

Objectifs visés :

- ✓ *Commande de mouvements articulés*
- ✓ *Séquence des mouvements par rapport au pupitre opérateur*

0. Pupitre Opérateur

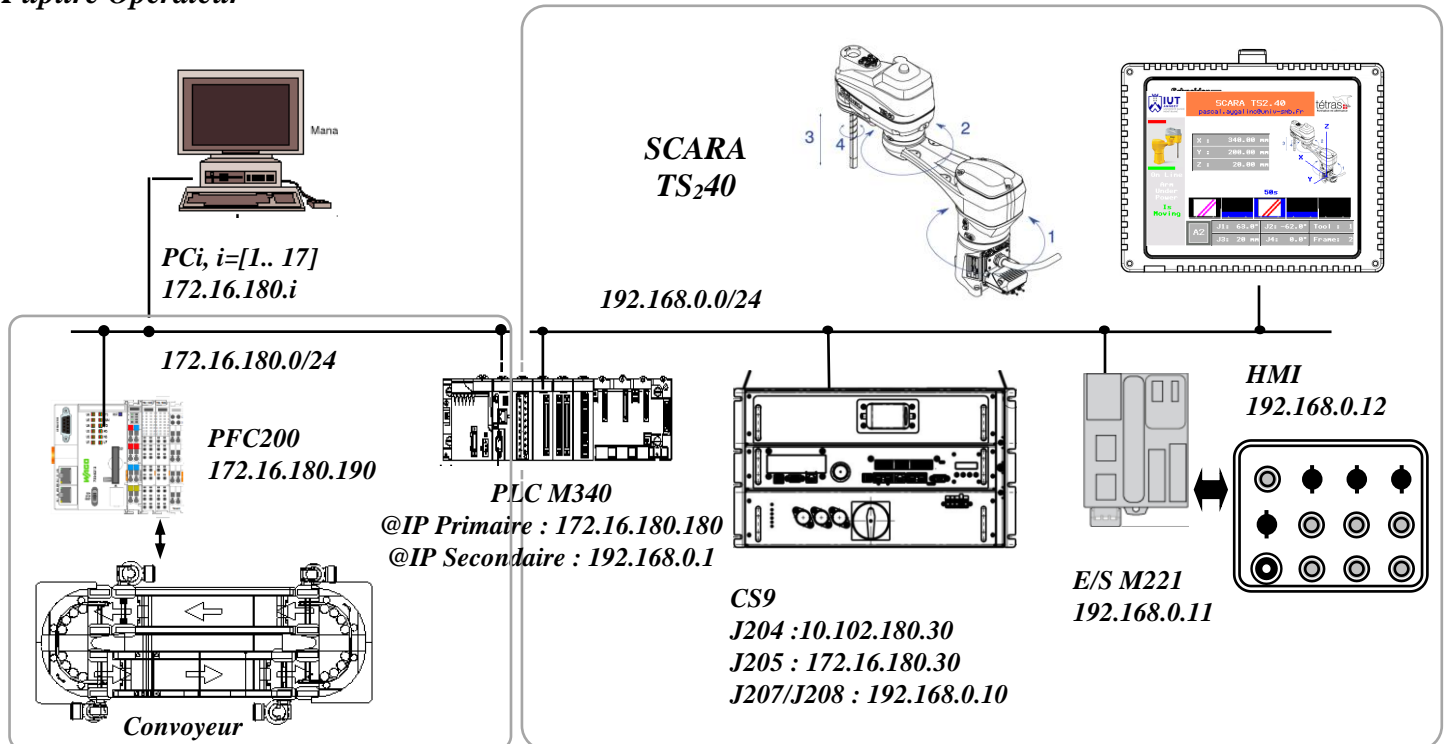


Figure 1 : Cellule robotisée de la salle C180 : (PLC M340, CS9, M221, HMI et PFC200)

Au niveau matériel, on dispose :

