Mise en œuvre d'UniVALplc sous Control Expert
Via Ethernet/IP (Partie 3)

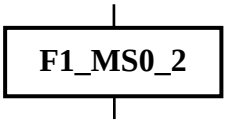
Objectifs visés :
- ✓ ***Extensions du langage SFC (multi-jetons, macro-modélisation)***
- ✓ ***Séquencement des mouvements par rapport à l'automatisme***
- ✓ ***Transformation géométrique (T_Trsf, VAL_ShiftPoint)***

0. Préambule : langage SFC & extensions de ce langage sous Control Expert
0.1 Macro-Modélisation

La notion de ***macro-étape*** n'existe pas ***nativement*** en langage ***SFC***. Sous ***Control Expert***, cette notion fait partie des extensions de ce langage et doit être explicitement demandée lors de la ***configuration du projet*** *(menu :* ***Outils*** *->* ***Options du projet*** *: rubrique* ***SFC*** *dans Programme puis Langage : «* ***Autoriser les sections de macros*** *»).*

Comme pour chaque ***étape*** d'une section ***SFC***, un ***objet*** du type ***SFCSTEP_STATE*** est systématiquement créé pour les ***étapes*** de l'expansion ainsi que pour la macro-étape (*le* ***nom*** *de cet objet est celui de la macro-étape*). On peut aussi associer à chaque étape de l'expansion un ***objet*** du type ***SFCSTEP_TIMES*** si on désire ***activer un contrôle temporel*** pour faciliter la mise au point, la maintenance, le diagnostic et la détection de défauts.

On adoptera les règles suivantes pour établir les ***noms des macro-étapes***, des ***étapes*** et des ***sections de transition*** apparaissant dans ***leur expansion***.

✓ Pour désigner une macro- étape : 	***XX_MSI*** avec : • ***XX :*** le nom du mode de fonctionnement qui y fait référence *(exemple : Mode* ***F1*** *=>* ***F1_MSI****).* • ***MS : M****acro-****S****tep (macro-étape)* • ***I :*** *numéro de la macro-étape dans le mode.*
✓ Pour désigner une étape de la macro- étape : 	***XX_MSI_J*** avec : • ***XX :*** le nom du mode de fonctionnement qui y fait référence *(exemple : Mode* ***F1*** *=>* ***F1_MS0_2****).* • ***I :*** numéro de la macro-étape dans le mode. • ***J :*** numéro de l'étape dans l'expansion de la macro-étape.
✓ Préfixe d'une section de transition de la macro-étape ***XX_MSI***:	***XX_MSI_T_*** *(exemple F1_MS0_T_MiseEnService)*

Nota : - *Si la macro-étape a pour nom* ***XX_MSI****, son* ***étape d'entrée*** *(respectivement* ***de sortie****) a pour nom* ***XX_MSI_IN*** *(resp.* ***XX_MSI_OUT****).*

0.2 Fonctionnement Multi-jetons

Nativement, un graphe ***SFC*** doit comporter qu'une seule étape initiale. ***Control Expert*** offre la possibilité d'en avoir plusieurs sur les graphes ***SFC*** si le fonctionnement ***multi-jetons*** est explicitement demandé dans les options du projet *(menu* ***Outils*** *->* ***Options du projet*** *: rubrique* ***SFC*** *«* ***Autoriser plusieurs jetons*** *»).*

