Mise en œuvre d'UniVALplc sous Ethernet/IP

Objectifs visés :

- ✓ ***Principes de base de la solution UniVALplc du constructeur Stäubli***
- ✓ ***Service IO-Scanning sous Ethernet/IP***
- ✓ ***Accès à la baie CS9 et aux E/S déportées réalisées à l'aide d'un contrôleur M221***
- ✓ ***Mise en œuvre de Blocs Fonctionnels***

0. Préambule

Différentes ***structures de commande*** sont maintenant ***disponibles*** quand il s'agit de faire intervenir ***des robots dans les systèmes de production***. Le ***choix*** de la structure dépend du ***savoir-faire de l'intégrateur*** mais aussi de l'***environnement dans lequel le robot s'inscrit***. En effet, les ***contrôleurs de robots*** modernes, qu'ils soient ***collaboratifs ou non***, proposent des interfaces ***TOR*** *et* ***ANAlogiques*** à l'aide de cartes additionnelles. Il en est de même pour les ports de communication allant des ***réseaux de terrain classiques*** comme *CanOpen, Profibus DP*, à des réseaux de type ***Ethernet industriel déterministe*** *(EtherCAT, Ethernet/IP, PowerLink, Profinet, ...)*. Pour ces ***derniers***, les ***contrôleurs*** de robot peuvent être configurés soit en ***esclave***, soit en ***maitre***.

Comme ***l'objectif*** de ce module est ***d'utiliser un automate programmable*** pour établir ***la commande d'un robot***, l'intégration du ***robot*** dans notre cellule *(cf. figure 1)* s'apparente à celle d'un ***objet industriel connecté***. Le ***contrôleur*** du robot conserve l'aspect « ***motion*** ». Son ***interpolateur*** est sollicité pour effectuer les déplacements, mais le ***séquencement*** de ces derniers est dicté par l'***automate***. La solution ***UniVALplc*** développée par ***Stäubli*** s'inscrit dans ce cadre et fait l'objet de ce tp et de ceux qui vont suivre.

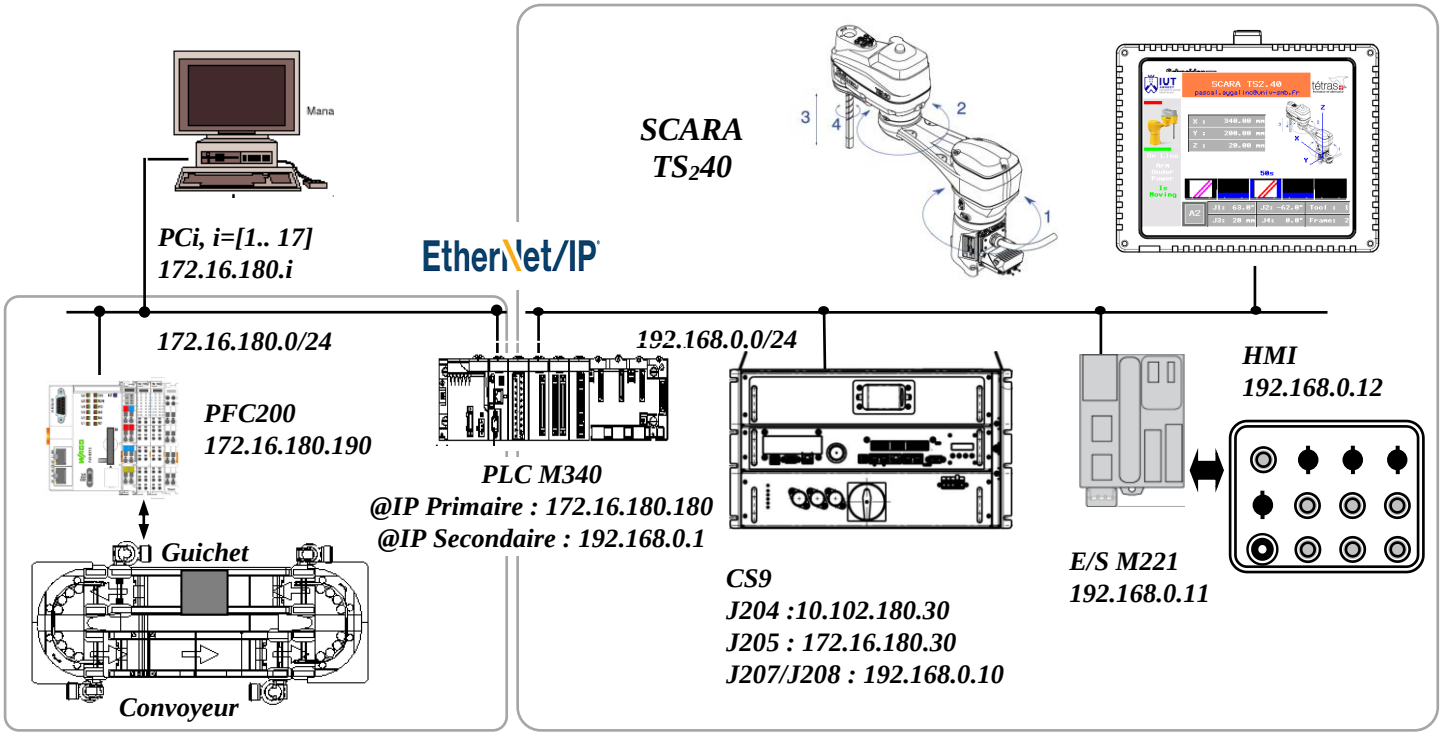


Figure 1. Cellule robotisée de la salle C180 : ***(PLC M340, CS9, M221, HMI et PFC200)***

On rappelle qu'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***,

Nota : *Il est possible de retirer les équipements* ***MCP*** *et* ***WMS9*** *en effectuant une configuration logicielle particulière du contrôleur* ***CS9*** *et de ses connecteurs* ***J101*** *et* ***J103*** *(par exemple, pour* ***J101****, le E-Stop contacts (5-7 / 6-8) sont à relier). L'intégrateur peut ainsi adopter une interface* ***HMI*** *de son choix à la place du pendant d'origine. Ici, le pendant a été conservé afin de vous permettre de vérifier l'exactitude des informations que vous afficherez sur l'interface* ***HMI*** *proposée.*

Le premier ***tp*** a porté sur la configuration du réseau ***Ethernet/IP*** et sur la communication à réaliser entre ***l'API M340*** et le contrôleur ***PFC200***. Elle a permis d'établir une première ébauche du mode de production normale (***mode F1***) du poste ***Guichet*** en isolation situé sur le convoyeur.