- ✓ d'un robot **Stäubli Scara TS240 collaboratif** (4 axes) et de son contrôleur **CS9**, équipé de son **pendant MCP** et de son **sélecteur de modes de marche WMS9**,
- ✓ d'un convoyeur **TS2plus Bosch RexRoth** sur lequel circulent des **palettes** portant des produits,
- ✓ d'un contrôleur **PFC200** du constructeur **WAGO** de type **Little Endian** qui gère une partie du convoyeur mais qui se comporte comme un **ilot d'E/S** accessible via **Ethernet/IP** pour la commande du **Guichet**,
- ✓ d'un automate **Schneider** de type **M340** dont l'orientation des données est de type **Little Endian**. Il est équipé de **deux coupleurs réseaux BMX NOC 0401.2** qui permettent d'obtenir :
 - pour le premier : une communication de type **Ethernet** avec les stations de développement (**PCs**) sur le **réseau privé de la salle C180 (classe B : 172.16.180.0/24)**. Elle est nécessaire pour le **téléchargement du code** dans l'automate et la **surveillance de son exécution** en mode connecté. Il permet aussi de scruter le **contrôleur PFC200** afin d'établir via **Ethernet/IP** la commande du **Guichet** du convoyeur sur lequel le robot **Scara** viendra saisir les pièces,
 - pour le second : un réseau de type **Ethernet/IP** pour les **échanges périodiques** à réaliser entre l'**API M340**, la **baie CS9** et le **contrôleur M221**.
- ✓ d'un contrôleur **Schneider** de type **M221** dont la programmation réalisée permet l'accès au **pupitre opérateur** via **Ethernet/IP** et de fournir les informations de supervision à l'écran **HMI** à l'aide du protocole **Modbus-Tcp**,

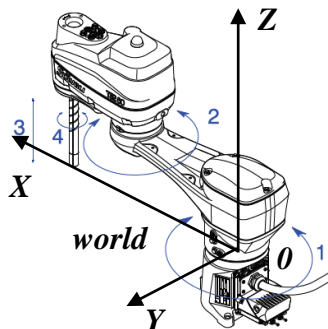
Lors des *tps* précédents, ont été étudiés et réalisés :

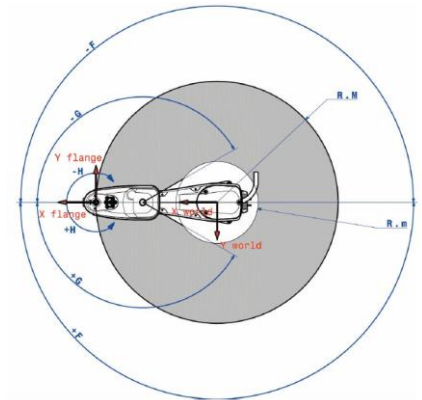
- ✓ la configuration du premier réseau **Ethernet/IP** comprenant un équipement (**PFC200**) et la communication à réaliser entre **l'API M340** et ce contrôleur. Elle a permis d'établir une première ébauche du mode de production normale (**mode F1**) du poste **Guichet** en **isolation** situé sur le convoyeur,
- ✓ la configuration du second réseau **Ethernet/IP** comprenant trois équipements (**l'API M340**, **la baie CS9** et **le contrôleur M221**) et leur communication respective,
- ✓ la structure de base logicielle du traitement utilisateur dans **l'API M340** qui nécessite la **réservation** d'un objet du type **T_StaeubliRobot** pour représenter **le robot et son contrôleur**. Cet objet doit être lu en début du cycle automate à l'aide du bloc fonctionnel **VAL_ReadAxesGroup** et doit être renvoyé à la baie en fin de cycle via **VAL_WriteAxesGroup** afin de transmettre les commandes de mouvement établies durant le cycle.
- ✓ La mise en œuvre d'une programmation hiérarchisée sous **Control Expert** basée sur l'exploitation d'un **Gemma** permettant d'obtenir à partir du **mode D1** les conditions initiales souhaitées pour la cellule (cf. **mode A1**) et un fonctionnement de production normale (**mode F1 et A2**) à partir de ce dernier.

