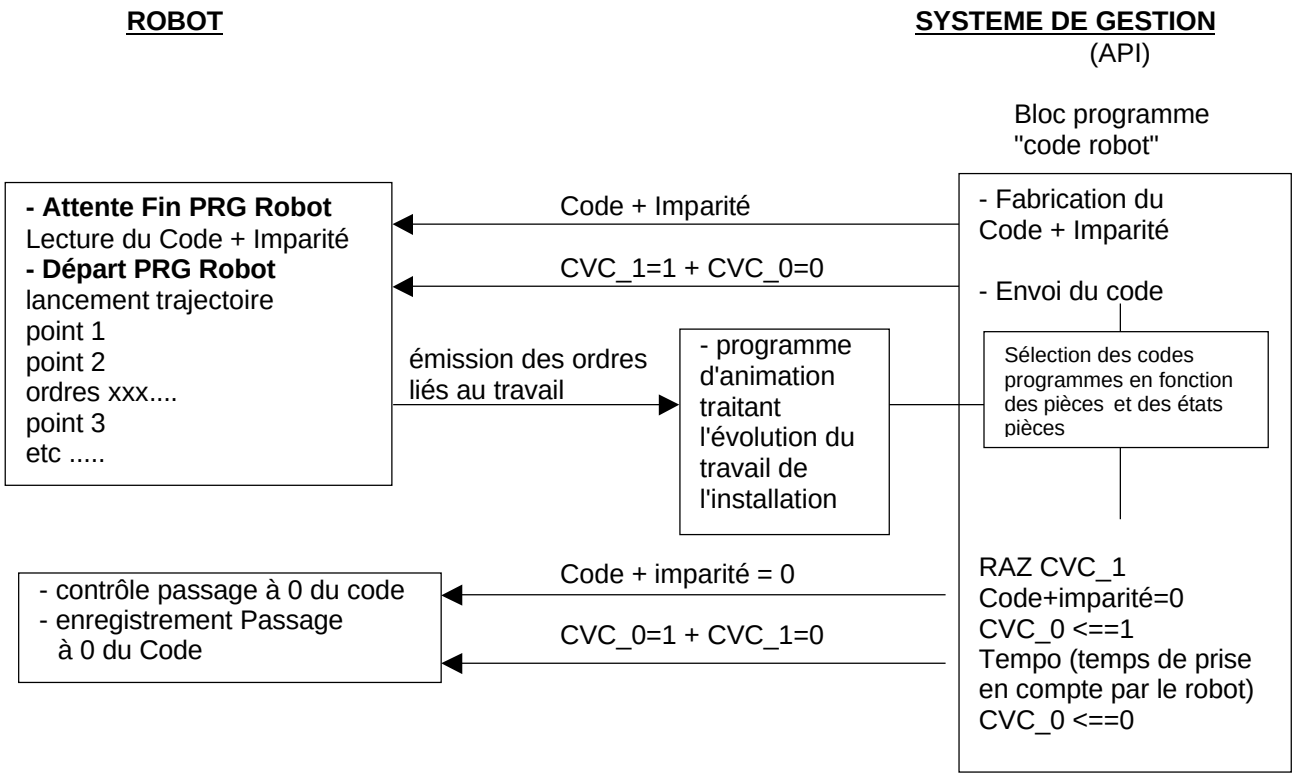
8.4 Principe des échanges

Le principe adopté pour les échanges de code programme doit permettre :

⇒ d'autoriser la lecture par le robot du code présent, d'éviter les erreurs de lecture dues à la non stabilisation des sorties code programme du système externe.
- C'est le rôle de l'information **"**Contrôle Validation Code à 1" (CVC_1)

⇒ de contrôler d'un programme à l'autre, par le passage à 0 de l'ensemble des informations liées au code, que l'ensemble de ces informations liées au code programme fonctionne.
- c'est le rôle de l'information **"**Contrôle Validation Code à 0" (CVC_0) qui autorise le robot à contrôler que l'ensemble des informations du code sont à 0.

⇒ Le rôle de l’entrée CPC0 est de contrôler l’imparité du code envoyé par l’automate (détection de fil coupé dans le cas d’une liaison parallèle).



9. ACTIONS DE BASE PROGRAMMABLES SUR TRAJECTOIRES

Voir également la Notice d'Utilisation NUPROMIA4.0.

9.1 Programmation des ordres

9.1.1 Action "ORDRE", sur point d'arrêt

Cette action permet dans une seule instruction la mise à 1 ou 0 des 16 sorties ordres "ORx".

Cette action doit être utilisée sur point d'arrêt.

Sur point de passage on lui préfèrera l'instruction ***Move_Ordre***. En effet, une programmation sur point de passage ne permet pas de maîtriser avec certitude le moment précis où les sorties sont réellement activées par rapport à la position du robot. Généralement, les sorties montent **avant** le passage sur le point. L'instruction ***Move_Ordre*** permet de remédier à cet asynchronisme.

La programmation des ordres à émettre se fait dans la fenêtre " **PROGRAMME** ".

Placer le curseur sur la ligne où l'on souhaite programmer des ordres, insérer l'instruction " **ORDRE** " de la liste commune 1. Documenter ensuite les arguments de la routine en plaçant le curseur sur l'instruction " **ORDRE** " et en validant " **ArgOpt** ".

Les arguments optionnels de la routine " **ORDRE** " à renseigner sont les suivants :

O1 (mise à 1 de la sortie OR1)
O2 (mise à 1 de la sortie OR2)
...
O16 (mise à 1 de la sortie OR16)

L'émission des ordres peut être retardé. Le paramètre optionnel **\Delai** permet de préciser un retard à l'émission des ordres (en secondes).

Pour insérer ou enlever un argument, il faut positionner le curseur sur le paramètre souhaité et valider " **Ajouter** " ou " **Suppr** ".

- L'action ORDRE positionne à 1 les ordres spécifiés et met les autres à 0. Il est donc nécessaire de répéter les ordres préalablement positionnés que l'on souhaite maintenir à 1.

- L'action ORDRE sans argument remet à zéro tous les ordres.

9.1.2 Action "MOVE_ORDRE" sur point de passage ou d'arrêt


Cette action intègre dans une même instruction un déplacement et la mise à 1 ou 0 synchronisée des 16 sorties ordres "ORx".

L'instruction " ***Move_Ordre*** " contient, en plus de la définition des ordres à émettre, le point visé, la zone, la vitesse, les données outil et le référentiel objet du déplacement.

⇒ Si l'instruction est programmée sur un point de passage les sorties Ordres définies dans l'instruction sont positionnées à 1 lors du passage du robot sur la position qui correspond à la projection du point visé sur la trajectoire.

Cette instruction permet donc d'activer les sorties Ordres de façon précise sur point de passage. La précision est de l'ordre de 5 millisecondes par rapport au point.

⇒ Si l'instruction est programmée sur un point d'arrêt, les sorties Ordres définies dans l'instruction sont positionnées à 1 lorsque le robot atteint le point appris.


La programmation des ordres à émettre se fait dans la fenêtre " **PROGRAMME** ".

Placer le curseur sur la ligne où l'on souhaite programmer des ordres, insérer l'instruction " **MOVE_ORDRE** " de la liste commune 1. Documenter ensuite les arguments de la routine en plaçant le curseur sur l'instruction " **MOVE_ORDRE** " et en validant " **ArgOpt** ".

Les arguments optionnels de la routine " **MOVE_ORDRE** " à renseigner sont les suivants :

\L Mouvement linéaire (articulaire par défaut)
O1 (mise à 1 de la sortie OR1)
O2 (mise à 1 de la sortie OR2)
...
O16 (mise à 1 de la sortie OR16)

Pour insérer ou enlever un argument, il faut positionner le curseur sur le paramètre souhaité et valider " **Ajouter** " ou " **Suppr** ".


- L'action MOVE_ORDRE positionne à 1 les ordres spécifiés et met les autres à 0. Il est donc nécessaire de répéter les ordres préalablement positionnés que l'on souhaite maintenir à 1.

- L'action MOVE_ORDRE sans argument remet à zéro tous les ordres.

9.1.3 Action "ORD_OUTIL"

Le comportement de cette action est strictement identique à l'action **ORDRE**, mais concerne les ordres Outil.

9.1.4 Action "MOVE_ORD_OUTIL" sur point de passage ou d'arrêt

Le comportement de cette action est strictement identique à l'action **MOVE_ORDRE**, mais concerne les ordres Outil.

9.2 Programmation des événements

9.2.1 Action "EVENT"

Cette action permet sur un point d'arrêt la programmation de l'attente de 1 à 16 entrées événements automate "EVx".

Seuls les événements déclarés dans l'action sont attendus (à 1), les autres sont ignorés. Si le contrôle dynamique des événements est actif, seuls les événements attendus seront ensuite contrôlés régulièrement.

Placer le curseur sur la ligne où l'on souhaite programmer des événements, insérer l'instruction "EVENT" de la liste commune 1. Documenter ensuite les arguments de la routine en validant "ArgOpt".

Les arguments de la routine "EVENT" à renseigner sont les suivants :

E1 (événement EV1 attendu)
E2 (événement EV2 attendu)
...
E16 (événement EV16 attendu)

- L'action EVENT contrôle à 1 les événements spécifiés et ignore les autres. Il est donc nécessaire de répéter les événements préalablement attendus que l'on souhaite continuer à contrôler.

- L'action EVENT sans argument remet à zéro tous les contrôles d'événements.

9.2.1.1 Messages associés.

Le message " **ATTENTE EVENEMENT** " apparaît sur l'écran principal tant que tous les événements attendus ne sont pas présents. Les événements attendus et non présents clignotent.

Si le contrôle dynamique est activé et que l'une de ces entrées n'est plus conforme à la programmation, le message " **PERTE EVENEMENT** " apparaît dans l'écran principal.

9.2.2 Action "MOVE_EVENT"

Cette action permet de programmer les événements sur point de passage.

Seuls les événements déclarés dans l'action sont attendus (à 1), les autres sont ignorés. Si le contrôle dynamique des événements est actif, seuls les événements attendus seront ensuite contrôlés régulièrement.

Placer le curseur sur la ligne où l'on souhaite programmer des événements, insérer l'instruction "MOVE_EVENT" de la liste commune 1. Documenter ensuite les arguments de la routine en validant "ArgOpt" pour les arguments optionnels.

En dehors des arguments classiques des instructions de mouvement (point visé, vitesse, zone, outil, objet) les arguments de la routine "EVENT" à renseigner sont les suivants :

\L Mouvement linéaire (articulaire par défaut)
E1 (événement EV1 attendu)
E2 (événement EV2 attendu)
...
E16 (événement EV16 attendu)

9.2.2.1 Messages associés.

Le message “ **ATTENTE EVENEMENT EN MOUVEMENT** ” apparaît sur l’écran principal tant que tous les événements attendus ne sont pas présents. Les événements attendus et non présents clignotent.

Si le contrôle dynamique est activé et que l'une de ces entrées ne sont plus conforme à la programmation, le message “ **PERTE EVENEMENT** ” apparaît dans l’écran principal.

9.2.3 Action "EVT_OUTIL"

Le comportement de cette action est strictement identique à l'action EVENT, mais concerne les événements Outil.

9.2.3.1 Messages associés.

Le message “ **ATTENTE EVENEMENT OUTIL** ” apparaît dans l’écran principal tant que tous les événements attendus ne sont pas présents. Les événements Outil attendus et non présents clignotent.