1. *Rappel sur la structure de commande adoptée*

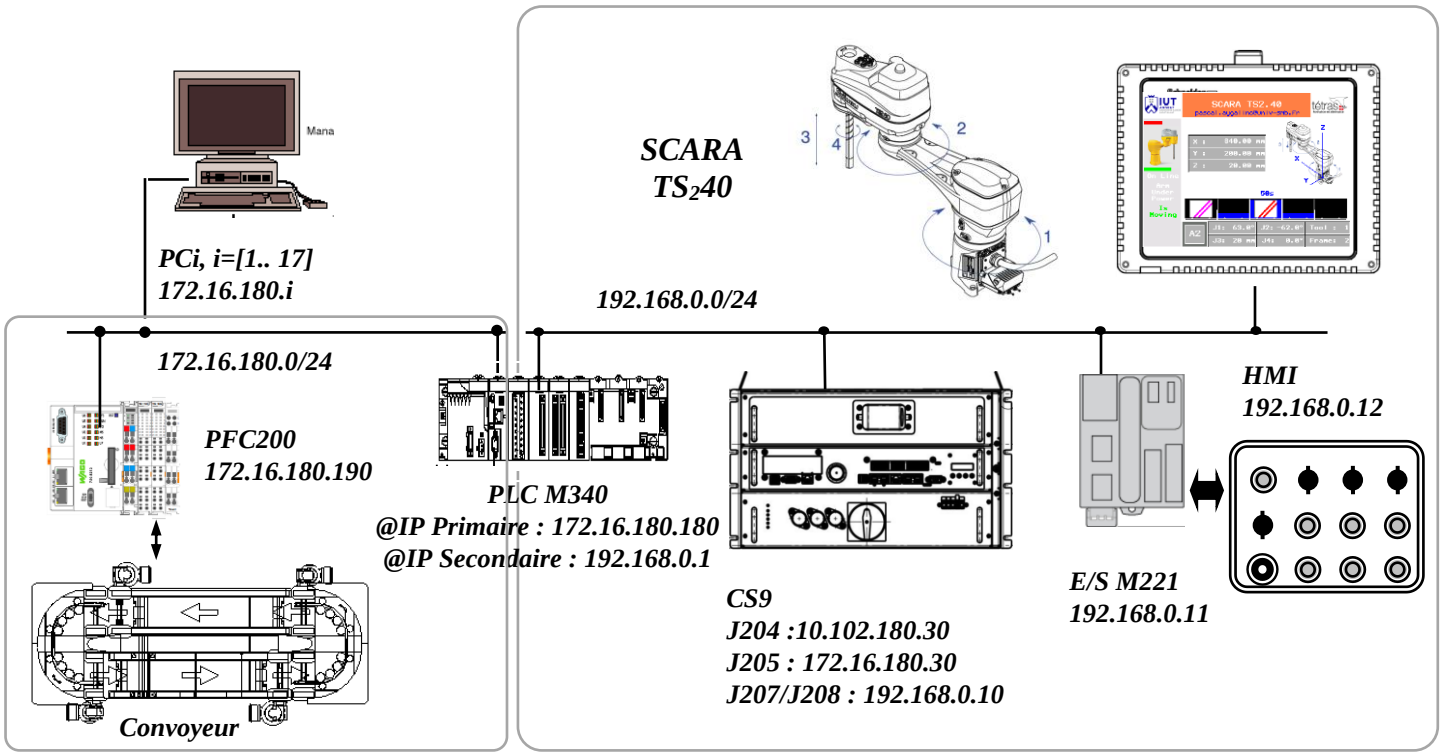


***Figure 1** : Cellule robotisée de la salle C180 : (**PLC M340, CS9, M221, HMI et PFC200**)*

Au niveau matériel, on dispose :

- ✓ *d'un robot **Stäubli Scara TS$_2$40 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 privatif 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*** précédant, 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****)*. Les coordonnées cartésiennes dans le repère *«* ***world*** *»* du robot sont *: (X=90, Y=-410, Z=150, RX=-180, RY=0, RZ=0).* Pour le guichet, le flux des palettes est passant et seule la composante verte de la colonne est allumée.

Ce ***tp*** portera sur l'utilisation conjointe du ***robot*** et du poste ***Guichet*** pour établir le mode de fonctionnement de production normale de la cellule. Il s'agira donc de ***coordonner les mouvements du robot*** en fonction de ***l'occupation du guichet***.

1. *Mode F1 : Production Normale*

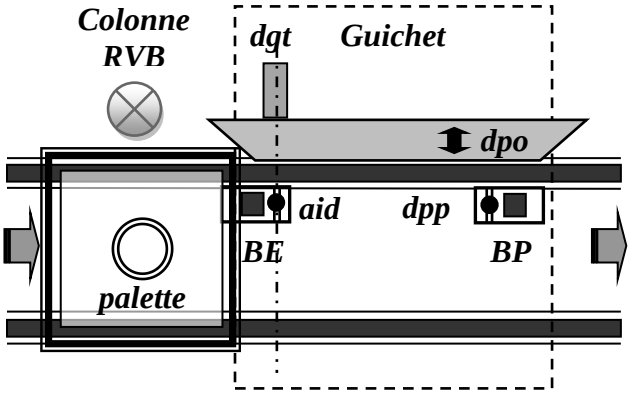


Figure 2 : Le guichet

Pour ce mode, on souhaite à l'aide du ***robot***, ***transférer*** le ***gobelet*** d'une palette pleine sur la ***prochaine palette*** détectée ***vide*** dans le guichet.