Ce *tp* portera sur l'utilisation du **pupitre opérateur** pour établir le mode de **fonctionnement manuel** de la cellule (**mode F4**). En effet, pour la maintenance, on souhaite pouvoir modifier la position angulaire de tous les axes (**simultanément ou non**) du robot en fonction des éléments de commandes dont dispose l'opérateur sur le **pupitre**. Pour ce mode, vous serez amené à mettre en œuvre le bloc fonctionnel **MC_MoveAxisAbsolute** dont on rappelle que les **coordonnées des mouvements sont articulaires** (type **T_JointPos** cf. annexe 2)

1. Les butées articulaires

Les domaines de variation admissibles configurés avec le pendant pour chacun des axes (**J1, J2, J3 et J4**) sur le robot **Scara** de la cellule en salle **C180** vous sont donnés par le tableau suivant :

	Axe 1 : J1	$[-180^{\circ}, +180^{\circ}]$ (550°/s)
	Axe 2 : J2	$[-141^{\circ}, +141^{\circ}]$ (720°/s)
	Axe 3 : J3 (Z)	$[0, 200 \text{ mm}]$ (2500mm/s)
	Axe 4 : J4	$[-400^{\circ}, +400^{\circ}]$ (2500°/s)
	Rayon Max : R.M	460 mm
	Rayon Min : R.m	150 mm
	Rappel : Frein	Axe 3 et 4



Les valeurs mentionnées définissent donc les valeurs limites à donner sur chacun des axes lors des commandes articulaires. Au-delà de ces valeurs, aucun mouvement ne sera fait avec en plus une erreur à la clé.

2. Gmma : Graphe des modes de marche et d'arrêt

Comme le montre le **gmma** de l'annexe, **toute erreur de communication** conduira le graphe dans **l'état D1**, et le **figera dans cet état** tant que la communication n'est pas opérationnelle. Par rapport au *tp* précédent, le **gmma** compte un mode de plus correspondant au **mode manuel** demandé (**rectangle F4**).

2.1 Renseigner les conditions de passage **GM_T_A1toF4**, **GM_T_A1toF1** et **GM_F4toA6** afin d'une part que **les structures de choix** soient **exclusives** et d'autre part que le mode **A6** ne commence que si aucun mouvement ne soit en cours dans le mode **F4**. En effet, lors de la conception du mode **A6**, on a supposé que la **pile de mouvements** était **vide** avec une **baie CS9 opérationnelle** (sans erreur).

3. Mode F4 : Mode Manuel

Durant ce mode, on interdira tout accès au poste **guichet** et la **colonne** sera de couleur **jaune clignotante (T#1s)**. On n'autorisera les commandes de mouvements via le pupitre opérateur que si le poste est vide.

3.1 Avant de s'intéresser à tous les axes, on commencera par **l'axe linéaire J3**. L'opérateur dispose de 2 boutons poussoirs pour monter ou descendre cet axe (cf. *annexe 3*). Il est clair que si l'opérateur effectue une demande antagoniste (*appui simultané sur les 2 boutons poussoirs*), aucun mouvement ne doit avoir lieu.

Si la demande de mouvement est valide et que le poste est vide, le principe que vous utiliserez consiste à :

- ✓ prendre comme position finale pour **l'axe Z**, la borne inférieure ou supérieure en fonction du bouton poussoir activé par l'opérateur. Pour les **autres axes**, on conserve leur **position courante**,
- ✓ puis exécuter un **mouvement articulaire autorisant la préemption (Aborting)** à l'aide des positions articulaires établies précédemment.

Durant le mouvement, deux attitudes de l'opérateur devront être considérées :

- soit il décide d'arrêter le mouvement en relâchant le bouton poussoir qui l'a provoqué (*ou en opérant une demande antagoniste malgré lui*). Ceci conduira à utiliser une instance du bloc fonctionnel **MC_GroupStop** pour mettre fin au mouvement et aussi vider la pile. Puis, afin d'autoriser de nouvelles éventuelles commandes de mouvement, il faudra faire appel à une instance du bloc **MC_Continue**,
- soit il maintient le bouton poussoir et décide donc d'aller jusqu'à la position finale. Lorsque le mouvement est terminé, il est sorti de la pile par le système. Il convient alors de se mettre en attente d'une nouvelle demande de mouvement valide lorsque l'opérateur relâche le bouton poussoir.