Ce ***tp*** s'intéresse maintenant à la configuration du ***2ème réseau Ethernet/IP*** *(192.168.0.0/24).* Il doit permettre la communication entre ***l'API M340*** avec la baie ***CS9***, ainsi qu'avec le contrôleur ***M221.***

Sur le contrôleur ***CS9*** du robot, un serveur ***uniVALplc*** est installé et répond à des commandes en provenance de l'automate ***M340*** qui assure l'automatisation de la cellule. Ces commandes sont émises à l'aide de blocs fonctionnels via le réseau ***Ethernet/IP***. Cet ***ensemble de blocs fonctionnels*** constitue ***la bibliothèque client uniVALplc*** et ils doivent être importés dans l'outil de développement fourni par le fabricant de l'***API***, ici ***Control Expert***. Certains de ces blocs sont spécifiques aux contrôleurs ***Stäubli*** *(préfixe* ***VAL_****)*, d'autres répondent à la norme « ***plcOpen »*** (*cf.* ***https://plcopen.org***) quand il s'agit de commander des mouvements *(préfixe* ***MC_****)*,

Au niveau du serveur, les commandes sont interprétées pour établir les mouvements demandés et ils peuvent faire référence à des objets contenus dans des bases de données internes au contrôleur ***CS9*** *(points, outils, frames, ...)*.

Nota : *L'installation du serveur* ***uniVALplc*** *est* ***dépendante*** *de* ***l'orientation des données*** *de* ***l'automate client****. Pour notre part, comme il s'agit d'un* ***API M340*** *du constructeur* ***Schneider****, c'est l'option* ***Little Endian*** *qui a été fournie lors de l'installation.*
Par ailleurs, pour que la baie ***CS9*** *accepte les commandes d'un client* ***API****, le sélecteur de modes de marche* ***WMS9****, s'il est présent, doit être sur «* ***Control Distant*** *».*

Le fonctionnement de base de la solution ***uniVALplc*** nécessite la ***réservation*** d'un objet du type ***T_StaeubliRobot*** pour représenter ***le robot*** *et* ***son contrôleur***. Cet objet contient trois champs : le champ ***Status*** de type ***T_Status***, le champs ***Command*** de type ***T_Command*** et le champ ***CommInterface*** de type ***T_CommInterface*** *(cf. annexe 1 pour les détails)*. Il est « ***primordial »*** car il est utilisé comme ***paramètre d'entrée/sortie*** par la plupart des blocs fonctionnels de la bibliothèque client ***uniVALplc***. Ainsi, à chaque cycle ***API***, le ***traitement utilisateur*** doit réaliser les fonctions suivantes :

i. tout d'abord, s'informer de l'état du robot et de son contrôleur à partir d'une mémoire image (***T->O :*** *données produites*) qui est actualisée via un service de communication périodique du réseau ***Ethernet/IP***. Elle sera recopiée au moyen d'un bloc fonctionnel dédié ***(VAL_ReadAxesGroup)*** dans l'objet du type ***T_StaeubliRobot*** pour maintenir à jour ses champs ***Status*** et ***CommInterface***,

ii. puis, établir à partir des blocs fonctionnels de la bibliothèque client ***uniVALplc*** les opérations que le robot doit effectuer par rapport au séquencement souhaité. Ces blocs utilisent l'objet du type ***T_StaeubliRobot*** actualisé précédemment afin d'en connaitre son état *(champs* ***Status*** *et* ***CommInterface****)* et d'en modifier la partie commande et communication *(champs* ***Command*** *et* ***CommInterface****),*

iii. enfin, écrire ces ordres à partir de l'objet modifié du type ***T_StaeubliRobot*** au moyen d'un bloc fonctionnel dédié ***(VAL_WriteAxesGroup)*** dans une autre mémoire image (***O->T :*** *données consommées*) de l'automate qui sera transmise aussi de façon périodique au contrôleur ***CS9*** via le réseau ***Ethernet/IP***.

Pour ***l'accès aux E/S*** du contrôleur ***M221***, le même principe sera appliqué : on effectuera une lecture des ***entrées déportées*** en début de cycle ***API*** à partir d'une mémoire image actualisée par un ***service d'assemblage***, et la mise à jour des ***sorties déportées*** se fera en fin de cycle dans une autre mémoire image obtenue par ce service.

Sous ***Ethernet/IP***, la ***configuration des services accessibles*** pour ces ***2 équipements*** (***CS9, M221***) s'effectue à l'aide des fichiers d'extension ***eds*** (***EDS : E****lectronic* ***D****ata* ***S****heet)* suivants :

- ✓ Pour le serveur ***uniVALplc*** installé sur un contrôleur ***CS9***, la configuration est établie à l'aide du fichier : ***Staeubli-CS9-uniVALplc-s4.6.eds***.
- ✓ Pour le ***service d'assemblage*** sous ***Ethernet/IP*** activé sur le contrôleur ***M221***, la configuration est donnée par le fichier ***M221-Scara.eds*** *établi à partir du fichier générique* ***M221_EDS_Model.eds***.