Le ***contrôle du flux des palettes*** établi à l'aide des ***bloqueurs BE*** et ***BP*** doit rendre la ***capacité*** du guichet ***unitaire*** *(identique au fonctionnement de celui d'un* ***sas*** *: pour qu'une palette puisse entrer dans le guichet, celui-ci doit être libre).* Il doit permettre aussi la ***saisie*** et la ***dépose des gobelets*** sur une ***palette à l'arrêt*** dans le poste *(le capteur* ***dpp*** *sera alors actif).*

On rappelle que le ***fonctionnement en isolation*** du poste ***Guichet*** a fait l'objet du premier ***tp***. Sur la figure 3 , la ***partie grisée*** du modèle décrit ***uniquement le fonctionnement du poste*** qui a été complétée par la partie non grisée pour prendre en compte l'***animation de la colonne lumineuse***.

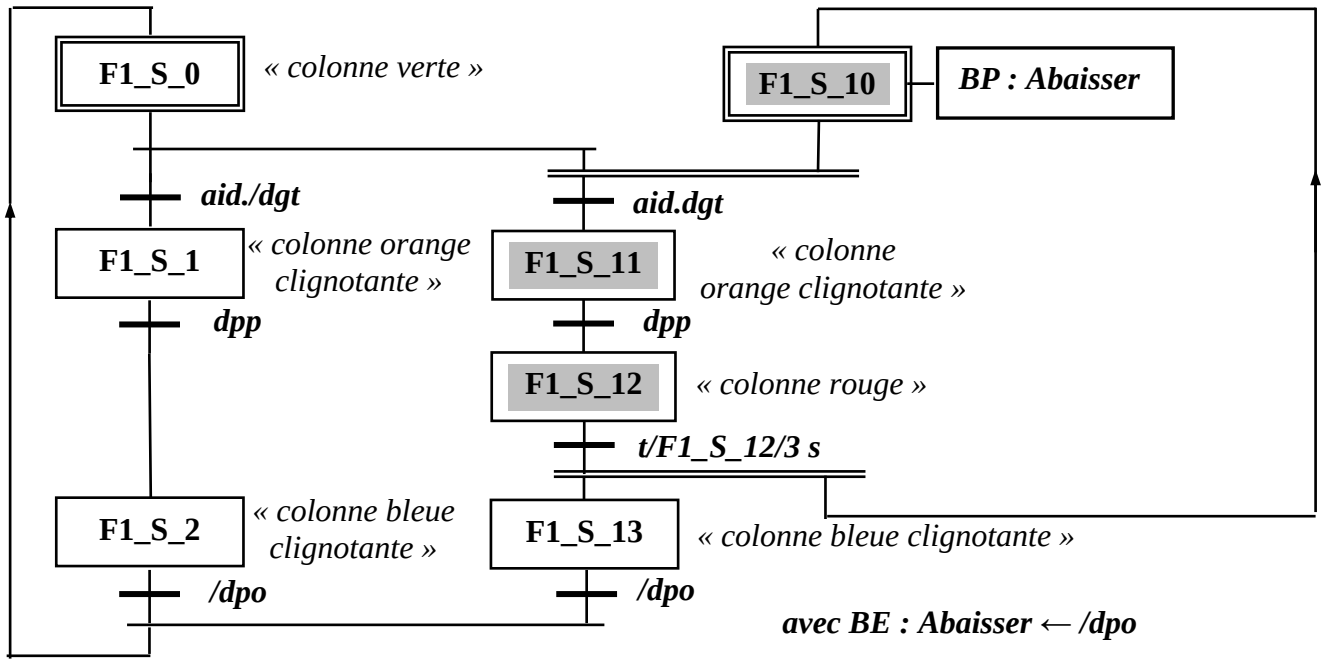


Figure 3 *: grafcet fonctionnel du poste guichet en isolation*

*1.1 A partir du modèle en isolation, établir le **grafcet fonctionnel** associé au nouveau fonctionnement demandé. On fera appel à la notion de **macro-étape** afin d'en limiter les détails. On désignera par **F1_MS1** la **macro étape** associée aux mouvements que doit accomplir le robot pour effectuer la saisie des gobelets et par **F1_MS2** la **macro étape** à ceux associés à la dépose. On rappelle que le distributeur pneumatique du venturi est bistable afin de faciliter la modélisation.*

*1.2 Donner le corps des expansions des deux macro-étapes **F1_MS1** et **F1_MS2**. Pour les **déplacements**, on prendra des commandes de mouvements avec **une interpolation linéaire**. Seule la position sur l'axe Z (axe J3) est à modifier. En effet, en **A1**, afin que le robot n'entrave pas la circulation des palettes, ses coordonnées sont définies par (X=90, Y=-410, **Z=150**, RX=-180, RY=0, RZ=0) alors que pour la dépose ou la saisie, elles deviennent : (X=90, Y=-410, **Z=16**, RX=-180, RY=0, RZ=0).*

***Nota** : Le calcul des positions de saisie et de dépose peut s'effectuer en relatif par rapport à la position initiale à l'aide d'une structure de transformation géométrique **T_Trsf** initialisée à (X=0, Y=0, **Z=-134**, RX=0, RY=0, RZ=0)) et d'une instance du bloc fonctionnel **VAL_ShiftPoint** (cf. Annexe 1).*

*1.3 L'implantation de cet automatisme s'effectuera suivant la structure d'une machine de **MEALY**.*