*Etablir le **grafcet fonctionnel** répondant à ce principe de fonctionnement pour établir le **mode manuel**. On veillera à ce que les structures de choix présentes dans le modèle soient mutuellement exclusives.*

3.2 L'implantation de cet automatisme s'effectuera suivant la structure d'une machine de **MEALY**. Etablir la traduction de votre **grafcet** dans une section en langage **SFC**, et en déduire le contenu de la section postérieure au graphe écrite en langage **ST**. C'est dans cette partie que vous serez amené à commander l'exécution des trois instances des blocs fonctionnels nécessaires pour réaliser ce mode (**MC_MoveAxisAbsolute**, **MC_GroupStop**, **MC_Continue**).

3.3 Compléter le graphe précédent afin de prendre en compte des demandes de mouvement sur les autres axes, qu'elles s'effectuent ou non les unes après les autres, ou pendant que l'une est en cours, une autre est demandée sur un autre axe.

Par rapport au principe énoncé précédemment, une attitude supplémentaire de l'opérateur est à considérer. C'est celle de vouloir ajouter ou retirer un ou plusieurs axes de déplacement durant le mouvement du robot. Pour connaître ce changement de consigne, vous serez amené à comparer l'état du pupitre qui a donné lieu au mouvement précédent avec son état actuel. La prise en compte de cette nouvelle consigne se traduira par une relance de la commande de mouvement qui préemptera la précédente et imposera une nouvelle destination finale.

4. Partie TP

La mise en œuvre associée à ce **TD/TP** reste générale et ne se substitue pas aux **instructions de sécurité** détaillées contenues dans les **manuels** des produits **Schneider Electric SA** et **Stäubli**, et leurs caractéristiques techniques. Ces documents doivent **être lus attentivement et appliqués** avant de se lancer dans la partie **TP**. On rappelle que le **non-respect** de ces **instructions** peut provoquer **la mort, des blessures graves** ou **des dommages matériels**.

4.1 Après s'être connecté(e) à une station PC, lancer la chaîne de développement **Control Expert** à l'aide du raccourci disponible sur le bureau.

4.2 Reprendre l'application du **tp** précédent et la terminer si besoin.

4.3 Programmation du **Gmma**

4.3.1 Réserver un objet booléen (**xF4**) nécessaire pour rendre la scrutation conditionnelle des sections relatives au mode **F4**

4.3.2 Section **GM_Gmma**

- ✓ Compléter le **gmma** afin de faire apparaitre l'étape supplémentaire : **GM_S_F4**.
- ✓ Puis renseigner les transitions **GM_T_A1toF4**, **GM_T_F4toA6** et **GM_A1toF1** conformément à votre analyse (cf. 2.1)

4.3.3 Section **GM_GmmaPost**

- ✓ Positionner le bit **xF4** qui autorise la scrutation des sections **F4_ModeManuel** et **F4_ModeManuelPost** en fonction de l'état du **gmma**.
- ✓ Compléter la structure de choix multiple afin de fournir à l'écran **hmi** l'état du **Gmma** sous forme hexadécimale pour le nouvel état **F4**.

4.4 Programmation du mode **F4 : Mode Manuel**

4.4.1 Créer deux sections conditionnelles (objet : **xF4**) l'une en langage **SFC F4_ModeManuel** et **F4_ModeManuelPost** en langage **ST**.

4.4.2 Section **F4_ModeManuel** en langage **SFC**

- ✓ Créer une instance du bloc fonctionnel **MC_MoveAxisAbsolute** (**InstMoveAxisAbsoluteF4**)
- ✓ Créer une donnée **moveAxisAbsoluteF4** du type **TMoveAbsolute** qui facilitera l'appel de l'instance précédente ainsi qu'une donnée de type **jointPosF4** de type **T_JointPos**.
- ✓ Réserver les instances des blocs fonctionnels associés aux commandes de contrôle de mouvement nécessaires pour effectuer ce mode (**MC_GroupStop**, et **MC_GroupContinue**).
- ✓ Réserver les objets suivants pour les appels de ces blocs fonctionnels à l'aide de l'éditeur de données onglet **Variables** : **ctrlStatusGroupStopF4** et **ctrlStatusGroupContinueF4** de type **TCtrlStatusGroup**
- ✓ Etablir le graphe afin qu'il corresponde à celui de votre étude (cf. 3.2)
- ✓ Utiliser les actions associées aux étapes de type impulsioneille **PI** pour modifier le point articulaire de destination à atteindre, et définir les caractéristiques et propriétés des déplacements. On prendra :
 - pour les vitesses (**Velocity**) et accélérations angulaires (**Acceleration**, **Deceleration**) : **100.0**
 - pour les vitesses max du centre outil (TCP) **CartesianVel** : **500 mm/s**, **RotationalVel** : **60 °/s**