Si le contrôle dynamique est activé et que l'une de ces entrées n'est plus conforme à la programmation, le message “ **PERTE EVENEMENT OUTIL** ” apparaît dans l’écran principal.

9.2.4 Evénements utilisés en aiguillage.

Cette fonction permet de dérouter le robot vers une trajectoire en cours de cycle. Cette programmation n'est pas utilisée sur trajectoire mais au cours du cycle dans l'enchaînement des trajectoires.

Les événements sont inter-verrouillés (voir exemple). Le message “ **ATTENTE AIGUILLAGE** ” apparaît dans l’écran principal tant que les événements testés ont une valeur identique.

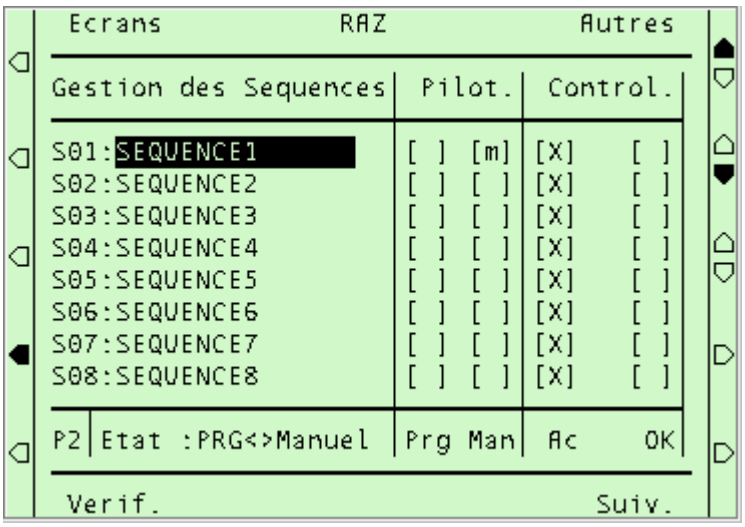
Exemple : choix de la trajectoire 1 ou 2 pour le programme 4

```
CASE 4
  Tab_Mess_Aff{MESS_ATT_AIGUILL}:=TRUE;
  WaitUntil DInput(EV3)<>DInput(EV4);
  Tab_Mess_Aff{MESS_ATT_AIGUILL}:=FALSE;
  IF DInput(EV3)=1 THEN
    autorise_traj:=TRUE;
    Traj1;
  ELSE
    autorise_traj:=TRUE;
    Traj2;
  ENDIF
```

9.3 Action "SEQ"

Cette action permet de programmer l'activation ou la désactivation de 1 à 16 actionneurs, d'attendre et de contrôler la bonne exécution de ces pilotages, et de lancer le contrôle dynamique des états attendus. Cette action est typiquement utilisée pour le pilotage des préhenseurs, mais elle permet également la gestion de tout organe mécanique piloté et contrôlé par des entrées/sorties tout ou rien (bridages, abattants rôdeuse...).

L'état des séquences "attendu" et "présent" est visible à tout moment dans l'écran dédié.



La programmation des séquences en apprentissage se fait dans la fenêtre " **PROGRAMME** ". Il faut placer le curseur sur la ligne où l'on veut programmer une action séquence et insérer l'instruction " **SEQ** " de la liste commune 1. Documenter ensuite les arguments de la routine en validant " **ArgOpt** " en bas de l'écran.

La sélection des pilotages des séquences se fait à l'aide des arguments suivants :

S1_A / S1_D (piloter la séquence 1 à l'état **A**ctivé ou **D**ésactivé).
S2_A / S2_D (piloter la séquence 2 à l'état **A**ctivé ou **D**ésactivé).
...
S16_A / S16_D (piloter la séquence 16 à l'état **A**ctivé ou **D**ésactivé 8).

Pour insérer ou enlever un argument, il faut positionner le curseur sur le paramètre souhaité et valider " **Ajouter** " ou " **Suppr** ".

- **Les séquences spécifiées sont activées ou désactivées, mais les autres séquences restent dans leur état précédemment programmé.**
- **L'utilisation de l'action SEQ sans argument optionnel n'a donc aucun effet**.

- Toute séquence doit être soit activée, soit désactivée. Par conséquent, pour éviter que certaines séquences soient dans un état inconnu, et donc non surveillées, il est conseillé de les activer toutes au moins une fois, par exemple dans la trajectoire " **T_Lance** ".

9.3.1 Programmation sur point d'arrêt :

La programmation classique des séquences se fait sur point d'arrêt. Les séquences sont alors pilotées et contrôlées sur le point.

Mais sur point d'arrêt on peut aussi :
contrôler une séquence sans la piloter ; utiliser l'argument optionnel **\SANS_ACT**,
ou
piloter une séquence sans la contrôler ; utiliser l'argument optionnel **\SANS_CTL**.

dans ce cas d'utilisation le contrôle dynamique doit être désactivé sinon le robot sera arrêté.

9.3.2 Programmation sur point de passage :

Sur point de passage on peut seulement piloter une séquence sans la contrôler ; utiliser l'argument optionnel **\SANS_CTL**.

dans ce cas d'utilisation le contrôle dynamique doit être désactivé sinon le robot sera arrêté.

9.3.3 Messages associés.

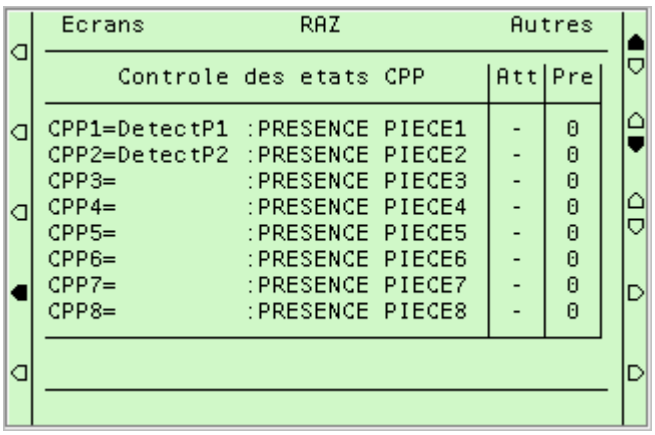
En cas d'attente, le message " **ATTENTE SEQUENCE n**" apparaît dans l'écran principal. On peut visualiser quelles séquences sont manquantes en validant l'écran " **SEQ** ":

Si le contrôle dynamique est activé et que l'une des séquences contrôlées n'est plus conforme à la programmation, le message " **PERTE SEQUENCE** " apparaît dans l'écran principal.

9.4 Action " CPP "

L'action CPP permet d'attendre les entrées présence pièce " CPPx ", de programmer leur état attendu et de lancer leur contrôle dynamique.

L'état des CPP "attendu" et "présent" est visible à tout moment dans l'écran dédié.



La programmation des états présences pièces se fait dans la fenêtre "PROGRAMME".
Il faut placer le curseur sur la ligne où on souhaite modifier l'état des présences pièces attendues et insérer l'instruction " CPP " de la liste commune 1. Documenter ensuite les arguments de la routine en plaçant le curseur sur l'instruction " CPP " et en validant " **ArgOpt** ".

Les arguments de la routine " CPP " à renseigner sont les suivants :

CPP1_1 / CPP1_0 (attend CPP1 à "1" ou à "0")
CPP2_1 / CPP2_0 (attend CPP2 à "1" ou à "0")
CPP3_1 / CPP3_0 (attend CPP3 à "1" ou à "0")
CPP4_1 / CPP4_0 (attend CPP4 à "1" ou à "0")
CPP5_1 / CPP5_0 (attend CPP5 à "1" ou à "0")
CPP6_1 / CPP6_0 (attend CPP6 à "1" ou à "0")
CPP7_1 / CPP7_0 (attend CPP7 à "1" ou à "0")
CPP8_1 / CPP8_0 (attend CPP8 à "1" ou à "0")

Pour insérer ou enlever un argument, il faut positionner le curseur sur le paramètre souhaité et valider " **Ajouter** " ou " **Suppr** ".

- Seules les entrées présence pièce spécifiées dans l'action sont attendues et contrôlées. Les autres sont ignorées. Il est donc nécessaire de répéter les CPP préalablement attendus que l'on souhaite continuer à contrôler.

- L'action CPP sans argument remet à zéro tous les contrôles de CPP (état indifférent).

- Cette programmation est à faire sur point d'arrêt. Elle est possible sur point de passage, uniquement si tous les états attendus sont indifférents (action CPP sans aucun argument).

9.4.1 Messages associés.

Quand on exécute l'action CPP le message " **ATTENTE PRESENCE PIECE** " apparaît dans l'écran principal tant que les entrées dédiées ne sont pas bien positionnées.

Si le contrôle dynamique est activé et que l'une de ces entrées n'est plus conforme à la programmation, le message " **PERTE PRESENCE PIECE** " apparaît dans l'écran principal.

9.5 Action "CTRL_DYN"

9.5.1 Utilisation

Le contrôle dynamique permet d'arrêter le robot en cas de perte de l'une des informations surveillées, et de le redémarrer automatiquement dès que la condition revient. Le programme robot n'est pas arrêté, seuls les mouvements sont "suspendus".
L'arrêt du robot se fait progressivement, et celui ci reste sur la trajectoire. Les asservissements ne sont pas coupés.

Les surveillances "dynamiques" sont:

- Les présences Pièces.
- Les événements automate.
- Les événements outils.
- Les séquences.

9.5.2 Mise en œuvre

La programmation des contrôles dynamiques se fait dans la fenêtre "PROGRAMME".
Il faut placer le curseur sur la ligne où on souhaite inhiber ou remettre le contrôle dynamique, insérer l'instruction " **CTRL_DYN** " de la liste commune 1. Documenter ensuite les arguments de la routine en plaçant le curseur sur l'instruction " **CTRL_DYN** " et en validant " **ArgOpt** ".

Les arguments de la routine " **CTRL_DYN** " à renseigner sont les suivants:

\OUI
\NON
\SEQ_OUI
\CPP_OUI
\EV_OUI
\EVOUT_OUI

9.5.2.1 Signification des arguments optionnels

- **Argument \Oui**

Si l'argument OUI est présent, tous les contrôles dynamiques sont remis à l'état par défaut défini dans le module DEF_SITE.SYS :

⇒ Active_Ctl_Seq (TRUE/FALSE).
⇒ Active_Ctl_Cpp (TRUE/FALSE).
⇒ Active_Ctl_Ev (TRUE/FALSE).
⇒ Active_Ctl_Evo (TRUE/FALSE).

Les contrôles dynamiques sont également remis à ces valeurs par défaut en cas de :

⇒ Spécial 3.
⇒ Spécial 4.
⇒ Recyclage.
⇒ Au Repli.

➢ Argument \Non

Si l'argument NON est présent, tous les contrôles sont inhibés.

La programmation sans aucun argument est équivalente à CTRL_DYN \NON.

➢ Autres arguments

Si un ou plusieurs des arguments \SEQ_OUI, \CPP_OUI, \EV_OUI, \EVOUT_OUI sont sélectionnés, seuls ces contrôles dynamiques correspondants sont activés.

Exemples :

CTRL_DYN ;	⇒ Inhibe tous les contrôles
CTRL_DYN\NON ;	⇒ Idem.
CTRL_DYN\OUI ;	⇒ Restaure les états par défaut.
CTRL_DYN\CPP_OUI ;	⇒ Active le contrôle CPP et désactive tous les autres.
CTRL_DYN\CPP_OUI\EV_OUI ;	⇒ Active les contrôles CPP et Evénements et désactive les autres.