Nota : *Pour le* ***contrôleur M221****, le fichier «* ***Instruction for EDS File Modification.pdf*** *» fournit la* ***procédure*** *qui permet de modifier le fichier générique* ***M221_EDS_Model.eds*** *pour l'adapter à son application (cf. U:/Documents/Licence Pro/RIF/UE2/Automatisme pour la robotique/Scara/M221). Ceci évite de le faire sous* ***Control Expert mais surtout de conserver la trace*** *de la configuration utilisée.*

Partie TD

1. Cartographie associée au 2ème réseau Ethernet/IP

Lors du ***Tp n°1****, la configuration suivante a été établie pour le 2ème coupleur* ***BMX NOC 0401.2 (NOC_ETHIP_ROBOT)****. Son image mémoire suit celle associée au premier coupleur, elle débute donc à partir du mot* ***%MW64*** *et comprend* ***128 mots*** *en entrée et* ***128 mots*** *en sortie.*

*Les données produites et consommées par les 2 équipements (****baie CS9*** *et* ***contrôleur M221****) vous sont données par les tableaux suivants :*

Equipement	***Entrée T -> O (Données Produites)***	***Sortie O->T (Données consommées)***
CS9 (UniVALplc serveur)	***148*** *octets*	***124*** *octets*
M221(Ilot E/S, Hmi)	***4*** *octets*	***40*** *octets*

L'ajout de ces équipements sur le réseau ***NOC_ETHIP_ROBOT*** *s'effectuera dans l'ordre mentionné par ce tableau.*

1.1 A partir de ces informations et celles fournies sur l'organisation mémoire des coupleurs ***BMX NOC 0401.2*** *(**cf. Tp n°1, paragraphe 2.1**), établir la cartographie de l'espace mémoire %**MW** de **l'API**.*

1.2 Le bloc fonctionnel ***VAL_ReadAxesGroup*** *réclame que* ***les données produites*** *soient du type* ***T_FromRobot*** *pour pouvoir* ***les convertir de façon algorithmique en un objet de type T_StaeubliRobot****, alors que pour le bloc fonctionnel* ***VAL_WriteAxesGroup*** *qui permet de fabriquer* ***les données à consommer*** *à partir d'un objet* ***de type T_StaeubliRobot****, il impose qu'elles soient du type* ***T_ToRobot*** *(cf. annexe pour connaitre le contexte des deux blocs).*

En utilisant la cartographie établie précédemment et ***les techniques de chevauchement*** *et* ***d'alias***, déclarer deux objets ***fromRobotScara*** de type ***T_FromRobot*** *et* ***toRobotScara*** de type ***T_ToRobot*** afin qu'ils représentent respectivement ***les données produites*** et ***les données consommées*** de la baie ***CS9***.

1.3 Pour le ***M221****, vous disposez de deux blocs fonctionnels* ***FB_ReadHmi*** *et* ***FB_WriteHmi*** *(cf annexe).* ***Le premier*** *utilise* ***les données produites*** *de l'îlot de type* ***TFromHmi*** *pour établir un* ***objet de type Hmi****, alors que le* ***second*** *transforme* ***l'objet de type Hmi*** *en* ***données à consommer de*** type ***TToHmi.***

En utilisant toujours la cartographie établie précédemment et ***les techniques de chevauchement*** *et* ***d'alias***, déclarer deux objets ***fromHmiScara*** de type ***TFromHmi*** *et* ***toHmiScara*** de type ***TToHmi*** afin qu'ils représentent respectivement ***les données produites*** et ***les données consommées*** du contrôleur ***M221***.

2. 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***.

2.1 Travail préparatoire : Importation de fichiers ***eds*** *et* ***Mise à jour de la bibliothèque de types***

Comme pour l'importation de ***fichiers EDS***, l'ajout de ***la bibliothèque client uniVALplc*** à ***Control Expert*** réclame des ***droits d'administrateur sur les PCs***. Ces ***travaux ont donc été réalisés*** après l'installation de ce logiciel. Pour les fichiers ***eds***, la procédure a été décrite lors du tp précédent. On ne regardera ici que la procédure pour la bibliothèque de types.

- ✓ Pour ajouter la bibliothèque client ***uniVALplc*** à ***Control Expert***, il faut exécuter ***en tant qu'administrateur*** *le logiciel* ***Mise à jour de la bibliothèque de types*** *associé à cette chaine de développement et l'importer à l'aide du fichier* ***FAMILY.DCS*** *décrivant cette bibliothèque. (cf. U:/Documents/Licence Pro/RIF/UE2/Automatisme pour la robotique/Scara/uniVALplc/Librairie Unity version4.5.2.*

2.2 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.*

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

2.4 Options du projet : Menu : ***Outils->Options du projet*** pour les configurer :

- ✓ Rubrique ***Général*** *->Gestion des messages de génération* : *le chevauchement d'adresses ne génère aucun message.*
- ✓ Rubrique ***Variables*** : *Autoriser les tableaux dynamiques (ANY_ARRAY_XXX)*

*2.5 Ouvrir les différents dossiers du **Navigateur du Projet** comme le montre la figure 2. On prendra pour habitude de sauver l'application régulièrement, au minimum après chaque rubrique.*