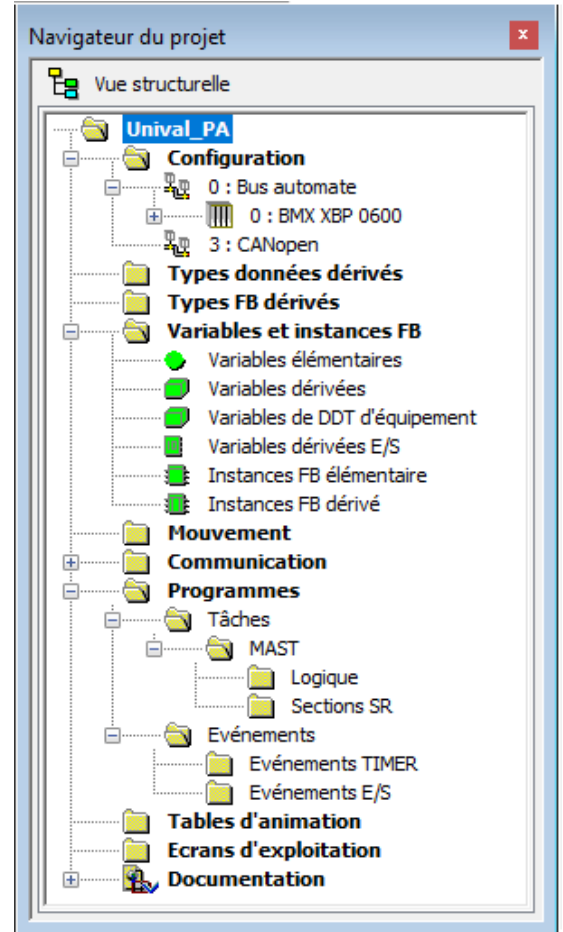


Figure 2

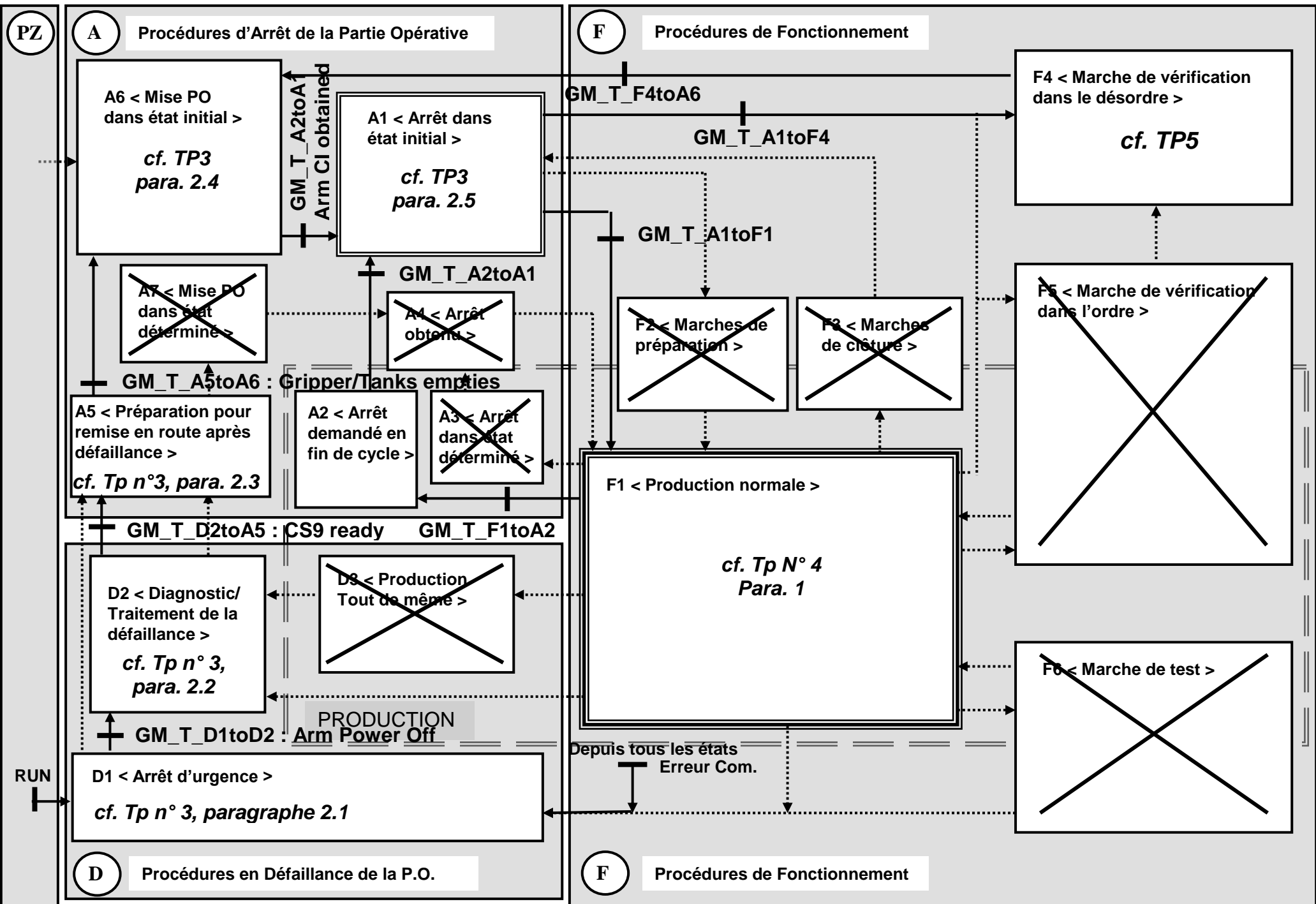
4.4.3 Section **F4_ModeManuelPost** en langage **ST** :