9.5.3 Messages associés.

Si un ou plusieurs contrôles sont inhibés (de manière permanente ou temporaire), un message d'information le signale dans l'écran de base de l'application.

Exemple : `[TCM22: CTRL DYN INHIBES:-Ev-Evo ]`

10. CONFIGURATION DU MODULE DE MAINTENANCE

Le module “MAINTENA.SYS” regroupe les différentes trajectoires de référence, de service et de contrôle du robot.

10.1 Liste des routines de ce module

Liste des routines principales devant être documentées

10.1.1.1 Routines de mouvement des axes

PROC Zm_mgv1()	Déplacement de l'axe 1
PROC Zm_mgv2()	Déplacement de l'axe 2
PROC Zm_mgv3()	Déplacement de l'axe 3
PROC Zm_mgv4()	Déplacement de l'axe 4
PROC Zm_mgv5()	Déplacement de l'axe 5
PROC Zm_mgv6()	Déplacement de l'axe 6
PROC Zm_endu()	Déplacement des 6 axes (trajectoire d'endurance)
PROC Zm_ext()	Déplacement de l'axe externe

Les trajectoires partent du point de Repli et finissent par ce même point.

10.1.1.2 Routine de contrôle de positions physiques

PROC Zm_vern()	Alignement du robot sur ses verniers (pour chaque axe)
PROC Zm_deout1()	Routine d'aide à la dépose de l'outil 1 (pince)
PROC Zm_deout2()	Routine d'aide à la dépose de l'outil 2 (préhenseur)
PROC Zm_cibro()	Alignement de la cible du robot sur la pointe au sol
PROC Zm_cibo1()	Alignement de l'outil 1 du robot sur la pointe au sol
PROC Zm_cibo2()	Alignement de l'outil 2 du robot sur la pointe au sol
PROC Zm_obj1()	Routine de contrôle du référentiel objet 1
PROC Zm_obj2()	Routine de contrôle du référentiel objet 2
PROC Zm_obj3()	Routine de contrôle du référentiel objet 3
PROC Zm_obj4()	Routine de contrôle du référentiel objet 4

Les trajectoires partent du point de Repli et finissent par ce même point.

10.1.1.3 Routine de maintenance

PROC Zm_grais()	Routine maintenance pour la lubrification du robot
PROC Zm_nett()	Routine maintenance pour le nettoyage du robot
PROC Zm_equi()	Routine maintenance pour l'échange des équilibreurs
PROC Zm_faisceau()	Routine de maintenance pour le remplacement des faisceaux

Les trajectoires partent du point de Repli et finissent par ce même point.

10.1.1.4 Routines d'aide à l'étalonnage du robot

PROC Zm_MoveToSync ()	Trajectoire du point de Repli vers le point d'init géométrique
PROC Zm_SynToFixPt()	Trajectoire d'approche de la pointe fixe
PROC Zm_FixPteToSy()	Trajectoire de retour vers la position de synchro

10.2 Mise en œuvre des routines de mouvement

Toutes les routines de mouvement doivent débuter et finir par le point de Repli.

Pour les trajectoires de mouvement d'un axe, utiliser la vitesse maximum et l'outil le plus important au niveau de la charge (Tool_Defaut).

Exemple pour la trajectoire de mouvement de l'axe 1

- Sélectionner la fenêtre programmation 
  Sélectionner le menu Spécial
  Choisir le menu "4 Pt prog -> Autre routine"
  Sélectionner la routine "Zm_mgv1"

Exemple de trajectoire non documentée :

```
    !---------------------------
    !ROUTINES DE MOUVEMENT ROBOT
    !---------------------------
    !Routine maintenance axe 1
    !-------------------------
    PROC Zm_mgv1()
      !attente validation pour mouvement axe 1
      !---------------------------------------
      TPReadFK N_MENU,"VALIDER OK POUR MVT ROBOT","","","","OK","ANNULER";
      TEST N_MENU
      CASE 5:
        GOTO FINI;
      ENDTEST
  AXE1:
      !Inserer la traj sous ce commentaire
      !---------------------------------

      IF CYCLE=1 THEN
        RETURN;
      ENDIF
      IF Dmd_repli=TRUE THEN
        GOTO R_AXE1;
      ENDIF
      GOTO AXE1;
  R_AXE1:
      T_Repli_Direct;
  FINI:
    ENDPROC
```

Insérer la trajectoire de mouvement sous ce point

10.3 Mise en œuvre de la routine "Etalon"

Programmation **des trajectoires**

- Sélectionner la fenêtre programmation
- Sélectionner le menu Spécial
- Choisir le menu "4 Pt prog -> Autre routine"

10.3.1 Trajectoire d'approche de la position d'init géométrique

La trajectoire du robot passe en premier lieu par le point d'init géométrique du robot, il est donc important de contrôler la trajectoire de allant de la position de Repli vers la position d'init géométrique (Zm_MoveToSync).

```
LOCAL PROC Zm_MoveToSync()
  MoveJ p_repli,v200,fine, Tool_Defaut;
  !inserer la trajectoire sous ce commentaire

  MoveAbsJ p_sync,v200,z10, Tool_Defaut;
  MoveAbsJ p_sync,v10,fine, Tool_Defaut;
ENDPROC
```

Insérer les points dans cette zone

10.3.2 Trajectoire d'approche de la pointe de mesure

- Sélectionner la routine "Zm_synToFixPt"
- Programmer la trajectoire d'approche de la pointe fixe.

Attention les deux derniers points d'approche cités ci-dessous ne sont pas programmés dans cette trajectoire, il s'agit de points de type "Jointtarget" modifiables uniquement par la saisie manuelle des valeurs.

- Noter les valeurs angulaires du point d'approche de la pointe (moins de 5 centimètres pour le point J_nearpte).

- Noter les valeurs du dernier point d'approche (J_Robpte1), celui-ci doit être au plus près de la pointe et aligné (alignement des pointes).

Exemple de valeurs pour ce dernier point
(Exemple :Axe 1 : 20.51° Axe 2 : 35° Axe 3 : 54° Axe 4 : -25° Axe 5 : 12° Axe 6 : -89°)

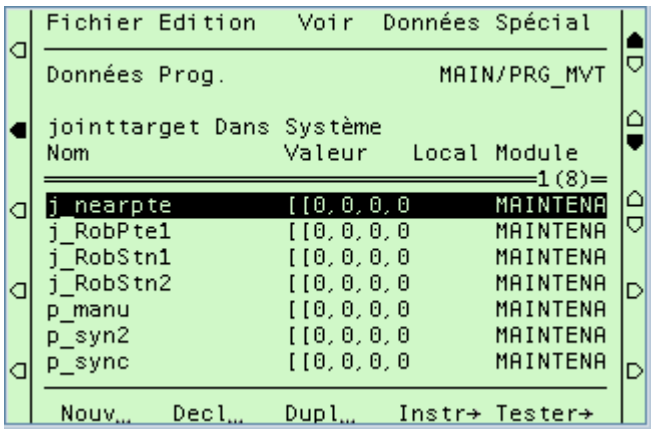
10.3.3 Trajectoire de retour vers le point de synchronisation

- Sélectionner la routine "Tm_FixPteToSy"
- Programmer la trajectoire de retour vers le point de synchronisation.

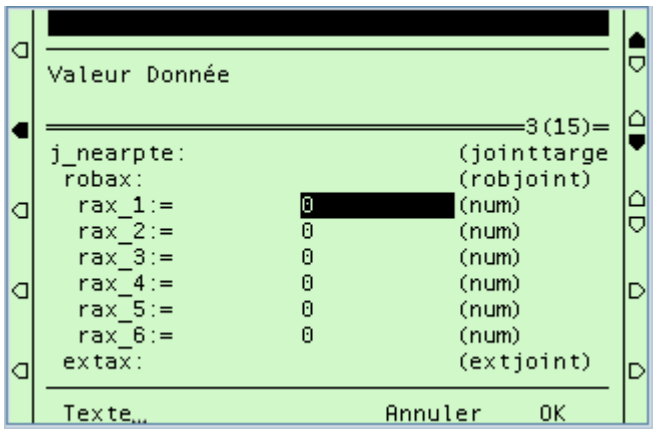
Cette trajectoire part du point d'alignement des cibles pour atteindre la position d'init géométrique (Position de synchronisation).

10.3.4 Intégration des valeurs angulaires pour les points identifiés "J_nearpte" et "J_Robpte1"

- Sélectionner la fenêtre programmation
- Sélectionner le menu Voir
- Sélectionner "Type de données" - puis
- Positionner le curseur sur le " Jointtarget" - puis
- Sélectionner le point "J_near" (point d'approche) – puis



- Saisir les valeurs angulaire de chaque axe. Valider la saisie par **OK**



Le module de base d'étalonnage est réalisé pour admettre trois positions de pointe outil (J_Robpte1, J_Robpte2, J_Robpte3), dans le cas d'une pointe amovible placée sur glissière donc sans risque de collision, les trois points peuvent admettre les mêmes valeurs. On utilise donc uniquement le point J_Robpte1.

10.4 Procédure pour personnaliser l'écran METIER

Dans le cas ou l'intégrateur désire utiliser le module maintenance pour rajouter des trajectoires et personnaliser le menu METIER, l'exemple ci dessous permet de visualiser les modifications nécessaires.

Exemple du menu MECA rajouté dans le menu METIER

```
    !---------------------
    !MENU DES METIERS
    !---------------------
METIERS:
    TPErase;
    TPWrite "CHOISIR LE METIER";
    TPReadFK N_MENU,"","MECAN","","","","ANNULER";
    TPErase;

    TEST N_MENU
    CASE 1:
      GOTO MECAN;
    CASE 5:
      GOTO DEPANNE1;
    ENDTEST
    GOTO DEPANNE1;

    !---------------------
    !MENU METIER MECANIQUE
    !---------------------
MECAN:
    TPErase;
    TPWrite "CONTROLE REFERENTIEL OBJET";
    TPReadFK N_MENU,"","OBJ1","OBJ2","OBJ3","OBJ4","ANNULER";
    TPErase;
    TEST N_MENU
    CASE 1:
      Zm_obj1;
    CASE 2:
      Zm_obj2;
    CASE 3:
      Zm_obj3;
    CASE 4:
      Zm_obj4;

    CASE 5:
      GOTO METIERS;
    ENDTEST
    GOTO DEPANNE1;
```

Menu MECA rajouté et test associé

Menu contrôle référentiel rajouté

Trajectoires rajoutées

Exemple de la routine rajoutée

```
!Routine contrôle référentiel objet 1
!-----------------------------------
PROC Zm_obj1()
  !attente validation pour mouvement axes
  !-----------------------------------
  TPReadFK N_MENU,"VALIDER OK POUR MVT ROBOT","","","","OK","ANNULER";
  TEST N_MENU
  CASE 5:
    GOTO FINI;
  ENDTEST
  !Inserer la traj sous ce commentaire
  !-----------------------------------
  MoveJ p_repli,v200,fine, Tool_Defaut;
  FINI:
ENDPROC
```

POUR L'UTILISATION DE CE MODULE REFEREZ VOUS A LA NOTICE UTILISATEUR Réf : NUPROMIA4.0

11. RECALAGE SUR TRAJECTOIRE / RECYCLAGE

 Définitions:

Suite à un arrêt de l'exécution du programme (même en apprentissage), deux modes de reprise se distinguent:

⇨ Le retour du robot sur sa trajectoire apprise (s'il a été déplacé) ou le simple redémarrage à partir de la position courante. Il s'agit du **RECALAGE**.