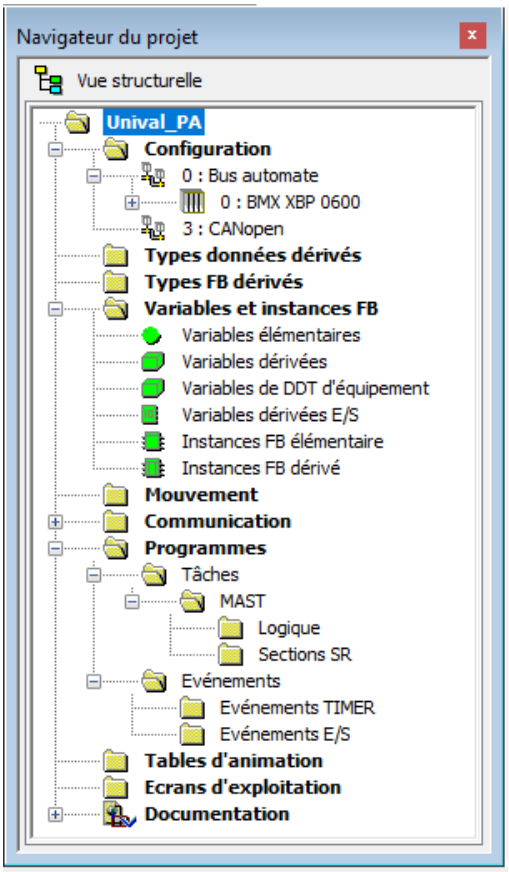


Figure 2

*2.6 Configuration des réseaux **NOC_ETHIP_ROBOT***

*2.6.1 Configuration des **adresses IP** des interfaces **BMX NOC 0401.2***

- ✓ Ouvrir le ***navigateur de DTM*** : Menu : ***Outils->Navigateur de DTM.*** *Double cliquer sur le réseau pour obtenir sa fenêtre de configuration.*

- ✓ *réseau **NOC_ETHIP_ROBOT** : Vérifier que la configuration faite lors du tp précédent est : **rubrique TCP/IP : @IP : 192.168.0.1**, **masque** : 255.255.255.0, **pas de passerelle** (0.0.0.0).*

*2.6.2 **Ajout et configuration des équipements***

- ✓ ***Attention : L'ajout des équipements doit se faire dans cet ordre** sinon le plan d'adressage établi précédemment ne sera pas valide*

- ✓ *réseau **NOC_ETHIP_ ROBOT** :*

- Via un clique droit sur le réseau ***NOC_ETHIP_ ROBOT***, sélectionner l'option ***Ajouter***.
- Dans la liste, l'équipement correspondant à la baie ***CS9*** a pour nom : ***uniVALplc v4.6 – CS9 Adapter Revision 1.1 (from EDS)***. Le sélectionner puis le valider en cliquant sur le bouton « ***Ajouter DTM*** ». Pour la gestion des noms de ***DTMs***, prendre : ***cs9Scara***.

- Recommencer cette opération pour ajouter l'équipement correspondant au ***M221*** dont le nom est donné par : ***TM221_for_Scara (payga) Revision 1.12 (from EDS)***. Pour la gestion des noms de ***DTMs***, prendre : ***m221Scara***.

- Sur le ***navigateur de DTM*** :
  Sélectionner l'équipement ***cs9Scara*** et ouvrir sa configuration via un double clique,
  Adapter le service « ***Exclusive Owner*** » à notre application :
  - Pour le paramètre ***RPI***, prendre une période compatible avec la périodicité de la tâche ***Mast***.

  Appuyer sur le bouton « ***Appliquer*** » pour valider vos saisies et fermer la fenêtre.

  Pour l'équipement ***m221Scara,*** configurer les services d'assemblage *(en entrée et en sortie)* que doit réaliser cet équipement. Pour cela :
  - Ouvrir sa fenêtre de configuration à l'aide d'un double clique sur son nom ***m221Scara***. Supprimer le service « ***Write To 150 »*** après l'avoir sélectionné pour le remplacer par le service « ***Read From 100 / Write To 150*** ». Vérifier que ce nouveau service offre ***4 octets en entrée*** et ***40 octets en sortie*** avec une ***lecture/écriture périodique*** de ***T#30ms*** (***RPI : R**equested **P**acket **I**nterval*).
  - Modifier le paramètre ***RPI*** afin qu'il soit compatible avec la périodicité de la tâche ***Mast***.

  Appuyer sur le bouton « ***Appliquer*** » pour valider vos saisies et fermer la fenêtre.

  Dans la liste des équipements sur le réseau ***NOC_ETHIP_ROBOT*** figurent :
  - en position ***[003] cs9Scara***. Le sélectionner. Avec l'***Onglet Paramétrage de l'adresse*** : modifier l'adresse ***IP*** donnée par défaut *(192.168.0.2)* afin qu'elle corresponde à celle du plan d'adressage de la ***figure 1 : 192.168.0.10***

- en position ***[004] m221Scara***. Le sélectionner. Avec l'***Onglet Paramétrage de l'adresse*** : modifier l'adresse ***IP*** donnée par défaut *(192.168.0.3)* afin qu'elle corresponde à celle du plan d'adressage de la ***figure 1 : 192.168.0.11***

2.6.3 Sauver votre application. La compiler et vérifier avec l'éditeur de données (variables dérivées) que les zones mémoires allouées pour les équipements ajoutés répondent à vos attentes :

- ✓ pour les entrées (***T->O***) :
  cs9Scara_IN de type ***T_ cs9Scara_IN*** *d'adresse de base* ***%MW80***.
  m221Scara_IN de type ***T_m221Scara_IN*** *d'adresse de base* ***%MW154***.
- ✓ pour les sorties (***O->T***):
  cs9Scara OUT *de type* ***T_cs9Scara_OUT*** *d'adresse de base* ***%MW208***.
  m221Scara_OUT *de type* ***T_m221Scara_OUT*** *d'adresse de base* ***%MW270***.

3.0 Communication via Ethernet/IP *entre* l'***API*** *et la baie* ***CS9***

3.1 Créer deux sections de programme sous la rubrique ***Logique*** *de la tâche* ***Mast : ReadEthIpRobot*** et ***WriteEthIpRobot*** *en langage* ***ST***.