*1.3.1 Pour la section en langage **SFC**, la traduction de votre grafcet sera directe si les structures de choix et de convergence de choix ne s'effectuent que sur des transitions simples. Si ce n'est pas le cas, vous serez amené à utiliser les règles de traduction énoncées en cours.*

*1.3.2 Pour la partie postérieure au graphe écrite en langage **ST**, elle aura pour rôle d'établir la commande des bloqueurs et de la colonne en fonction de bits d'activités des étapes et d'éventuelles sécurités secondaires. De plus, c'est dans cette partie que l'instance du bloc fonctionnel associée aux commandes de mouvement devra être scrutée, ainsi que celle autorisant un calcul relatif pour les positions de saisie et de pose.*

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*** précédant, les rectangles ***F1*** et ***A2*** ont été retenus. Ils sont associés au fonctionnement décrit précédemment *(cf. 1)*.

2.1 Renseigner les conditions de passage ***GM_T_A1toF1***, ***GM_T_F1toA2*** *et* ***GM_A2toA1***. Il est clair que tout ***transfert de gobelet débuté*** devra être ***terminé*** avant de regagner le mode ***A1***.

Partie TP

Le 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***.

*3.1 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.*

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

*3.3 **Modifier les options du projet** pour autoriser les **extensions du langage SFC** (menu **Outils** -> **Options du projet** : rubrique **SFC**). Choisir :*

- ✓ « ***Autoriser plusieurs jetons*** » (***multi-jetons***) *afin d'avoir plusieurs étapes initiales dans un* ***SFC****,*
- ✓ « ***Autoriser les sections de macros*** » *afin d'autoriser l'utilisation des macro-étapes.*

3.4 Programmation du ***Gmma***

3.4.1 Section ***GM_Gmma*** *:*

- ✓ *Compléter le* ***gmma*** *afin de faire apparaitre deux étapes supplémentaires :* ***GM_S_F1*** *et* ***GM_S_A2****.*
- ✓ *Puis renseigner les transitions* ***GM_T_A1toF1****,* ***GM_T_F1toA2*** *et* ***GM_A2toA1*** *conformément à votre analyse (cf. 2.1).*

3.4.2 Section ***GM_GmmaPost***

- ✓ *Positionner le bit* ***xF1A2*** *qui autorise la scrutation des sections* ***F1_ProductionNormale*** *et* ***F1_ProductionNormalePost*** *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 les états* ***F1*** *et* ***A2****.*