⇨ Les autres cas s'appellent **RECYCLAGE**. Ils se produisent notamment dans les circonstances suivantes:
- Déplacement du **curseur** sur une instruction différente de l'instruction courante (**Spécial [✔ 2]**).
- Déplacement du **robot**, puis réponse négative à la proposition de retour sur trajectoire.

Les fonctions spéciales de déplacement du curseur, sur une autre routine (**Spécial [✔ 4]**) ou en début de programme (**Spécial [✔ 3]**) entraînent, outre le comportement de recyclage décrit ci-dessous, une réinitialisation plus complète de l'application.

 En cas de déplacement du curseur, les logiciels ABB ne permettent pas de différencier les recyclages « corrects » (positionnement sur une instruction de mouvement) des recyclages « incorrects » (positionnement sur une autre instruction)

 Le fonctionnement de l'application lors de ces exceptions est le suivant :

1 – Lors du déplacement manuel du robot hors trajectoire :

- Les contrôles dynamiques sont inhibés.
- Les ordres et ordres outil sont mis à 0.
- Les éventuels CPP, événements ou événements outil en attente n'empêchent pas le déplacement du robot
- Les séquences demeurent pilotées et contrôlées uniquement pour affichage en cas de défaut (pas d'arrêt robot). Il est possible de piloter les séquences manuellement.

2- Lors d'un redémarrage "simple" ou à la fin d'un recalage:

- Les contrôles dynamiques sont restitués.
- Les ordres et ordres outil sont restitués.
- Les éventuels CPP, événements ou événements outil en attente sont à nouveau attendus.
- Les pilotages des séquences sont maintenus et les séquences sont toujours contrôlées selon leurs états **pilotés avant le décalage**. En cas de contrôle OK par rapport au pilotage en manuel mais non conforme au pilotage programmé, le défaut est signalé par le message « PILOTAGE SEQ MODIFIE EN MANU ».

3- Lors d'un recyclage

Le fonctionnement actuel de l'application est le suivant :

- Les contrôles dynamiques sont placés dans les états par défaut définis par l'intégrateur.
- Les ordres et les ordres outils sont mis à 0.
- Les CPP, événements et événements outil sont initialisés à l'état indifférent
- Les pilotages des séquences sont maintenus et les séquences sont toujours contrôlées selon leurs états **de pilotage courant**.
- La vitesse est limitée à 250 mm/s ; ce comportement est signalé par le message « Recyclage en cours Vit <=250 mm/s ».

12. CARTES D'ENTREES SORTIES

12.1 CARTE MANUT

Carte « MANUT » pour toutes les applications (DSQC 328 à l'emplacement 13)

ENTREES			SORTIES		
N°	**Mnémonique**	**Libelle**	**N°**	**Mnémonique**	**Libelle**
DI 1	**DREPLI**	demande de repli	DO1	**VREPLI**	Voyant de repli
DI 2	**CPAIR**	contrôle présence air	DO2		
DI3	**CTRL1**	Contrôle préhenseur (non dédié)	DO3	**PIL1**	Actionneur préhenseur (non dédié)
DI4	**CTRL2**	Contrôle préhenseur (non dédié)	DO4	**PIL2**	Actionneur préhenseur (non dédié)
DI5	**CTRL3**	Contrôle préhenseur (non dédié)	DO5	**PIL3**	Actionneur préhenseur (non dédié)
DI6	**CTRL4**	Contrôle préhenseur (non dédié)	DO6	**PIL4**	Actionneur préhenseur (non dédié)
DI7	**CTRL5**	Contrôle préhenseur (non dédié)	DO7	**PIL5**	Actionneur préhenseur (non dédié)
DI8	**CTRL6**	Contrôle préhenseur (non dédié)	DO8	**PIL6**	Actionneur préhenseur (non dédié)
DI9	**CTRL7**	Contrôle préhenseur (non dédié)	DO9	**PIL7**	Actionneur préhenseur (non dédié)
DI10	**CTRL8**	Contrôle préhenseur (non dédié)	DO10	**PIL8**	Actionneur préhenseur (non dédié)
DI11	**CTRL9**	Contrôle préhenseur (non dédié)	DO11	**PIL9**	Actionneur préhenseur (non dédié)
DI12	**CTRL10**	Contrôle préhenseur (non dédié)	DO12	**PIL10**	Actionneur préhenseur (non dédié)
DI13	**CTRL11**	Contrôle préhenseur (non dédié)	DO13		
DI14	**CTRL12**	Contrôle préhenseur (non dédié)	DO14		
DI15	**CTRL13**	Contrôle préhenseur (non dédié)	DO15		
DI16	**di_ClimOk**	Contrôle climatiseur (optionnel)	DO16		

Dans le cas où les entrées/sorties préhenseur sont situées sur une carte InterBUS Maître (DSQC512), la carte « MANUT » ne comprend que les signaux DREPLI, VREPLI, CPAIR et di_ClimOk. Dans ce cas, le mapping des entrées/sorties préhenseur dans la partie "maître" de la carte DSQC512 est au libre choix de l'intégrateur.

12.2 CARTE DIALOGUE GESTION STANDARD

Carte « Cart_Gest_Std » pour les applications sans réseau de terrain (DSQC 328 à l'emplacement 10)

ENTREES			SORTIES		
N°	**Mnémonique**	**Libelle**	**N°**	**Mnémonique**	**Libelle**
DI 1	**AEVRB**	Autorisation évolution robot	DO1	**RCReady**	robot prêt pour recevoir infos API
DI 2	**DMSP**	Demande mise sous puissance	DO2	**xMotorOn**	robot sous asservissements
DI3	**STOPPE**	Demande de stop	DO3	**AUTO**	commutateur robot sur auto
DI4	**DDCY**	Demande départ cycle	DO4	**MANU**	commutateur robot sur manu
DI5	**DEVERM**	mode déverminage	DO5	**REPLI**	robot au repli
DI6	**CVC_0**	Contrôle validation code à 0	DO6	**XcycleOn**	robot en cycle
DI7	**CVC_1**	Contrôle validation code à 1	DO7	**Reb_prg**	Robot au point de rebouclage
DI8	**CPCO**	Contrôle parité code	DO8		*Réserve*
DI9	**C1**	Code poids 1	DO9	**HPR**	Robot hors production
DI10	**C2**	Code poids 2	DO10	**AOP**	appel opérateur
DI11	**C4**	Code poids 4	DO11	**DE**	Dérive
DI12	**C8**	Code poids 8	DO12	**TC**	top cycle
DI13	**C16**	Code poids 16	DO13	**AI**	arrêt induit
DI14	**C32**	Code poids 32	DO14	**APE**	arrêt pour exploitation
DI15			DO15	**APF**	arrêt fonctionnel
DI16	**ENPRO**	robot en production	DO16	**APP**	arrêt pour panne

12.3 CARTE DIALOGUE AUTOMATE

Carte « Cart_Dial_Autom » pour les applications sans réseau de terrain (DSQC 328 à l'emplacement 12).

ENTREES			SORTIES		
N°	**Mnémonique**	**Libelle**	**N°**	**Mnémonique**	**Libelle**
DI 1	**EV1**	Evénement 1	DO1	**OR1**	ordre 1
DI 2	**EV2**	Evénement 2	DO2	**OR2**	ordre 2
DI3	**EV3**	Evénement 3	DO3	**OR3**	ordre 3
DI4	**EV4**	Evénement 4	DO4	**OR4**	ordre 4
DI5	**EV5**	Evénement 5	DO5	**OR5**	ordre 5
DI6	**EV6**	Evénement 6	DO6	**OR6**	ordre 6
DI7	**EV7**	Evénement 7	DO7	**OR7**	ordre 7
DI8	**EV8**	Evénement 8	DO8	**OR8**	ordre 8
DI9	**EV9**	Evénement 9	DO9	**OR9**	ordre 9
DI10	**EV10**	Evénement 10	DO10	**OR10**	ordre 10
DI11	**EV11**	Evénement 11	DO11	**OR11**	ordre 11
DI12	**EV12**	Evénement 12	DO12	**OR12**	ordre 12
DI13	**EV13**	Evénement 13	DO13	**OR13**	ordre 13
DI14	**EV14**	Evénement 14	DO14	**OR14**	ordre 14
DI15	**EV15**	Evénement 15	DO15	**OR15**	ordre 15
DI16	**EV16**	Evénement 16	DO16	**OR16**	ordre 16

12.4 CARTE DIALOGUE OUTIL

Carte « Cart_Dial_Outil » simulée par défaut.

ENTREES			SORTIES		
N°	**Mnémonique**	**Libelle**	**N°**	**Mnémonique**	**Libelle**
DI 1	**EVOU1**	Evénement Outil 1	DO1	**OROU1**	ordre Outil 1
DI 2	**EVOU2**	Evénement Outil 2	DO2	**OROU2**	ordre Outil 2
DI3	**EVOU3**	Evénement Outil 3	DO3	**OROU3**	ordre Outil 3
DI4	**EVOU4**	Evénement Outil 4	DO4	**OROU4**	ordre Outil 4
DI5	**EVOU5**	Evénement Outil 5	DO5	**OROU5**	ordre Outil 5
DI6	**EVOU6**	Evénement Outil 6	DO6	**OROU6**	ordre Outil 6
DI7	**EVOU7**	Evénement Outil 7	DO7	**OROU7**	ordre Outil 7
DI8	**EVOU8**	Evénement Outil 8	DO8	**OROU8**	ordre Outil 8
DI9	**EVOU9**	Evénement Outil 9	DO9	**OROU9**	ordre Outil 9
DI10	**EVOU10**	Evénement Outil 10	DO10	**OROU10**	ordre Outil 10
DI11	**EVOU11**	Evénement Outil 11	DO11	**OROU11**	ordre Outil 11
DI12	**EVOU12**	Evénement Outil 12	DO12	**OROU12**	ordre Outil 12
DI13	**EVOU13**	Evénement Outil 13	DO13	**OROU13**	ordre Outil 13
DI14	**EVOU14**	Evénement Outil 14	DO14	**OROU14**	ordre Outil 14
DI15	**EVOU15**	Evénement Outil 15	DO15	**OROU15**	ordre Outil 15
DI16	**EVOU16**	Evénement Outil 16	DO16	**OROU16**	ordre Outil 16

12.5 CARTE INTERBUS

Carte « INTERBUS » pour les applications équipées du réseau Interbus S (passerelle DSQC 351 à l'emplacement 10 ou partie esclave de la carte DSQC 512 – dans ce dernier cas inverser **di_ComOk**).