- ✓ La première ***ReadEthIpRobot*** aura pour but de ***lire les données du robot*** à l'aide d'une instance du bloc fonctionnel ***VAL_ReadAxesGroup***, puis de ***lire les informations de l'interface HMI*** *(pupitre et écran)* à l'aide d'une instance du bloc fonctionnel ***BF_ReadHmi. Elle devra être la première à être scrutée dans le cycle API.***

- ✓ La seconde ***WriteEthIpRobot*** aura pour but de ***fournir les informations de supervision à l'interface HMI*** *(voyants et écran)* à l'aide d'une instance du bloc fonctionnel ***BF_WriteHmi***, puis d'***écrire des données sur le robot*** à l'aide d'une instance du bloc fonctionnel ***VAL_WriteAxesGroup***. ***Elle devra être la dernière à être scrutée dans le cycle API.***

Nota : ***L'explorateur du projet*** définit ***l'ordre d'exécution des sections***. Pour modifier cet ordre, il suffit de ***sélectionner la section à déplacer***, et à l'aide de la souris, ***de la faire glisser à l'endroit souhaité*** *dans le cycle* ***API.***

3.2 Pour effectuer et simplifier la programmation (éparpillement des données), on définira le type de données ***TCtrlStatusGroup*** *avec l'éditeur de données, onglet* ***Types DDT***.

```
TYPE TCtrlStatusGroup
    STRUCT
        m_xEnable     : BOOL ;
        m_xBusy       : BOOL ;
        m_xDone       : BOOL ;
        m_xError      : BOOL ;
        m_udiErrorID  : UDINT ;
        m_xTimeOut    : BOOL ;
    END_STRUCT
END_TYPE
```

3.3 Pour visualiser sur l'interface ***HMI*** l'état du robot et prendre en compte les éléments de commande du pupitre, vous aurez besoin d'importer les deux blocs fonctionnels ***FB_ReadHmi*** *et* ***FB_WriteHmi***. Pour cela, faire ***un clique droit*** sur le dossier ***Types FB dérivés*** et choisir l'option ***importer***. Les fichiers d'extension ***xdb*** à importer se trouvent sous *« U:/Documents/Licence Pro/RIF/UE2/Automatisme pour la robotique/Scara/import »* et ont pour nom ***FB_ReadHmi.xdb*** et ***FB_WriteHmi.xdb***. Comme ils utilisent ***des types déjà définis dans l'application***, ***sélectionner*** le bouton *« **Garder Tout »*** si l'import le demande avant d'appuyer sur le bouton *« **OK ».***

3.4 Effectuer les réservations suivantes à l'aide de l'éditeur de données, onglet ***Variables.***

Nom	***Type***	*Adresse*	
ctrlStatusGroupRead	***TCtrlStatusGroup***		
ctrlStatusGroupWrite	***TCtrlStatusGroup***		
fromRobotScara	***T_FromRobot***		(*)
toRobotScara	***T_ToRobot***		(*)
fromHmiScara	***TFromHmi***		(*)
toHmiScara	***TToHmi***		(*)
staeubliRobotScara	***T_StaeubliRobot***		

(*) Ne fixer les adresses qu'aux objets qui le nécessitent. Se reporter à vos déclarations du paragraphe 1.2 et 1.3

3.4.1 Toujours avec l'éditeur de données, onglet ***Bloc fonction****, créer les instances des blocs fonctionnels* ***VAL_ReadAxesGroup*** *et* ***VAL_WriteAxesGroup*** *de la bibliothèque* ***UnivalPlc*** *nécessaires aux sections de programme* ***ReadEthInRobot*** *et* ***WriteEthIpRobot.***

3.4.2 Renseigner les corps des sections de programme ***ReadEthInRobot*** *et* ***WriteEthIpRobot*** *à l'aide des objets réservés précédemment. Ne valider les instances des blocs fonctionnels utilisées que si le bit de santé lié à la communication implicite avec la baie* ***CS9*** *l'autorise (cf. NOC_ETHIP_ROBOT_IN.HEALTH_BITS_IN[0].0).*

Nota *: La documentation relative à la bibliothèque client* ***uniVALplc*** *est disponible sous le répertoire : U:/Documents/Licence Pro/RIF/UE2/Automatisme pour la robotique/Scara/uniVALplc/doc*

3.5 Créer une table d'animation ***Scara*** *dans laquelle on visualisera les quatre données suivantes :* ***NOC_ETHIP_ROBOT_IN.HEALTH_BITS_IN****[0],* ***ctrlStatusGroupRead****,* ***ctrlStatusGroupWrite*** et ***staeubliRobotScara***

3.6 Mise en Œuvre

- ✓ *Compiler puis transférer l'application sur l'****API*** *d'adresse @IP :* ***172.16.180.180****. Puis, mettre en* ***RUN*** *l'automate.*
- ✓ *A l'aide de la table d'animation* ***Scara****, vérifier si l'état de santé de l'équipement* ***CS9*** *autorise les transferts dans les deux sens entre la baie* ***CS9*** *et l'automate. Si tel est le cas, et qu'il n'a pas d'erreur (cf* ***ctrlStatusGroupRead*** *et* ***ctrlStatusGroupWrite****) la donnée* ***staeubliRobotScara*** *permet alors de connaitre l'état du robot et de la baie (par exemple : sa position angulaire, sa position cartésienne dans le repère World).*

4. Communication via Ethernet/IP *entre* l'***API*** *et le contrôleur* ***M221***

4.1 Avec ***l'éditeur de données****, onglet* ***Bloc fonction****, créer les instances des blocs fonctionnels importés nécessaires pour compléter les sections de programme* ***ReadEthIpRobot*** *et* ***WriteEthIpRobot.***