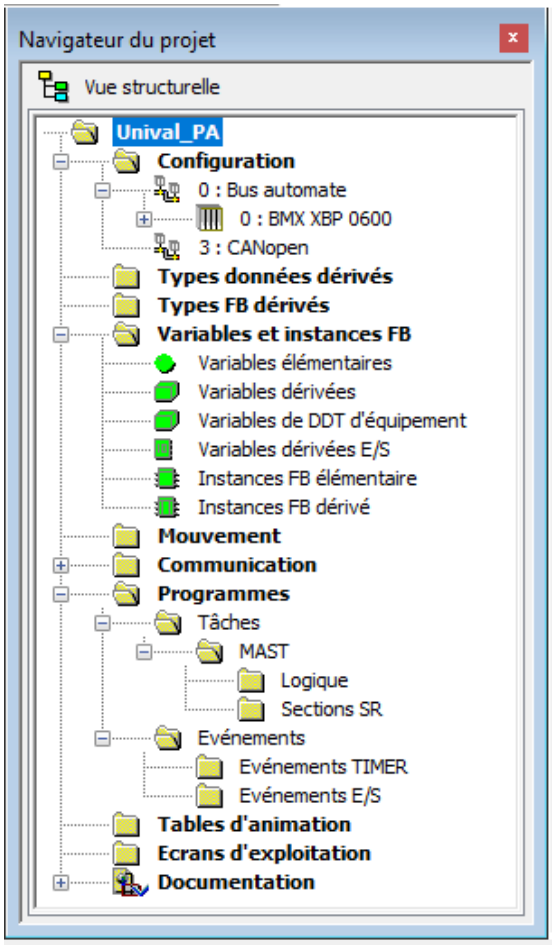


Figure 2

3.5 Programmation du mode ***F1 : Production Normale***

3.5.1 Section ***F1_ProductionNormale*** *en langage* ***SFC*** *:*

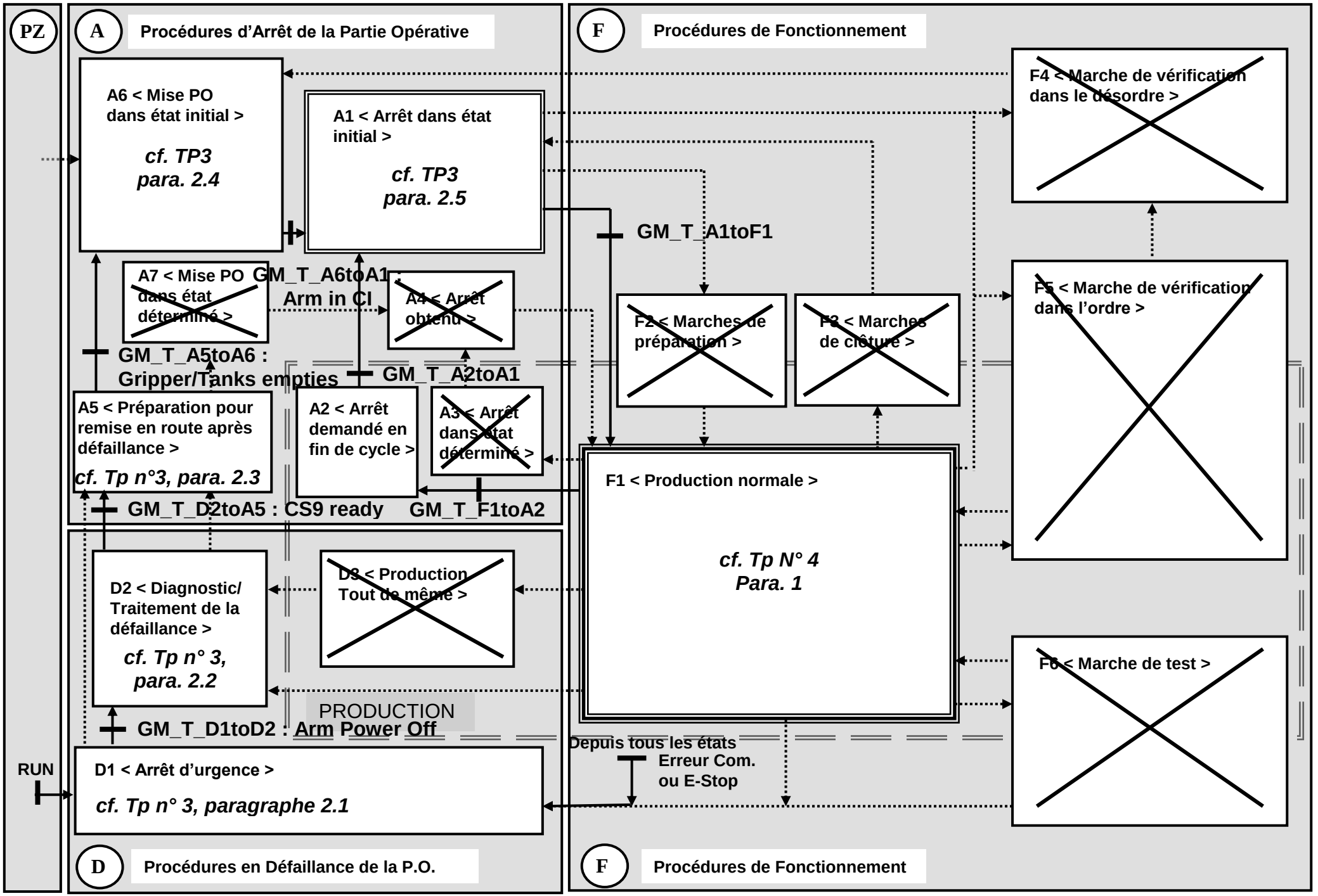
- ✓ *Créer une instance du bloc fonctionnel* ***MC_MoveLinearAbsolute*** ***(InstMoveLinearAbsoluteF1)***
- ✓ *Créer une donnée* ***moveAxisLinearAbsF1*** *du type* ***TMoveAbsolute*** *qui facilitera l'appel de l'instance précédente.*
- ✓ *Modifier le graphe afin qu'il corresponde à celui de votre étude (cf. 1.3.1)*
- ✓ *Compléter le corps des deux macro étapes* ***F1_MS1*** *et* ***F1_MS2****. Vous utiliserez les actions associées aux étapes de type impulsionnelle* ***P1*** *pour modifier le point de destination à atteindre, et définir les caractéristiques et propriétés des déplacements (descripteur de mouvement):*

  - *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***
  - *comme on n'utilise pas la base de données de la baie* ***CS9****, le système de coordonnées sera «* ***world*** *»* (***CoordSystem****:=0*) *et l'indice d'outil sera la bride «* ***flange*** *»* (***ToolNumber****:=0*).

3.5.2 Section ***F1_ProductionNormalePost*** *en langage* ***ST*** *:*

- ✓ *Modifier les équations des actionneurs du guichet (la colonne et les bloqueurs)*
- ✓ *Faire scruter l'instance du bloc fonction* ***MC_MoveLinearAbsolute*** *en utilisant l'objet* ***moveAxisLinearAbsF1*** *pour le passage des paramètres.*
- ✓ *Si vous faites appel à un calcul en relatif pour définir les positions de saisie et de dépose, déclarer un objet* ***trsfPosF1*** *de type* ***T_Trsf*** *initialisée à (X=0, Y=0,* ***Z=-134****, RX=0, RY=0, RZ=0) et une instance du bloc fonctionnel* ***VAL_ShiftPoint*** *qui devra être scrutée aussi dans cette section.*

*3.5.3 Compiler puis transférer l'application. Appeler l'enseignant avant de mettre en **RUN** l'automate. Sur l'écran doit apparaitre quand le robot est en mouvement une jauge verte proportionnelle au pourcentage déjà effectué du mouvement à réaliser ainsi que l'actualisation des coordonnées du robot.*