ENTREES			SORTIES		
N°	**Mnémonique**	**Libelle**	**N°**	**Mnémonique**	**libelle**
DI 1	**AEVRB**	Autorisation évolution robot	DO1	**RCReady**	robot prêt pour recevoir infos API
DI 2	**DMSP**	Demande mise sous puissance	DO2	**xMotorOn**	robot sous asservissements
DI3	**STOPPE**	Demande de stop	DO3	**AUTO**	commutateur robot sur auto
DI4	**DDCY**	Demande départ cycle	DO4	**MANU**	commutateur robot sur manu
DI5	**DEVERM**	mode déverminage	DO5	**REPLI**	robot au repli
DI6	**CVC_0**	Contrôle validation code à 0	DO6	**XcycleOn**	robot en cycle
DI7	**CVC_1**	Contrôle validation code à 1	DO7	**Reb_prg**	Robot au point de rebouclage
DI8	**CPCO**	Contrôle parité code	DO8		*réserve*
DI9	**C1**	Code poids 1	DO9	**HPR**	Robot hors production
DI10	**C2**	Code poids 2	DO10	**AOP**	appel opérateur
DI11	**C4**	Code poids 4	DO11	**DE**	dérive
DI12	**C8**	Code poids 8	DO12	**TC**	top cycle
DI13	**C16**	Code poids 16	DO13	**AI**	arrêt induit
DI14	**C32**	Code poids 32	DO14	**APE**	arrêt pour exploitation
DI15		*Réserve*	DO15	**APF**	arrêt fonctionnel
DI16		*Réserve*	DO16	**APP**	arrêt pour panne
DI17	**EV1**	Evénement 1	DO17	**OR1**	ordre 1
DI18	**EV2**	Evénement 2	DO18	**OR2**	ordre 2
DI19	**EV3**	Evénement 3	DO19	**OR3**	ordre 3
DI20	**EV4**	Evénement 4	DO20	**OR4**	ordre 4
DI21	**EV5**	Evénement 5	DO21	**OR5**	ordre 5
DI22	**EV6**	Evénement 6	DO22	**OR6**	ordre 6
DI23	**EV7**	Evénement 7	DO23	**OR7**	ordre 7
DI24	**EV8**	Evénement 8	DO24	**OR8**	ordre 8
DI25	**EV9**	Evénement 9	DO25	**OR9**	ordre 9
DI26	**EV10**	Evénement 10	DO26	**OR10**	ordre 10
DI27	**EV11**	Evénement 11	DO27	**OR11**	ordre 11
DI28	**EV12**	Evénement 12	DO28	**OR12**	ordre 12
DI29	**EV13**	Evénement 13	DO29	**OR13**	ordre 13
DI30	**EV14**	Evénement 14	DO30	**OR14**	ordre 14
DI31	**EV15**	Evénement 15	DO31	**OR15**	ordre 15
DI32	**EV16**	Evénement 16	DO32	**OR16**	ordre 16
DI33	**ENPRO**	robot en production	DO33		*réserve*
DI34		*Réserve*	DO34		*réserve*
DI35		*Réserve*	DO35		*réserve*
DI36		*Réserve*	DO36		*réserve*
DI37		*Réserve*	DO37		*réserve*
DI38		*Réserve*	DO38		*réserve*
DI39		*Réserve*	DO39		*réserve*
DI40		*Réserve*	DO40		*réserve*
DI41		*Réserve*	DO41		*réserve*
DI42		*Réserve*	DO42		*réserve*
DI43		*Réserve*	DO43		*réserve*
DI44		*Réserve*	DO44		*réserve*
DI45		*Réserve*	DO45		*réserve*
DI46		*Réserve*	DO46		*réserve*
DI47		*Réserve*	DO47		*réserve*
DI48		*Réserve*	DO48		*réserve*
DI49		*Réserve*	DO49	**MOT_COM**	mot commentaire suivi
DI50		*Réserve*	DO50	**MOT_COM**	mot commentaire suivi
DI51		*Réserve*	DO51	**MOT_COM**	mot commentaire suivi
DI52		*Réserve*	DO52	**MOT_COM**	mot commentaire suivi
DI53		*Réserve*	DO53	**MOT_COM**	mot commentaire suivi
DI54		*Réserve*	DO54	**MOT_COM**	mot commentaire suivi
DI55		*Réserve*	DO55	**MOT_COM**	mot commentaire suivi
DI56		*Réserve*	DO56	**MOT_COM**	mot commentaire suivi
DI57		*Réserve*	DO57	**MOT_COM**	mot commentaire suivi
DI58		*Réserve*	DO58	**MOT_COM**	mot commentaire suivi
DI59		*Réserve*	DO59	**MOT_COM**	mot commentaire suivi
DI60		*Réserve*	DO60	**MOT_COM**	mot commentaire suivi
DI61		*Réserve*	DO61	**MOT_COM**	mot commentaire suivi
DI62		*Réserve*	DO62	**MOT_COM**	mot commentaire suivi
DI63		*Réserve*	DO63	**MOT_COM**	mot commentaire suivi
DI64		*Réserve*	DO64	**MOT_COM**	mot commentaire suivi
DI65	**di_ComOk**	*Contrôle dialogue automate*			

12.6 CARTE PROFIBUS

Carte « PROFIBUS » pour les applications équipées du réseau Profibus DP (DSQC 352 à l'emplacement 10).

ENTREES			SORTIES		
N°	**Mnémonique**	**Libelle**	**N°**	**Mnémonique**	**libelle**
DI 1	**AEVRB**	Autorisation évolution robot	DO1	**RCReady**	robot prêt pour recevoir infos API
DI 2	**DMSP**	Demande mise sous puissance	DO2	**xMotorOn**	robot sous asservissements
DI3	**STOPPE**	Demande de stop	DO3	**AUTO**	commutateur robot sur auto
DI4	**DDCY**	Demande départ cycle	DO4	**MANU**	commutateur robot sur manu
DI5	**DEVERM**	mode déverminage	DO5	**REPLI**	robot au repli
DI6	**CVC_0**	Contrôle validation code à 0	DO6	**XcycleOn**	robot en cycle
DI7	**CVC_1**	Contrôle validation code à 1	DO7	**Reb_prg**	Robot au point de rebouclage
DI8	**CPCO**	Contrôle parité code	DO8		*réserve*
DI9	**C1**	Code poids 1	DO9	**HPR**	Robot hors production
DI10	**C2**	Code poids 2	DO10	**AOP**	appel opérateur
DI11	**C4**	Code poids 4	DO11	**DE**	dérive
DI12	**C8**	Code poids 8	DO12	**TC**	top cycle
DI13	**C16**	Code poids 16	DO13	**AI**	arrêt induit
DI14	**C32**	Code poids 32	DO14	**APE**	arrêt pour exploitation
DI15		*Réserve*	DO15	**APF**	arrêt fonctionnel
DI16		*Réserve*	DO16	**APP**	arrêt pour panne
DI17	**EV1**	Evénement 1	DO17	**OR1**	ordre 1
DI18	**EV2**	Evénement 2	DO18	**OR2**	ordre 2
DI19	**EV3**	Evénement 3	DO19	**OR3**	ordre 3
DI20	**EV4**	Evénement 4	DO20	**OR4**	ordre 4
DI21	**EV5**	Evénement 5	DO21	**OR5**	ordre 5
DI22	**EV6**	Evénement 6	DO22	**OR6**	ordre 6
DI23	**EV7**	Evénement 7	DO23	**OR7**	ordre 7
DI24	**EV8**	Evénement 8	DO24	**OR8**	ordre 8
DI25	**EV9**	Evénement 9	DO25	**OR9**	ordre 9
DI26	**EV10**	Evénement 10	DO26	**OR10**	ordre 10
DI27	**EV11**	Evénement 11	DO27	**OR11**	ordre 11
DI28	**EV12**	Evénement 12	DO28	**OR12**	ordre 12
DI29	**EV13**	Evénement 13	DO29	**OR13**	ordre 13
DI30	**EV14**	Evénement 14	DO30	**OR14**	ordre 14
DI31	**EV15**	Evénement 15	DO31	**OR15**	ordre 15
DI32	**EV16**	Evénement 16	DO32	**OR16**	ordre 16
DI33	**ENPRO**	robot en production	DO33		*réserve*
DI34		*Réserve*	DO34		*réserve*
DI35		*Réserve*	DO35		*réserve*
DI36		*Réserve*	DO36		*réserve*
DI37		*Réserve*	DO37		*réserve*
DI38		*Réserve*	DO38		*réserve*
DI39		*Réserve*	DO39		*réserve*
DI40		*Réserve*	DO40		*réserve*
DI41		*Réserve*	DO41		*réserve*
DI42		*Réserve*	DO42		*réserve*
DI43		*Réserve*	DO43		*réserve*
DI44		*Réserve*	DO44		*réserve*
DI45		*Réserve*	DO45		*réserve*
DI46		*Réserve*	DO46		*réserve*
DI47		*Réserve*	DO47		*réserve*
DI48		*Réserve*	DO48		*réserve*
DI49		*Réserve*	DO49	**MOT_COM**	mot commentaire suivi
DI50		*Réserve*	DO50	**MOT_COM**	mot commentaire suivi
DI51		*Réserve*	DO51	**MOT_COM**	mot commentaire suivi
DI52		*Réserve*	DO52	**MOT_COM**	mot commentaire suivi
DI53		*Réserve*	DO53	**MOT_COM**	mot commentaire suivi
DI54		*Réserve*	DO54	**MOT_COM**	mot commentaire suivi
DI55		*Réserve*	DO55	**MOT_COM**	mot commentaire suivi
DI56		*Réserve*	DO56	**MOT_COM**	mot commentaire suivi
DI57		*Réserve*	DO57	**MOT_COM**	mot commentaire suivi
DI58		*Réserve*	DO58	**MOT_COM**	mot commentaire suivi
DI59		*Réserve*	DO59	**MOT_COM**	mot commentaire suivi
DI60		*Réserve*	DO60	**MOT_COM**	mot commentaire suivi
DI61		*Réserve*	DO61	**MOT_COM**	mot commentaire suivi
DI62		*Réserve*	DO62	**MOT_COM**	mot commentaire suivi
DI63		*Réserve*	DO63	**MOT_COM**	mot commentaire suivi
DI64		*Réserve*	DO64	**MOT_COM**	mot commentaire suivi
DI65		*libre*	DO65		*libre*

		DI65 à DI128 → libre			*DO65 à DO128 → libre*
DI128		*libre*	DO128		*libre*
DI129	**di_ComOk**	*Contrôle dialogue automate*			

12.7 CARTE FIP

Carte « FIP » présente pour les applications équipées du réseau FIP I/O (DSQC XXX à l'emplacement 10).