4.2 A l'aide des réservations précédentes et d'une nouvelle donnée à réserver ***hmiScara*** *de type* ***THmi****, compléter :*

- ✓ *la section* ***ReadEthIpRobot*** *en plaçant la scrutation de l'instance de* ***FB_ReadHmi après celle de VAL_ReadAxesGroup***
- ✓ *la section* ***WriteEthIpRobot*** *en plaçant la scrutation de l'instance de* ***FB_WriteHmi avant celle de SRC_WriteAxesGroup***

*Pour valider les instances des blocs fonctionnels nouvellement utilisées, vous serez amené à regarder le bit de santé lié à la communication implicite avec le contrôleur **M221** (cf. **NOC_ETHIP_ROBOT_IN.HEALTH_BITS_IN[0].1**) mais aussi avec celui associé à la communication avec la baie **CS9.***

*4.3 Compléter la table d'animation **Scara** pour visualiser la donnée **hmiScara***

*4.4 Compiler puis transférer l'application. Mettre en **RUN** l'automate. Sur l'écran doivent apparaitre les positions des différents axes et la position cartésienne du robot. A l'aide de la donnée **hmiScara**, il vous sera possible de lire les **éléments de commande** et **d'agir sur la signalisation** (voyants, ...).*

5. Système de préhension

Le ***système de préhension*** est réalisé à l'aide d'un *« **venturi** »* et d'une ***ventouse***. Il nécessite donc un ***distributeur pneumatique*** pour sa commande. Comme le robot ***Scara*** de notre cellule ne dispose pas ***en interne*** de cet équipement, il est réalisé en ***externe***. Sur le connecteur ***J212*** de la baie ***CS9***, deux sorties rapides ***TOR : fastout0 (J212-4/J212-9)*** et ***fastout1 (J212-1/J212-5)*** sont disponibles. Comme le ***distributeur externe*** a été choisi ***bistable***, ces deux sorties nous permettront d'en établir ses ***commandes électriques***. Les conventions prises sont les suivantes : la sortie ***fastout0*** contrôle l'action de ***saisir***, la sortie ***fastout1*** l'action de ***déposer***. Le système de préhension n'étant pas instrumenté, les ***durées** « **enveloppes** »* à prendre en compte pour dire qu'une ***saisie*** est effective est de ***T#300ms***, et pour la ***dépose T#1s***.

*5.1 Etablir **en langage ST** une structure de données **TPrehenseur** permettant de décrire le système de préhension : ses commandes et aussi la dynamique de celles-ci.*

*5.2 Créer **en langage ST** un objet **prehenseur** de type **TPrehenseur** et initialiser ses membres à partir des caractéristiques de notre venturi.*

*5.3 Le bloc fonctionnel **VAL_ReadWriteIO** permet l'accès aux entrées/sorties du connecteur **J212**. Son interface est donnée par :*

```
FUNCTION_BLOCK VAL_ReadWriteIO
    (* Update the digital outputs of the controller with values passed as parameter.
    Return the state of the digital inputs available on the controller *)
    VAR_INPUT
        Enable          : BOOL ; (*function block content is executed as long as this input is TRUE *)
        Valve1          : BOOL ; (* 6 axis robot only : the internal valve *)
        Valve2          : BOOL ; (* 6 axis robot only : the internal valve *)
        Fout0           : BOOL ; (* Command Fast output (J212-4 / J212-9) CS9 controller *)
        Fout1           : BOOL ; (* Command Fast output (J212-1 / J212-5) CS9 controller *)
        EnableValve1    : BOOL ; (* 6 axis robot only : the internal valve 5-3 closed middle *)
        EnableValve2    : BOOL ; (* 6 axis robot only : the internal valve 5-3 closed middle *)
    END_VAR
    VAR_IN_OUT
        AxesGroup       : T_StaeubliRobot; (* Data block for a robot *)
    END_VAR
    VAR_OUTPUT
        Valid           : BOOL ; (*Set when function block is executing. Reset when Done or Error*)
        UsrInput0       : BOOL ; (* Digital input (J109-11 / J109-30) CS8 controller *)
        UsrInput1       : BOOL ; (* Digital input (J109-16 / J109-35) CS8 controller *)
        Fin0            : BOOL ; (* Fast input (J111-2/7) CS8 controller, J212-2/7 CS9 controller *)
        Fin1            : BOOL ; (* Fast input (J111-3/8) CS8 controller, J212-3/8 CS9 controller *)
        ValvesSafeState : BOOL ; (* CS9 controlling 5-3 ways valves ONLY *)
    END_VAR
```

Nota : *Bien que sur un robot **SCARA** tous les paramètres ne soient pas disponibles, vous devez quand même les fournir.*

*5.3.1 Etablir une structure de données **Tj212IO** qui permet une utilisation simple de ce bloc et un non éparpillement des données.*

*5.3.2 A l'aide d'une instance du bloc fonctionnel **VAL_ReadWriteIO** et d'un objet **j212IO** de type **Tj212IO**, compléter la section **WriteEthIpRobot** permettant de **faire le lien** entre l'objet **prehenseur** et **la commande** des deux sorties rapides **TOR***

*5.4 Compléter les sections **F1_ProductionNormale et F1_ProductionNormalePost** afin de commander la saisie puis une dépose durant l'arrêt d'une palette pleine dans le guichet. On espacera ces 2 commandes de **T#1s**.*

*5.5 Compiler puis transférer l'application. Mettre en **RUN** l'automate pour effectuer la recette de la commande du système de préhension.*

