**Binôme 18 TDSI**

**DALIES Pierre**

**GUENOT-FALQUE Raphaël**

# TP M3-M4 Chaîne de traitement de surface

### But du tp:

Le but de ce TP est de se familiariser à l’utilisation de l’approche orientée composants d'automatismes. Contrairement à l’approche GEMMA GRAFCET, cette méthode de programmation permet d'avoir une bonne réutilisabilité sur ce qui est developpé: on créer des objets génériques (sous forme de composant) que l'on ajoute à la libraire classique, on utilise intègre ensuite le composant dans notre application en définissant uniquement ses entrées/sorties.

### Cadre du tp:

Le dévellopement se fera sur le logiciel Unity Pro, dans lequel nous créerons des composants appelés Blocs Fonctions Dérivés (DFB) et des structures de données appelées Type de Données Dérivées (DDT).

L’objectif sera d’arriver à contrôler deux chariots A et B, les faire s’arrêter au dessus de cuves et les faire descendre un bras. Ceci permet de modéliser une application industrielle de pièces devant subir plusieurs traitements par plongée dans des bains de produits.

Nous repartierons pour cela d'un projet dont certaines composantes importantes seront déjà existantes (comme le DFB Decodage\_Clavier qui permet de gérer un clavier ou la section gestion\_local\_distant permetant la mémorisation des variables).

## Travail demandé:

### Partie A: Analyse fonctionnelle

Il faut pouvoir exécuter mes modes de fonctionnements suivant:

|  |  |
| --- | --- |
| Mode MANU: | Mode AUTO: |
| mode_manu.jpg | mode_manu.jpg |

Pour le mode MANU, il y a juste besoin d'un composant qui gère le mouvement d'un chariot. Il permettra de faire avancer, reculer un chariot et de monter et descendre le bras du chariot.

Propriétés du composant de gestion du mouvement:

Les entrées du composant devront donc être:

- commande d'avance

- commande de recule

- commande de montée

- commande de descente

- sélecteur du chariot (utilisé de manière similaire à un ENABLE du composant)

Les sorties devront être:

- ordre d'avance

- ordre de recule

- ordre de montée

- ordre de descente

Il est de plus nécessaire d'utiliser un ENABLE pour activer le composant uniquement lors de l'utilisation en mode MANU.

Pour le mode AUTO, nous avons pensé utiliser un FDB pour l'initialisation, un composant pour le fonctionnement en production normale et un composant pour la fin de cycle. Chacun des composants utilisera un composant de mouvement du chariot.

Propriétés du composant d'initialisation:

Les entrées du composant devront donc être:

- ordre de mise en marche (bouton MARCHE)

- capteur de position haute (pour le bras)

- capteur de position basse (pour le bras)

- capteur de position (position sur le déplacement horizontale)

Les sorties devront être:

- commande d'avance

- commande de recule

- commande de montée

- commande de descente

Les connexions inter-composant devront être:

- fin d'initialisation (vers le composant de production normal)

Il est de plus nécessaire d'utiliser un ENABLE pour activer le composant uniquement lors de l'utilisation en mode AUTO.

Propriétés du composant de mise en marche:

Les entrées du composant devront donc être:

- fin d'initialisation

- capteur de position haute (pour le bras)

- capteur de position basse (pour le bras)

- capteur de position (position sur le déplacement horizontale)

Les sorties devront être:

- commande d'avance

- commande de recule

- commande de montée

- commande de descente

Il est de plus nécessaire d'utiliser un ENABLE pour activer le composant uniquement lors de l'utilisation en mode AUTO.

Propriétés du composant de fin de cycle:

Les entrées du composant devront donc être:

- ordre d'arrêt

- capteur de position haute (pour le bras)

- capteur de position basse (pour le bras)

- capteur de position (position sur le déplacement horizontale)

Les sorties devront être:

- commande d'avance

- commande de recule

- commande de montée

- commande de descente

Il est de plus nécessaire d'utiliser un ENABLE pour activer le composant uniquement lors de l'utilisation en mode AUTO.