ENTREES			SORTIES		
N°	**Mnémonique**	**Libelle**	**N°**	**Mnémonique**	**libelle**
DI 1	**AEVRB**	Autorisation évolution robot	DO1	**RCReady**	robot prêt pour recevoir infos API
DI 2	**DMSP**	Demande mise sous puissance	DO2	**xMotorOn**	robot sous asservissements
DI3	**STOPPE**	Demande de stop	DO3	**AUTO**	commutateur robot sur auto
DI4	**DDCY**	Demande départ cycle	DO4	**MANU**	commutateur robot sur manu
DI5	**DEVERM**	mode déverminage	DO5	**REPLI**	robot au repli
DI6	**CVC_0**	Contrôle validation code à 0	DO6	**XcycleOn**	robot en cycle
DI7	**CVC_1**	Contrôle validation code à 1	DO7	**Reb_prg**	Robot au point de rebouclage
DI8	**CPCO**	Contrôle parité code	DO8		
DI9	**C1**	Code poids 1	DO9	**HPR**	Robot hors production
DI10	**C2**	Code poids 2	DO10	**AOP**	appel opérateur
DI11	**C4**	Code poids 4	DO11	**DE**	dérive
DI12	**C8**	Code poids 8	DO12	**TC**	top cycle
DI13	**C16**	Code poids 16	DO13	**AI**	arrêt induit
DI14	**C32**	Code poids 32	DO14	**APE**	arrêt pour exploitation
DI15		*Réserve*	DO15	**APF**	arrêt fonctionnel
DI16	**ENPRO**	robot en production	DO16	**APP**	arrêt pour panne
DI17	**EV1**	événement 1	DO17	**OR1**	ordre 1
DI18	**EV2**	événement 2	DO18	**OR2**	ordre 2
DI19	**EV3**	événement 3	DO19	**OR3**	ordre 3
DI20	**EV4**	événement 4	DO20	**OR4**	ordre 4
DI21	**EV5**	événement 5	DO21	**OR5**	ordre 5
DI22	**EV6**	événement 6	DO22	**OR6**	ordre 6
DI23	**EV7**	événement 7	DO23	**OR7**	ordre 7
DI24	**EV8**	événement 8	DO24	**OR8**	ordre 8
DI25	**EV9**	événement 9	DO25	**OR9**	ordre 9
DI26	**EV10**	événement 10	DO26	**OR10**	ordre 10
DI27	**EV11**	événement 11	DO27	**OR11**	ordre 11
DI28	**EV12**	événement 12	DO28	**OR12**	ordre 12
DI29	**EV13**	événement 13	DO29	**OR13**	ordre 13
DI30	**EV14**	événement 14	DO30	**OR14**	ordre 14
DI31	**EV15**	événement 15	DO31	**OR15**	ordre 15
DI32	**EV16**	événement 16	DO32	**OR16**	ordre 16

ABB propose en option un coupleur FIP-IO de 128 entrées / 128 sorties.
Dans ce cas, le mapping est identique à celui de la carte Profibus, cf. 12.6 CARTE PROFIBUS sans utilisation de l'entrée 129 di_ComOk qui reste sur carte simulée

13. CONFIGURATIONS - LIMITES DE PRESTATIONS - RECEPTIONS

13.1 Vérification de la version logicielle de base ABB

≥ La vérification de la configuration du logiciel de base ABB se fait de la façon suivante :

- Sélectionner **AUTRE FENETRES** 
- placer le curseur sur Maintenance – puis 
- valider Voir / Informations système – puis 
- positionner le curseur sur Version du système – puis 

13.2 Documentation et fourniture de l'intégrateur

Voir le CD « Livret intégrateur »

14. REGLES A RESPECTER

14.1 Echanges automate lies aux états et événements robot

14.1.1 Pré affectations des entrées/sorties

Dans la configuration des entrées/sorties, plusieurs sorties et entrées de la baie sont reliées à des états ou des événements système du robot. Des équations logiques (cross connexions) sont également pré-programmées.

(extrait de la configuration Entrées/Sorties)

```
#
SYSSIG_OUT:

      -Status "CycleOn" -Signal "xCycleOn"        robot en cycle

      -Status "MotOnState" -Signal "xMotorOn"     robot sous puissance

      -Status "AutoOn" -Signal "xAutoOn"          robot en mode automatique

      -Status "RunchOk" -Signal "xRunchOk"        chaîne de sécurité fermée

#
SYSSIG_IN:

      -Signal "MSP" -Action "MotorOn"             mise sous puissance distante

      -Signal "STARTT" -Action "Start"            départ distant

      -Signal "xMotorOff" -Action "MotorOff"      mise hors puissance distante

      -Signal "xstop" -Action "Stop"              Stop distant
#
EIO_CROSS:

      -Lact "DDCY & *xCycleOn & *Retour & ENPRO" -Lres "STARTT"

      -Lact "DMSP & ENPRO" -Lres "MSP"
```

Pour de plus amples informations sur les entrées et sorties système ou sur les cross connexions, voir le Guide Utilisateur des baies ABB, chapitre Paramètres Système.

Par ailleurs, par logiciel, les équations suivantes sont également générées (indiquées ici sous le même format que les cross connexions):

```
-Lact " xAutoOn & xRunChOk" -Lres " LRCReady "
-Lact " xAutoOn " -Lres " AUTO "
-Lact " *xAutoOn " -Lres " MANU "
```

Les sorties automate “Autorisation d’évolution robot” reliée à AEVRB et “robot dans le flux de production” sont à programmer en fonction des spécificités du site.

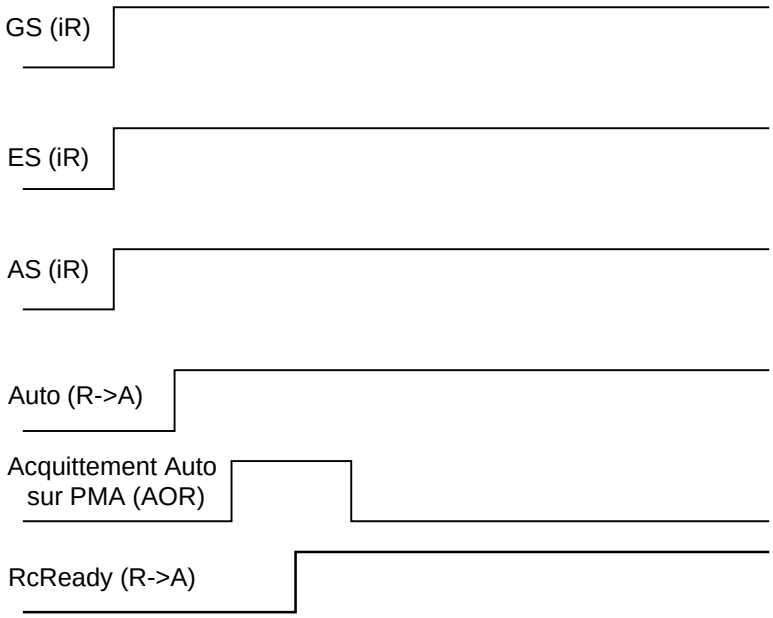
14.1.2 Chronogrammes à respecter

14.1.2.1 Légende des chronogrammes

IR	=	signal interne au robot
AOR	=	Action opérateur sur le robot
iA	=	signal interne à l'automate
R->A	=	Sortie robot, entrée automate
A->R	=	Sortie automate, entrée robot

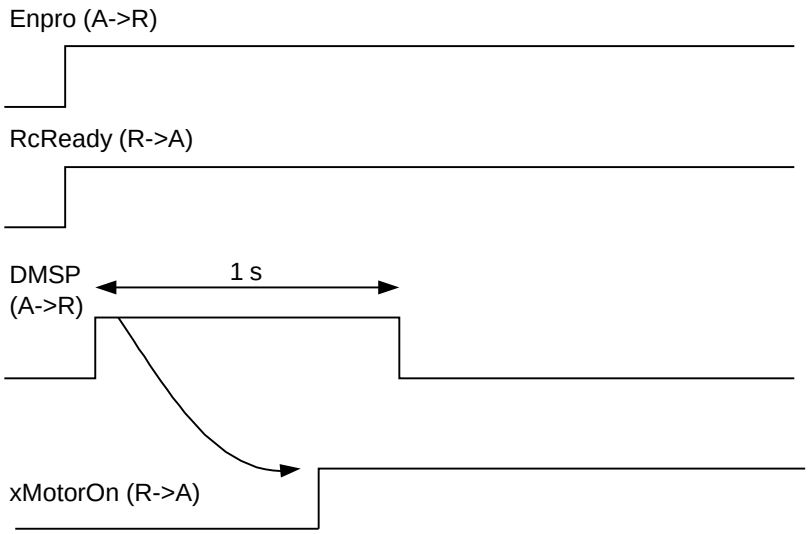
14.1.2.2 Chronogramme « baie prête »

La sortie RcReady indique que la baie est prête à prendre les commandes de l'automate. Elle est positionnée lorsque la chaîne de sécurité est correcte, et que la baie est en mode automatique.

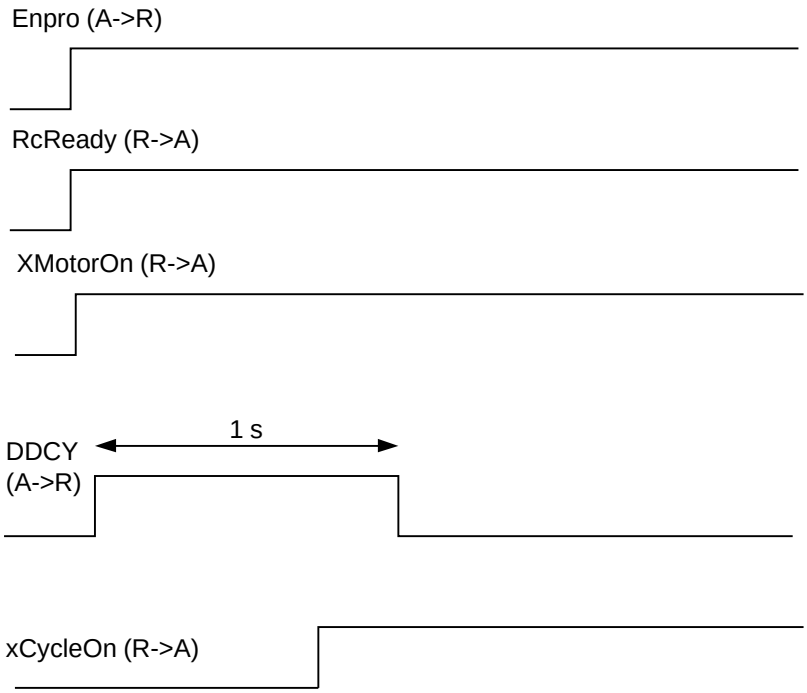


14.1.2.3 Chronogramme de mise sous puissance

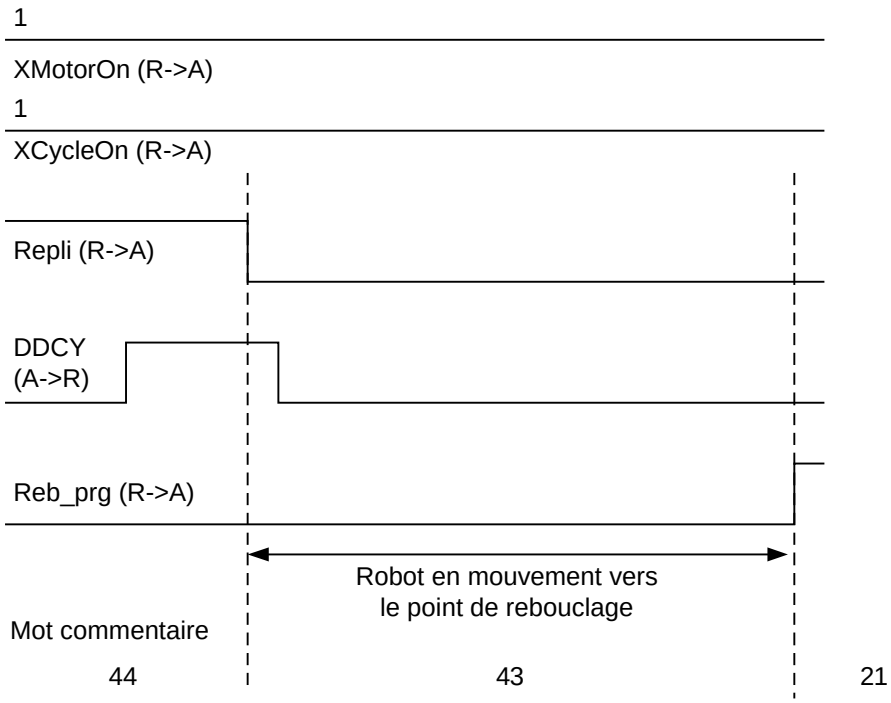
NB L'entrée DMSP n'est prise en compte qu'en mode automatique, et en production (ENPRO=1)