Annexe 1 :

```
TYPE T_StaeubliRobot :
        STRUCT
          Status              : T_Status ;
          Command             : T_Command ;
          CommandInterface    : T_CommandInterface ;
        END_STRUCT
END_TYPE
```

avec :

```
TYPE T_Status :
        STRUCT
          Initialized         : BOOL ;    (* True = The library is initialized *)
          Online              : BOOL ;    (* True = robot can receive commands *)
          ErrorPending        : BOOL ;    (* True = an error is pending *)
          IsMoving            : BOOL ;    (* True = robot is moving *)
          EStopActive         : BOOL ;    (* True = E-stop is pending *)
          DummyPlug           : BOOL ;    (* True = staubli teach pendant replaced by dummy
                                           plug *)
          ExternalMcp_Wms     : BOOL ;    (* True = Robot is properly configured to use both
                                           user- supplied WMS and Teach Pendant. *)
          ActualSpeed         : REAL ;    (* Cartesian speed of the current TCP *)
          ActualOverride      : REAL ;    (* Current override value [0.01 .. 100] *)
          ActualErrorNumber   : UINT ;    (* Total number of pending errors in robot *)
          ActualOperationMode : UINT ;    (* Actual working mode 0= invalid , 1=manu,
                                           3=Auto, 4=remote(extaut) *)
          ActualErrorID       : T_PendingErrors ;  (* List of pending error on server side *)
          CartesianPos        : T_CartesianPos ;   (* Current cartesian position *)
          JointPos            : T_JointPos ;       (* Current joint position *)
          ActualCoordSystem   : UINT ;    (* Number of the user frame in which the position of
                                           the Tool Center Point is reported *)
          ActualTool          : UINT ;    (* Number of the Tool Center Point for which the
                                           position is reported *)
          RobotStateMachine   : UINT ;    (* State Machine of the robot *)
          MovementID          : INT ;     (* Identifier of motion currently executed *)
          MovementProgress    : INT ;     (* Percentage of actual movement that has been
                                           completed *)
          RobotModel          : INT       (* Robot model connected to controller *)
          ControllerModel     : UINT ;    (* Stäubli controller 8=CS8C / 9=CS9 *)
          CS9Safety           : T_CS9SftyFbk ;     (* CS9 Only - Safety features *)
          Heartbeat           : UINT ;
          ServerMajorVersion  : UINT ;    (* Major version of unival PLC server *)
          ServerMinorVersion  : UINT ;    (* Minor version of unival PLC server *)
          ServerEdit          : UINT ;    (* Edit of the uniVAL plc server *)
          ClientMajorVersion  : UINT ;    (* Major version of unival PLC client library *)
          ClientMinorVersion  : UINT ;    (* Minor version of unival PLC client library *)
          ClientEdit          : UINT ;    (* Edit of the unival PLC client library *)
        END_STRUCT
END TYPE
```

```
TYPE T_Command :
    STRUCT
      EnableVerbose        : BOOL ;   (* TRUE=Enable tracing activity of the server. It is
                                         strongly recommended to enable trace for debugging
                                         purpose only. Starting with FALSE *)
      OverrideCmd          : UINT ;   (* Commanded Override [1..100] (monitor speed) Starting
                                         with 0 *)
      OperationModeCmd     : UINT ;   (* Commanded operation mode 0= invalid , 1= Manu,
                                         3=Auto, 4 remote(extAut) *)
      ToolCmd              : UINT ;   (* Select the Tool used for movement. Starting with 0
      CoordSystemCmd       : UINT ;   (* Select the coordinate system used for movement.
                                         Starting with 0 *)
      CS9Safety            : T_CS9SftyCmd ;  (* CS9 Only - Safety features *)
      LifebitPeriod        : TIME ;   (* Select the period of the internal lifebit
                                         (100ms<Period<1000ms) . starting with t#200ms *)
    END_STRUCT
END TYPE
```

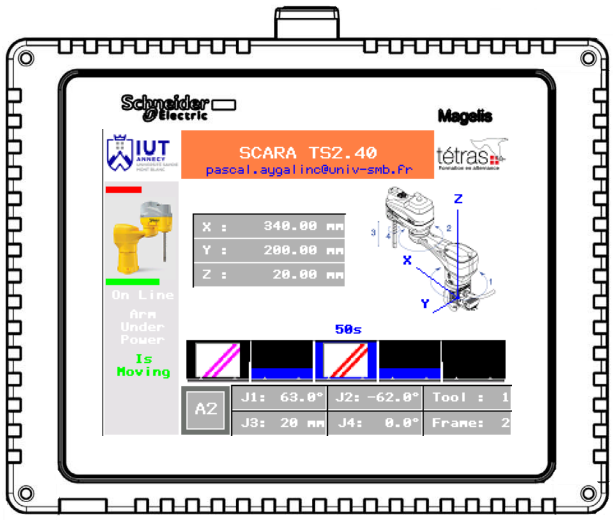
Annexe 2 :