- ✓ Modifier les équations des actionneurs du guichet (la colonne et les bloqueurs).
- ✓ Faire scruter les instances des différents blocs fonctionnels nécessaire à ce mode en fonction de l'état du graphe **SFC**.

4.4.4 Compiler puis transférer l'application. Appeler l'enseignant avant de mettre en **RUN** l'automate. Mettre le commutateur sur **mode Manuel** et vérifier que les commandes **Up** et **Down** sur l'axe **J3** sont opérationnelles. Pour cela, agir sur le pupitre opérateur afin de tester les différentes attitudes que peut avoir un opérateur (arrêt avant ou après la fin du mouvement, demande antagonisme).

4.4.5 Compléter l'application afin de prendre en compte toutes les demandes possibles de l'opérateurs (cf.3.3). Afin de détecter un changement de consigne (comparaison du pupitre actuel avec celui qui a provoqué le mouvement en cours), vous établirez une structure **TPupitre** qui contient autant de membres qu'il y a de boutons du pupitre opérateur commandant les 4 axes.

4.4.6 Compiler puis transférer l'application. Vérifier tout d'abord qu'il est possible d'effectuer des mouvements sur chacun des axes en les prenant un par un. Lorsque ces vérifications sont faites, agir sur le pupitre opérateur afin de tester le cas de demandes simultanées sur plusieurs axes.



Annexe 1 :

FUNCTION BLOCK MC_MoveAxisAbsolute

(* This function block commands a joint interpolated movement to an end position. They represent the position of each axis in degrees *)

(* The speed of the robot is calculated with both :

- Parameters related to axes of the robot (Velocity / Acceleration / Deceleration)
- Parameter related to the Tool Center Point (CartesianVel / RotationalVel)

The **final robot speed** is set by the **most limiting of these parameters**. *)

VAR_INPUT

Execute : BOOL ; (* Rising edge triggers function execution *)
JointPosition : T_JointPos ; (* Absolute end position for each axis *)
Velocity : REAL ; (* Joint velocity : % of nominal speed : [0.01,500] *)
Acceleration : REAL ; (* Joint acceleration: % of nominal acceleration [0.01,500] *)
Deceleration : REAL ; (* Joint deceleration: % of nominal deceleration [0.01,500] *)
CartesianVel : REAL ; (* Max. cartesian velocity at Tool Center Point in mm/s *)
RotationalVel : REAL ; (* Max. rotational velocity at Tool Center Point degree/s *)
CoordSystem : UINT ; (* 0 : WORLD, others Coordinate System in a database *)
ToolNumber : UINT ; (* 0 : FLANGE, others Index of the TOOL in a database *)
BufferMode : UINT ; (* cf. type eMC_BUFFER_MODE *)
TransitionMode : UINT ; (* defined the profile of the robot trajectory nearby the set points : 0=None/3=Corner distance/10=blending distances *)
TransitionParam : MC_transitionParameter ;