14.1.2.4 Chronogramme de mise en exécution

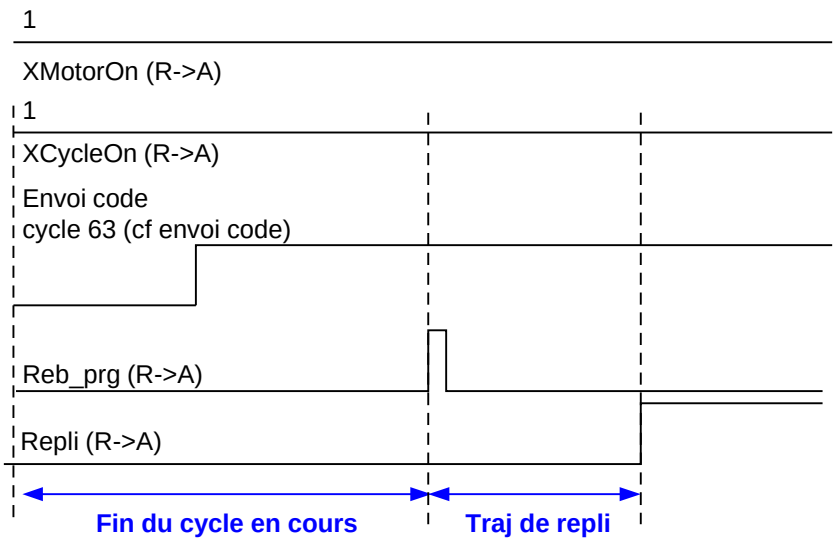


14.1.2.5 Chronogramme de mise en position de rebouclage depuis le repli



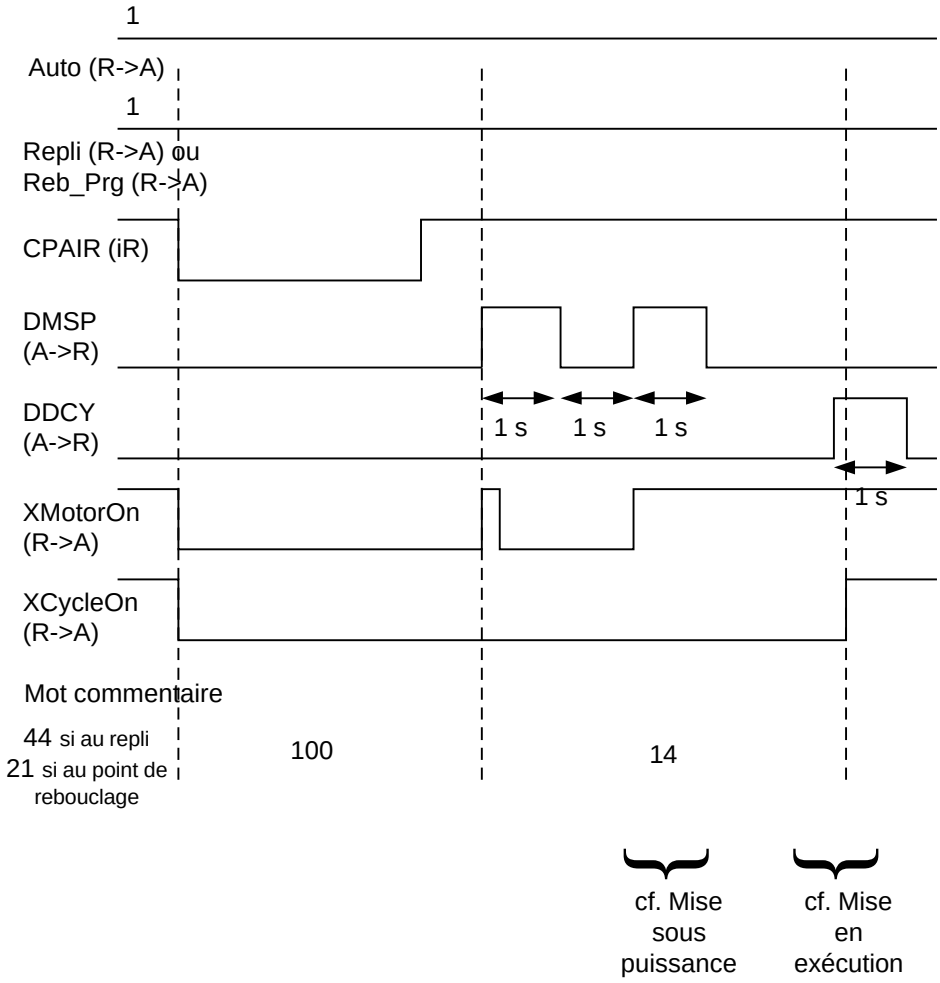
14.1.2.6 Chronogramme de mise en position de repli

NB Sur les conditions de repli, voir la Notice Utilisateur PROMIA, chapitre 3.3



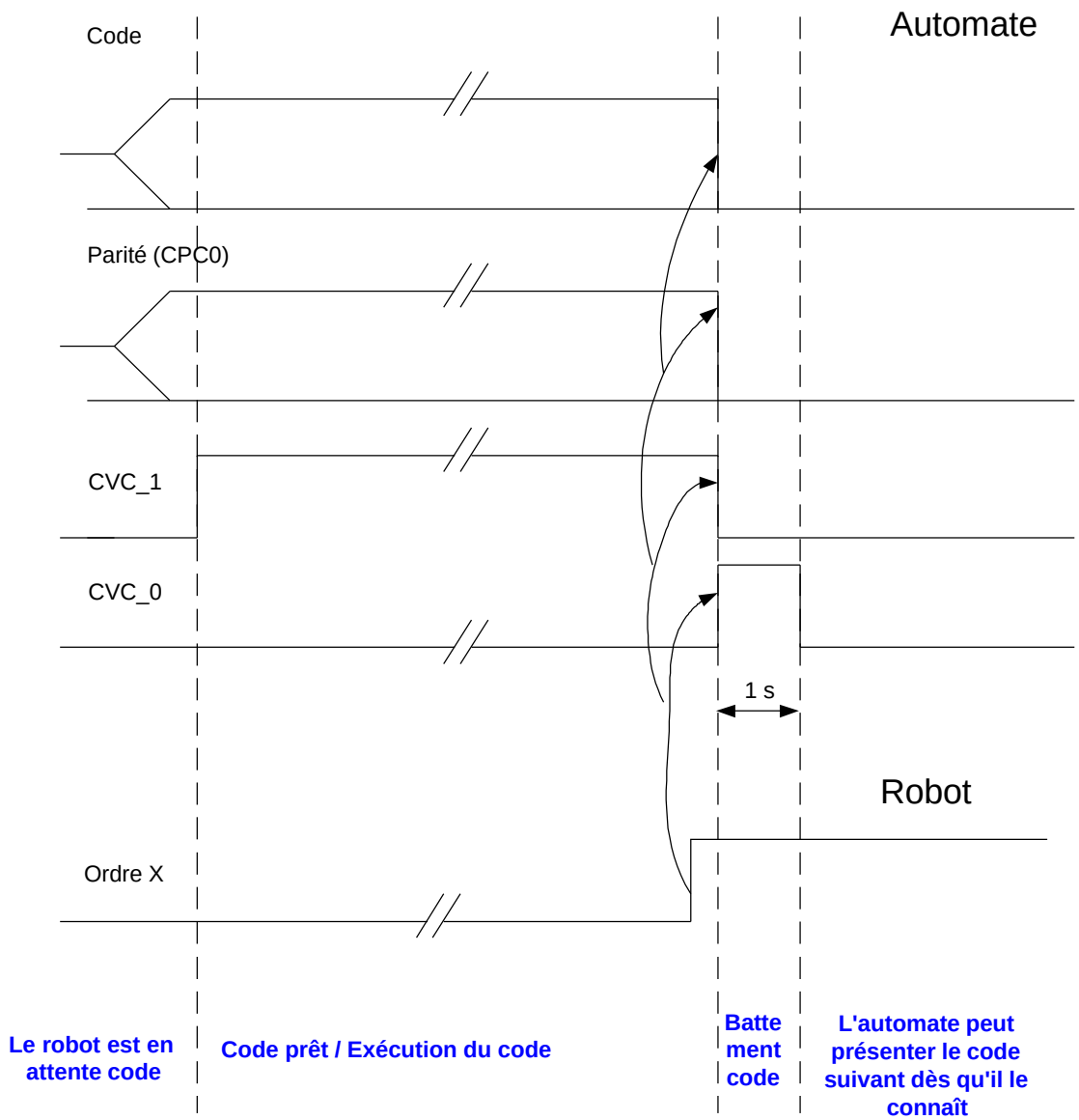
14.1.2.7 Chronogramme de réarmement du défaut air

Le réarmement des défauts à distance est en général impossible. Seuls les défauts avec validation (défaut air) peuvent automatiquement acquittés à distance sur la position de repli ou le point de rebouclage, en mode automatique.



14.1.2.8 Chronogramme de l'échange code

- Voir l'extrait de la norme EM81.EA.010, donné en annexe.
- A noter qu'il n'y a pas d'information spécifique à l'acquittement de la prise en compte du code par le robot. C'est en général un ordre émis par le robot qui signale à l'automate que le programme est effectivement lancé. L'automaticien et le programmeur robot doivent se mettre d'accord sur cet ordre.
- Une fois un programme lancé, un nouveau code ne peut être présenté qu'après « un battement code » (passage à 0 du code, de la parité, de CVC_1 et battement de CVC_0).



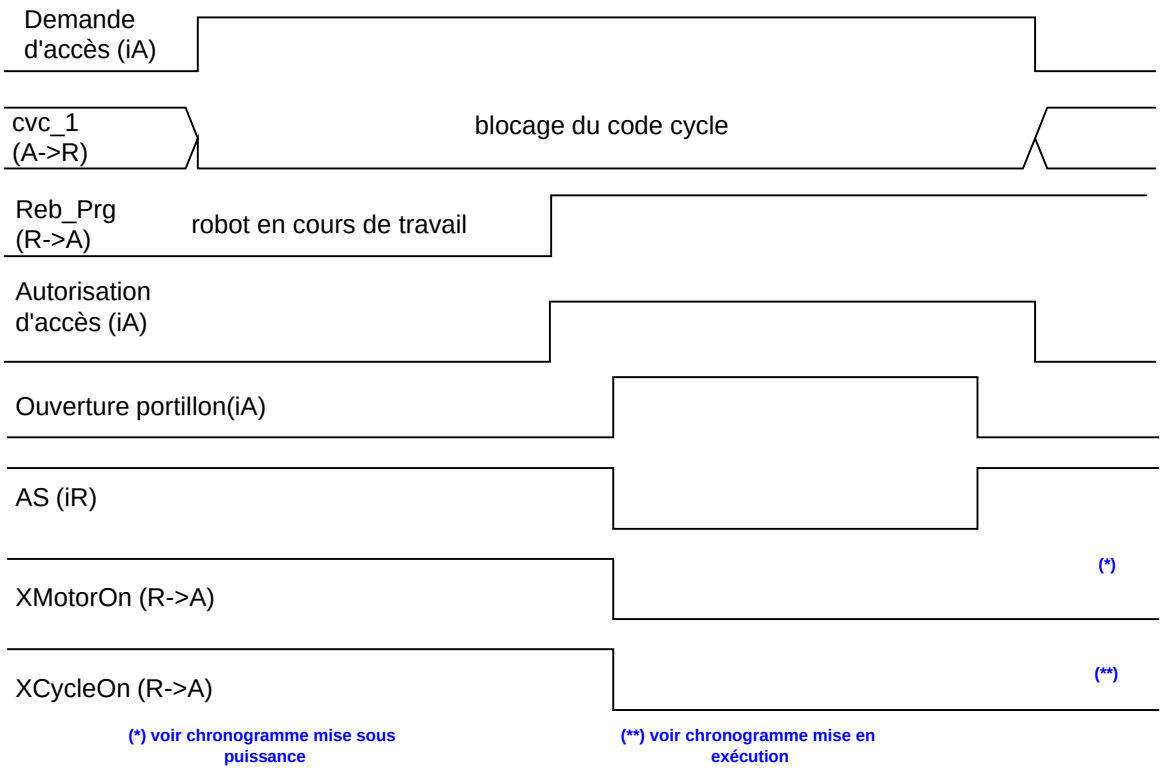
14.1.2.9 Codage CPCO

L'algorithme qui régit la valeur de CPCO est le suivant : le nombre total de bits à 1, du mot formé par le code et CPCO, doit être impair.