PZ
A
Procédures d'Arrêt de la Partie Opérative
F
Procédures de Fonctionnement
A6 < Mise PO dans état initial >
cf. TP3 para. 2.4
A1 < Arrêt dans état initial >
cf. TP3 para. 2.5
GM_T_A1toF1
F4 < Marche de vérification dans le désordre >
A7 < Mise PO dans état déterminé >
GM_T_A6toA1 : Arm in CI
A4 < Arrêt obtenu >
F2 < Marches de préparation >
F3 < Marches de clôture >
F5 < Marche de vérification dans l'ordre >
GM_T_A5toA6 : Gripper/Tanks empties
GM_T_A2toA1
A5 < Préparation pour remise en route après défaillance >
cf. Tp n°3, para. 2.3
A2 < Arrêt demandé en fin de cycle >
A3 < Arrêt dans état déterminé >
F1 < Production normale >
cf. Tp N° 4 Para. 1
GM_T_D2toA5 : CS9 ready
GM_T_F1toA2
D2 < Diagnostic/ Traitement de la défaillance >
cf. Tp n° 3, para. 2.2
D3 < Production Tout de même >
F6 < Marche de test >
PRODUCTION
GM_T_D1toD2 : Arm Power Off
Depuis tous les états
Erreur Com.
ou E-Stop
RUN
D1 < Arrêt d'urgence >
cf. Tp n° 3, paragraphe 2.1
D
Procédures en Défaillance de la P.O.
F
Procédures de Fonctionnement

Annexe 1 :

FUNCTION_BLOCK ***MC_MoveLinearAbsolute***

(This function block commands a linear interpolated movement from the current robot position to the specified absolute position in the specified coordinate system. The values specified for the end-position are interpreted as* ***cartesian values****. 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 *)*
Position	*:* ***T_CartesianPos*** *;*	*(* Absolute end position for each dimension *)*
Velocity	*: REAL ;*	*(* Joint velocity : % of nominal speed : [0.01,500]*)*
Acceleration	*: REAL ;*	*(* Joint accelaration:% of nominal acceleration [0.01,500] *)*
Deceleration	*: REAL ;*	*(* Joint decelaration:% 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

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 TMoveAbsolute
    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 *)
        m_uiToolNumber      : UINT ;        (* Index of the tool in the bank *)
        m_uiBufferMode      : UINT ;        (* cf. type eMC_BUFFER_MODE *)
        m_uiTransitionMode  : UINT ;        (* cf. type eMC_TRANSITION_MODE *)
        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_Trsf : (* This data type represents a geometrical transformation *)
    STRUCT
        X       : REAL;     (* X coordinate *)
        Y       : REAL;     (* Y coordinate *)
        Z       : REAL;     (* Z coordinate *)
        RX      : REAL;     (* RX coordinate *)
        RY      : REAL;     (* RY coordinate *)
        RZ      : REAL;     (* RZ coordinate *)
    END STRUCT
END TYPE
```

```
FUNCTION_BLOCK VAL_ShiftPoint
    (* This function block computes a new cartesian position by applying a geometrical transformation.
    The geometrical transformation specified by 'Offsets' parameter can be applied relative to different
    coordinate systems:
        User frames (equivalent to native compose() function)
        Tool (equivalent to native appro() function ) *)
    VAR_INPUT
        Execute      :BOOL ;          (* Rising edge triggers function execution *)
        CoordSystem  : INT ;          (* 0 : WORLD, others Coordinate System in a database *)
        Position     : T_CartesianPos ;  (* Absolute end position for each dimension *)
        ShiftInFrame : BOOL ;         (* True=Position will be shifted in specified user frame.
                                      False=shift in according to orientation of the Position parameter *)
        FrameIdx     : INT ;          (* Index of the user frame in which Position will be shifted
                                      Applicable only if ShiftInFrame is true *)
        Offsets      : T_Trsf ;       (* Offsets in all directions to apply to the Position *)
    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 *)
        Error             : BOOL ;    (* TRUE when function bloc has terminated with error *)
        ErrorID           : UDINT ;   (* error code *),
        RelativePosition  : T_CartesianPos ;  (* New Position after the offets have been applied *)
    END VAR

TYPE T_Trsf : (* This data type represents a geometrical transformation *)
    STRUCT
        X   : REAL;   (* X coordinate *)
        Y   : REAL;   (* Y coordinate *)
        Z   : REAL;   (* Z coordinate *)
        RX  : REAL;   (* RX coordinate *)
        RY  : REAL;   (* RY coordinate *)
        RZ  : REAL;   (* RZ coordinate *)
    END STRUCT
END TYPE
```