END VAR

VAR_IN_OUT

AxesGroup : T_StaeubliRobot ; (* Data block for a robot *)

END VAR

VAR OUTPUT

Busy : BOOL ; (* TRUE when function bloc is executing *)
Done : BOOL ; (* TRUE when function bloc has terminated with success *)
Active : BOOL ; (* TRUE when commanded movement is currently executed *)
CommandTransferred : BOOL ; (* TRUE when command motion is successfully buffered *)
CommandAborted : BOOL ; (* TRUE when aborted by another command *)
Error : BOOL ; (* TRUE when function bloc has terminated with error *)
ErrorID : UDINT ; (* error code *)
MovementID : INT ; (* identifier for this commanded movement *)

END VAR

Annexe 2 :

TYPE
eMC_BUFFER_MODE :
 Aborting:=0, (* A FB with buffer mode "Aborting" aborts any ongoing motion and starts the new motion immediately *)
 Buffered:=1, (* The next FB affects the axes group as soon as the previous motion is fully completed (Robot has reached commanded destination) *)
 BlendingJoint:=6, (* The current and the next motion FBs are blended, so the axes group will not stop between the motions. During blending, the robot is performing a joint interpolated motion *)
 BlendingCartesian:=7; (* The current and the next motion FBs are blended, so the axes group will not stop between the motions. The shape of the robot's motion during the blending is a Bezier curve *)

END TYPE

TYPE MC_TransitionParameter :

STRUCT

leave: REAL; (distance from the destination point at which the nominal trajectory is left *)*
reach: REAL; (distance from the destination point at which the nominal trajectory is joined again *)*

END STRUCT

END TYPE

TYPE TMoveAxisAbs

STRUCT

m_xExecute : BOOL ; (Rising edge triggers move execution *)*
m_cartesianPos : T_CartesianPos ; (Absolute end position for each dimension *)*
m_jointPos : T_JointPos ; (Absolute end position for each joint *)*
m_rVelocity : REAL ; (Joint velocity : % of nominal speed *)*
m_rAcceleration : REAL ; (Joint acceleration : % of nominal acceleration *)*
m_rDeceleration : REAL ; (Joint deceleration : % of nominal deceleration *)*
m_rCartesianVel : REAL ; (Max. cartesian velocity at Tool Center Point in mm/s *)*
m_rRotationalVel : REAL ; (Max. rotational velocity at Tool Center Point degree/s *)*
m_uiCoordSystem : UINT ; (Number of the coordinate system (0 by default) *)*
m_uiToolNumber : UINT ; (Index of the tool in the bank (0 by default) *)*
m_uiBufferMode : UINT ; (cf. type eMC_BUFFER_MODE *)*
m_uiTransitionMode : UINT ; (defined the profile of the robot trajectory *)*
m_transitionParam : MC_transitionParameter ;
m_xBusy : BOOL ;
m_xDone : BOOL ;
m_xActive : BOOL ;
m_xCommandTransferred : BOOL ;
m_xCommandAborted : BOOL ;
m_xError : BOOL ;
m_udiErrorID : UDINT ;
m_iMovementID : INT ;