code	nb bits du code à 1	CPC0	code	nb bits du code à 1	CPC0
0	0	1	**32**	1	**0**
1	1	**0**	**33**	2	**1**
2	1	**0**	**34**	2	**1**
3	2	**1**	**35**	3	**0**
4	1	**0**	**36**	2	**1**
5	2	**1**	**37**	3	**0**
6	2	**1**	**38**	3	**0**
7	3	**0**	**39**	4	**1**
8	1	**0**	**40**	2	**1**
9	2	**1**	**41**	3	**0**
10	2	**1**	**42**	3	**0**
11	3	**0**	**43**	4	**1**
12	2	**1**	**44**	3	**0**
13	3	**0**	**45**	4	**1**
14	3	**0**	**46**	4	**1**
15	4	**1**	**47**	5	**0**
16	1	**0**	**48**	2	**1**
17	2	**1**	**49**	3	**0**
18	2	**1**	**50**	3	**0**
19	3	**0**	**51**	4	**1**
20	2	**1**	**52**	3	**0**
21	3	**0**	**53**	4	**1**
22	3	**0**	**54**	4	**1**
23	4	**1**	**55**	5	**0**
24	2	**1**	**56**	3	**0**
25	3	**0**	**57**	4	**1**
26	3	**0**	**58**	4	**1**
27	4	**1**	**59**	5	**0**
28	3	**0**	**60**	4	**1**
29	4	**1**	**61**	5	**0**
30	4	**1**	**62**	5	**0**
31	5	**0**	**63**	6	**1**

14.1.2.10 Chronogramme de la demande d'accès

Sur une demande d'accès, l'automate bloque le code cycle par la mise à 0 de CVC_1. Lorsque le robot atteint le point de rebouclage, il autorise l'accès, l'opérateur ouvre le portillon qui fait tomber AS. En fin d'intervention, l'automate repositionne CVC_1. Le redémarrage en cycle s'effectue ensuite selon les chronogrammes « mise sous puissance » et « mise en exécution ».



Ce chronogramme est un chronogramme de principe. En pratique, le programme automate devra tenir compte des éventuelles erreurs pouvant survenir pendant le cycle en cours et réagir en fonction du mot d'état du suivi des moyens (cf ci après).

➢ En cas de défaut APP ou AI, autorisation immédiate de l'accès sans attendre l'information « Reb_Prg »

➢ En cas de défaut de type APF ou APE, attente de la fin de la routine de service ou de la demande de repli, tout en générant les signaux éventuellement attendus (DDCY), qui repositionneront finalement le robot au point de rebouclage

➢ En cas de défaut DE, armer une tempo. Si le défaut disparaît avant la fin de la tempo, continuer à attendre la fin de cycle. S'il persiste, forcer la fin de cycle, tout en positionnant AEVRB à 0 pour éviter d'éventuels aléas.

14.2 Echanges automates en trajectoire (ordres - événements)

Pour la gestion des interférences robot, les ordres émis par un robot doivent être reflétés par l'automate dans les événements émis pour les autres robots.

14.3 Suivi des moyens (Norme réf. : EM81.EA.040)

14.3.1 Définition des mots d'état

Utilisés par l'application pour remonter au suivi des moyens les défauts rencontrés.

HPR : Hors Production.

APP : Arrêt pour Panne
C'est un défaut qui demande une intervention opérateur (Exemples : réarmement local de la baie, acquittement d'un défaut sur la baie,...).

APF : Arrêt Fonctionnel
Le robot fonctionne, mais fait autre chose que de la production (Exemple : programme de service en cours).

APE : Arrêt pour Exploitation
Le robot est arrêté suite à une demande opérateur (Exemples : Robot au repli ou est en train de gagner la position de repli...).

AI : Arrêt Induit
Le robot est arrêté, car il lui manque une information logique de la part de l'automate (Exemples : attente du code de cycle, attente événement de l'automate...).

TC : Top Cycle
Le robot signale à l'automate qu'il commence un cycle de travail : il met cette information à l'état 1 pendant 1 seconde. Le TC n'est pas activé sur les trajectoires de service 1, service 2, service 3,
service 4 et de REPLI.

DE : Dérive
Le robot signale un problème qui ne l'arrête pas obligatoirement (Exemple : défaut Présence Pièce...). Note: si le problème dure, c'est l'automate qui compte les temps de cycle des composants d'automatisme, et c'est lui qui signale au suivi central (au bout d'un certain temps) que le robot est arrêté pour " dépassement de temps de cycle ".

AOP = Appel OPérateur
Le process robot signale une demande d'intervention opérateur (Exemple : Alarme changement d'outillage manuel)

14.3.2 Définition des mots commentaires

Il décrit le numéro (entier de 0 à 65535) de l'événement suivi. Ce numéro est à renseigner par l'application. Si plusieurs événement sont présent simultanément, seul celui qui la priorité la plus élevée est documenté.
Ordre des priorités : HPR>APP>APE>AI>APF>DE>AOP

CODE	Libellé de l'état du robot	Nature
1	PROGRAMME CELLULE ABANDONNE	APP
2	DEFAUT RESEAU DE TERRAIN ENTREES/SORTIES	APP
10	ROBOT HORS PUISSANCE	APP
11	STOP	APP
12	PERTE MARCHE AUTOMATIQUE	APP
13	ROBOT HORS PRODUCTION	HPR
14	ECRAN SYSTEME A VALIDER	APP
20	ATTENTE BATTEMENT CODE	AI
21	ATTENTE CODE PIECE	AI
22	DEFAUT PARITE	AI
23	PROGRAMME ROBOT INEXISTANT	AI
24	ATTENTE AUTORISATION EVOLUTION	AI
30	ATTENTE DEPART PROGRAMME ROBOT	APE
38	ATTENTE POINT REBOUCLAGE ATTEINT	APE
39	DEP SERVICES AU Pt_Reb SEULEMENT	APE
40	METTRE LE ROBOT AU REPLI	APE
41	ROBOT AU REPLI	APE
42	ROBOT PENDANT REPLI	APE
43	ROBOT PENDANT LANCE	APE
44	ROBOT ATTENTE DEPART POUR LANCE	APE
45	PROGRAMME DE SERVICE 1 EN COURS	APF
46	PROGRAMME DE SERVICE 2 EN COURS	APF
47	PROGRAMME DE SERVICE 3 EN COURS	APF
48	PROGRAMME DE SERVICE 4 EN COURS	APF
50	ATTENTE EVENEMENT NON DOCUMENTE	AI
51	ATTENTE EVENEMENT 1	AI
52	ATTENTE EVENEMENT 2	AI
53	ATTENTE EVENEMENT 3	AI
54	ATTENTE EVENEMENT 4	AI
55	ATTENTE EVENEMENT 5	AI
56	ATTENTE EVENEMENT 6	AI
57	ATTENTE EVENEMENT 7	AI
58	ATTENTE EVENEMENT 8	AI
59	ATTENTE EVENEMENT 9	AI
60	ATTENTE EVENEMENT 10	AI
61	ATTENTE EVENEMENT 11	AI
62	ATTENTE EVENEMENT 12	AI
63	ATTENTE EVENEMENT 13	AI
64	ATTENTE EVENEMENT 14	AI
65	ATTENTE EVENEMENT 15	AI
66	ATTENTE EVENEMENT 16	AI
70	ATTENTE EVENEMENT OUTIL NON DOCUMENTE	DE
71	ATTENTE EVENEMENT OUTIL 1	DE
72	ATTENTE EVENEMENT OUTIL 2	DE
73	ATTENTE EVENEMENT OUTIL 3	DE
74	ATTENTE EVENEMENT OUTIL 4	DE
75	ATTENTE EVENEMENT OUTIL 5	DE
76	ATTENTE EVENEMENT OUTIL 6	DE
77	ATTENTE EVENEMENT OUTIL 7	DE
78	ATTENTE EVENEMENT OUTIL 8	DE
79	ATTENTE EVENEMENT OUTIL 9	DE
80	ATTENTE EVENEMENT OUTIL 10	DE
81	ATTENTE EVENEMENT OUTIL 11	DE
82	ATTENTE EVENEMENT OUTIL 12	DE
83	ATTENTE EVENEMENT OUTIL 13	DE

84	ATTENTE EVENEMENT OUTIL 14	**DE**
85	ATTENTE EVENEMENT OUTIL 15	**DE**
86	ATTENTE EVENEMENT OUTIL 16	**DE**
100	DEFAUT PRESENCE AIR	**APP**
110	DEFAUT SEQUENCE NON DOCUMENTE	**DE**
111	DEFAUT SEQUENCE 1	**DE**
112	DEFAUT SEQUENCE 2	**DE**
113	DEFAUT SEQUENCE 3	**DE**
114	DEFAUT SEQUENCE 4	**DE**
115	DEFAUT SEQUENCE 5	**DE**
116	DEFAUT SEQUENCE 6	**DE**
117	DEFAUT SEQUENCE 7	**DE**
118	DEFAUT SEQUENCE 8	**DE**
130	DEFAUT PRESENCE PIECE	**DE**
131	DEFAUT PRESENCE PIECE 1	**DE**
132	DEFAUT PRESENCE PIECE 2	**DE**
133	DEFAUT PRESENCE PIECE 3	**DE**
134	DEFAUT PRESENCE PIECE 4	**DE**
135	DEFAUT PRESENCE PIECE 5	**DE**
136	DEFAUT PRESENCE PIECE 6	**DE**
137	DEFAUT PRESENCE PIECE 7	**DE**
138	DEFAUT PRESENCE PIECE 8	**DE**
168	REDEMARRAGE IMPOSSIBLE => STOP/EXEC	**APP**

15. FEUILLE DE REMARQUES

Cette feuille est à retourner à :

ABB MC
M . BEKAERT
Département R&D Support Produit
Rue de l'équerre - ZI des Béthunes
95310 st Ouen l'aumône
Fax : 01.34 40 25 10
☎ : 01.34 40 23 34
E-MAIL : michel.bekaert@fr.abb.com

Emetteur :

Date :

Nom :

Société :

Adresse :

Appréciation fournisseur :

	Oui	Non
Les problèmes à résoudre sont-ils clairement explicités ?		
Les explications fonctionnelles sont-elles suffisamment développées ?		
La partie mise en service est-elle facilement exploitable ?		
Les exemples développés sont-ils suffisants pour concrétiser les principes ?		

Remarques :