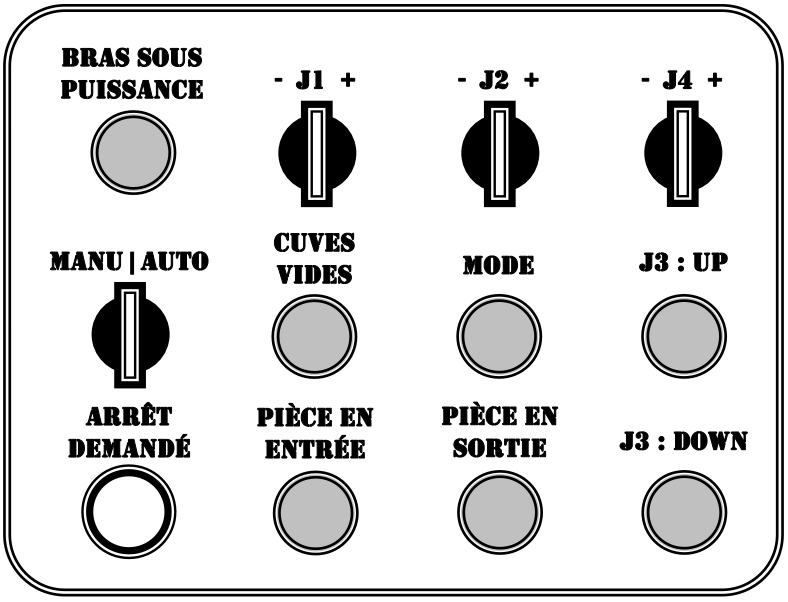
```
FUNCTION_BLOCK VAL_ReadAxesGroup
    VAR_INPUT
        Enable      : BOOL ;          (* TRUE : contents of the FB is executed *)
    END_VAR
    VAR_IN_OUT
        AxesGroup   : T_StaeubliRobot ;  (* Data block for a robot *)
        IN          : T_FromRobot ;
    END_VAR

    VAR_OUTPUT
        Error       : BOOL ;          (* set when function block has terminated with error *)
        ErrorID     : UDINT ;         (* error code *)
        Timeout     : BOOL ;          (* set when a timeout is detected *)
    END_VAR

FUNCTION_BLOCK VAL_WriteAxesGroup
    VAR_INPUT
        Enable      : BOOL ;          (TRUE : contents of the FB is executed *)
    END_VAR
    VAR_IN_OUT
        AxesGroup   : T_StaeubliRobot ;  (* Data block for a robot *)
    END_VAR
    VAR_OUTPUT
        Error       : BOOL ;          (* set when function block has terminated with error *)
        ErrorID     : UDINT ;         (* error code *)
        OUT         : T_ToRobot ;
        Timeout     : BOOL ;          (* set when a timeout is detected *)
    END_VAR
```

Annexe 3** : **Pupitre HMI, Eléments de commandes et Voyants






```
TYPE THmi
    STRUCT
        m_Inputs    : THmiInputs ;
        m_Outputs   : THmiOutputs ;
    END_STRUCT
END_TYPE
```

avec :

```
TYPE THmiInputs                                  (* NOpen, NClose *)     (* M221 : @*)
    STRUCT
        m_nixJ1Plus          : BOOL;             (* Switch NO *)         %I0
        m_nixJ1Minus         : BOOL;             (* Switch NO *)         %I1
        m_nixJ2Plus          : BOOL;             (* Switch NO *)         %I2
        m_nixJ2Minus         : BOOL;             (* Switch NO *)         %I3
        m_nixJ3Up            : BOOL;             (* Push Button NO *)    %I4
        m_nixJ3Down          : BOOL;             (* Push Button NO *)    %I5
        m_nixJ4Plus          : BOOL;             (* Switch NO *)         %I6
        m_nixJ4Minus         : BOOL;             (* Switch NO *)         %I7
        m_nixAuto            : BOOL;             (* Switch NO *)         %I8
        m_nixManu            : BOOL;             (* Switch NO *)         %I9
        m_nixEmptyTanks      : BOOL;             (* Push Button NO *)    %I10
        m_nixUnloadTank4     : BOOL;             (* Push Button NO *)    %I11
        m_nixLoadTank0       : BOOL;             (* Push Button NO *)    %I12
        m_nixEmptyTank0      : BOOL;             (* Sensor NC *)         %I13
        m_nixEmptyTank4      : BOOL;             (* Sensor NC *)         %I14
        m_nixStopRequest     : BOOL;             (*Push Button NC *)    %I15
    END_STRUCT
END_TYPE
```

```
TYPE THmiOutputs
    STRUCT
        m_qToR      : THmiQTOR
        m_Screen    : THmiScreen
    END_STRUCT
END_TYPE
```

```
TYPE THmiQTOR                                                          (* M221
    STRUCT                                                              : @*)
        m_nqxRedColumn              : BOOL;                             %Q0
        m_nqxGreenColumn            : BOOL;                             %Q1
        m_nqxBlueColumn             : BOOL;                             %Q2
        m_nqxArmUnderPower          : BOOL;                             %Q3
        m_na4qxHex7Seg              : ARRAY[0..3]OF BOOL;  (* [0]:LSB, [3]:MSB *)  %Q4
        m_nqxLightLoadTank0         : BOOL;                             %Q8
        m_nqxLightLoadTank1         : BOOL;                             %Q9
        m_nqxLightLoadTank2         : BOOL;                             %Q10
        m_nqxLightLoadTank3         : BOOL;                             %Q11
        m_nqxLightUnLoadTank4       : BOOL;                             %Q12
        m_nqxLightStopRequest       : BOOL;                             %Q13
    END_STRUCT
END_TYPE

TYPE THmiScreen
    STRUCT
        m_iStateGmma                : INT;
        m_iCurvSpeed_mm_per_s       : INT;
        m_a5tankLine                : ARRAY[0..4]OF TTank;
    END_STRUCT
END_TYPE

TYPE TTank
    STRUCT
        m_intPartNumber             : INT
        m_xBusy                     : BOOL;
        m_tElapsedTime              : TIME;
    END_STRUCT
END_TYPE

FUNCTION_BLOCK FB_ReadHmi
    VAR_INPUT
        xEnable     : BOOL ;        (* TRUE : contents of the FB is executed *)
    END_VAR
    VAR_IN_OUT
        AxesGroup   : T_StaeubliRobot ;  (* Data block for a robot *)
        fromHmi     : TfromHmi ;
        hmi         : THmi ;
    END_VAR

FUNCTION_BLOCK FB_WriteHmi
    VAR_INPUT
        xEnable     : BOOL ;        (* TRUE : contents of the FB is executed *)
    END_VAR
    VAR_IN_OUT
        AxesGroup   : T_StaeubliRobot ;  (* Data block for a robot *)
        hmi         : THmi ;
    END_VAR
    VAR_OUTPUT
        toHmi       : TtoHmi ;
    END_VAR
```