END STRUCT

END TYPE

TYPE T_JointPos

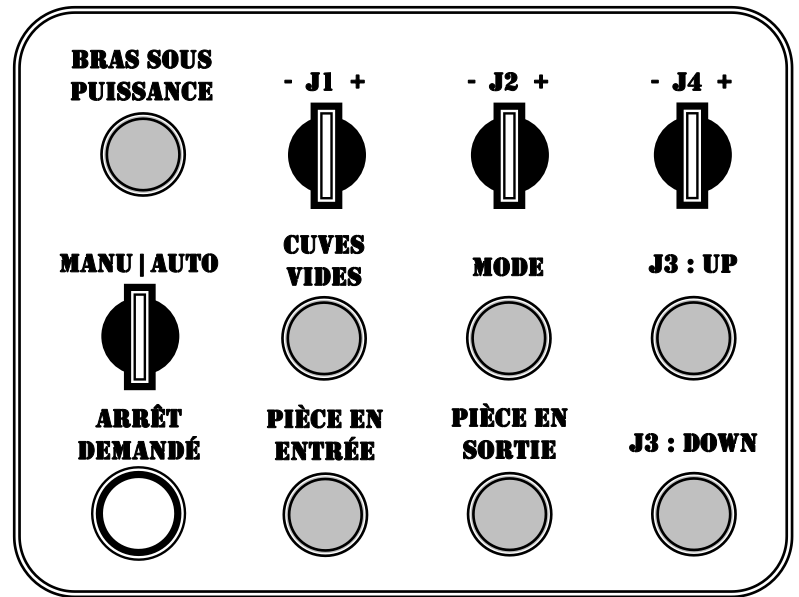
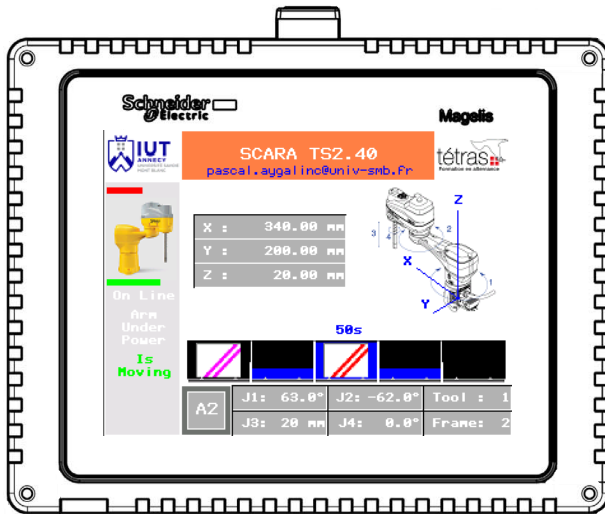
STRUCT

J1 : REAL; (Position of axis 1 *)*
J2 : REAL; (Position of axis 2*)*
J3 : REAL; (Position of axis 3*)*
J4 : REAL; (Position of axis 4*)*
J5 : REAL; (Position of axis 5 if applicable. For 4 axis robot, this value is 0*)*
J6 : REAL; (Position of axis 6 if applicable. For 4 axis robot, this value is 0*)*

END STRUCT

END TYPE

Annexe 3 : Pupitre HMI, Eléments de commandes et Voyants



TYPE THmi

STRUCT (* se reporter au TP n°2 pour connaitre en détail l'un des deux membres de cette structure *)

m_Inputs : THmiInputs ;
m_Outputs : THmiOutputs ;

END_STRUCT

END_TYPE

TYPE THmiInputs

STRUCT

m_ixJ1Plus : BOOL;
m_nixJ1Minus : BOOL;
m_ixJ2Plus : BOOL;
m_nixJ2Minus : BOOL;
m_ixJ3Up : BOOL;
m_nixJ3Down : BOOL;
m_ixJ4Plus : BOOL;
m_nixJ4Minus : BOOL;
m_nixAuto : BOOL;
m_nixManu : BOOL;
m_nixEmptyTanks : BOOL;
m_nixUnloadTank4 : BOOL;
m_nixLoadTank0 : BOOL;
m_nixEmptyTank0 : BOOL;
m_nixEmptyTank4 : BOOL;
m_nixStopRequest : BOOL;

END_STRUCT

END_TYPE

(* NOpen, NClose *)

(* M221

: @*)

(* Switch NO *) %I0
(* Switch NO *) %I1
(* Switch NO *) %I2
(* Switch NO *) %I3
(* Push Button NO *) %I4
(* Push Button NO *) %I5
(* Switch NO *) %I6
(* Switch NO *) %I7
(* Switch NO *) %I8
(* Switch NO *) %I9
(* Push Button NO *) %I10
(* Push Button NO *) %I11
(* Push Button NO *) %I12
(* Push Button NO *) %I13
(* Sensor NC *) %I14
(* Sensor NC *) %I15