### Partie B: Le mode manuel :

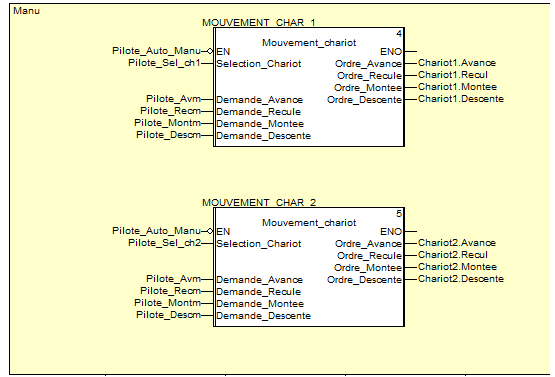
Cette partie du TP est consacrée à la programmation de la gestion des chariots en mode manuel.

Nous avons décidé de créer un DFB correspondant a un chariot: il permet de faire avancer ou reculer le chariot et de faire monter ou descendre le bras du chariot. Le passage par les variables définit dansla section "gestion\_local\_distant" permet de mémoriser l’ordre à l’entrée du composant. Nous avons aussi activé et complémenté la broche « Enable » afin d’activer cette boîte uniquement quand le mode MANUEL est sélectionné (Mode AUTO = 1 ; Mode MANUEL = 0). Cette astuce permettra d’éviter d’éventuels conflits entre les modes manuel et automatique.

Nous avons programmé de manière à reponsabiliser l’utilisateur au maximum : le chariot avancera tant que l’ordre d’avancer sera envoyé ; en revanche, il arrêtera de monter (ou de descendre) lorsque le capteur chaut (cbas) sera activé.

Une amélioration envisegeable de ce mode aurait été d’empêcher le chariot de continuer d’avancer ou de reculer au-delà de ses limites afin d’éviter tout accident durant les manœuvres manuelles.

Voici l'implémentation des DFB dans le programme principal (placés dans le MAIN):



Le DFB MOUVEMENT\_CHAR est constitué de 4 fonction logiques différentes qui ont respectivement en sorties logique Ordre\_Avance, Ordre\_Recule, Ordre\_Montee et pour la dernière Ordre\_Descente.

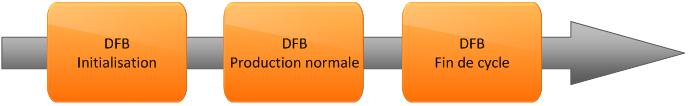
Par exemple pour la commande d'ordre d'avance on avait :

Ordre\_Avance = Demande\_Avance ET Selection\_Chariot (oubli d'imprim écran du contenu du DFB)

### Partie C: Le mode automatique :

Lors de l'étude de la solution possible pour commander les chariots en mode automatiques nous avions imaginés trois composants indépendants qui s'effectuaient l'un après l'autre.

L'idée globale peut donc être modélisée comme sur le schéma suivant:



Le mode automatique peut être lancé à partir de n’importe quelle position des chariots (un simple appui sur le bouton auto/manu du clavier active ce mode). L'étape d’initialisation est donc indispensable. Nous avons décidé d’utiliser la variable interne ChariotX.Position qui permet de récupérer le numéro de la cuve au dessus de laquelle est le chariot X.

#### Conception du DFB d'initialisation:

L'initialisation est réalisée en trois étapes successives:

- la remontée du bras du chariot

- le retour au dessus du premier bac pour le chariot (reculer)

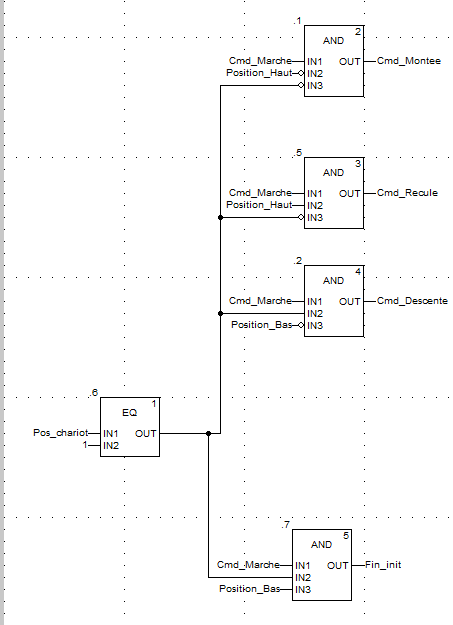
- la descente du bras du chariot

Nous avons donc programmé le composant d'initialisation de la manière suivante:

Tant que le chariot X n’est pas au dessus du bac initial (bac 1 pour chariot 1 et bac 9 pour chariot 2, dans les 2 cas, cela correspond respectivement à leur position « 1 »), et que le bras n’est pas en position haute, on ordonne la montée de celui-ci. Dès que le capteur « chaut » (position\_haut) est activé, on recule jusqu’à la position « 1 ». A cet endroit, on ordonne la descente du bras jusqu’à ce que le capteur « cbas » (Position\_bas) soit activé.

La marche d’initialisation est ensuite terminée.

Le contenu du DFB est détaillé de manière précise dans la figure suivante.



*Figure : Contenu du FBD initialisation*

Les premiers tests de l'implémentation du composant dans le MAIN ont été concluant quel que soit la position du chariot (testé sur le premier chariot).

L'implémentation du BFD dans le MAIN est disponible dans l'annexe 1.

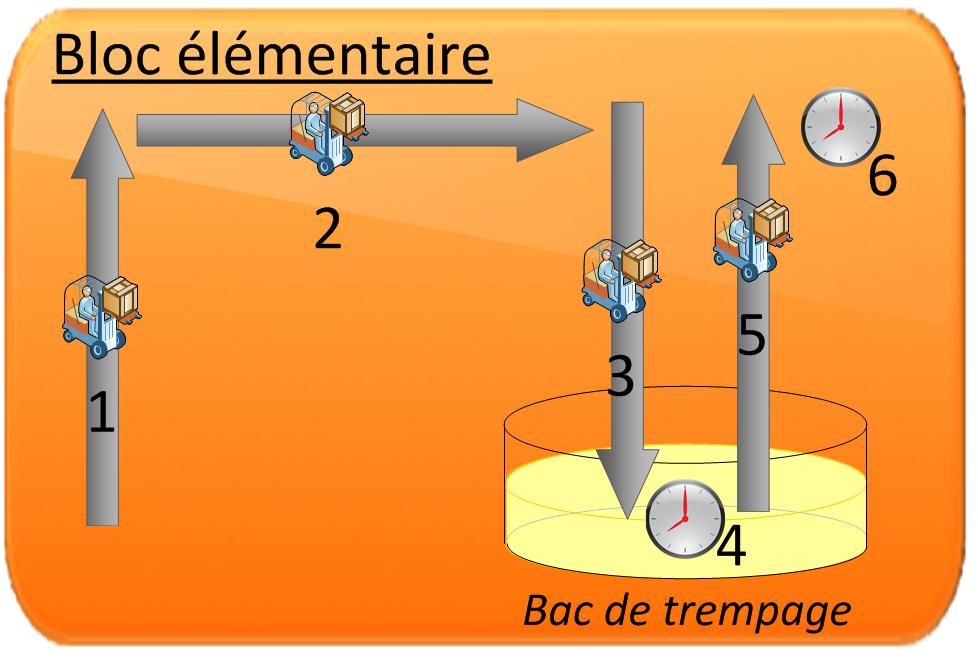
Pour cette partie nous avons utilisé uniquement les variables d'entrées (capteur de position) pour définir les conditions des ordres envoyés. Cela nous a ensuite porté préjudice par la suite comme nous allons le voir sur le composant suivant.

#### Conception du DFB de production normale:

Pour le DFB de production normale, nous avons utilisé des combinaisons de bloc élémentaire qui permettait ensuite de pouvoir respecter des étapes sur la production du produit (notion de différentes recettes de production pour différents résultats de production illustrées dans l'annexe 2).

Ce bloc élémentaire a pour but de monter jusqu'à la position haute (il est nécessaire de monter après l'initialisation), de se positionner au dessus du bac voulu, de descendre dans le bac, d'attendre le temps d'une temporisation de trempage, de remonter en position haute et d'attendre.

L'action qu'il est censé réaliser est décrite dans le schéma ci-dessous (les flèches représentent des déplacements du chariot et les horloges des temporisations):

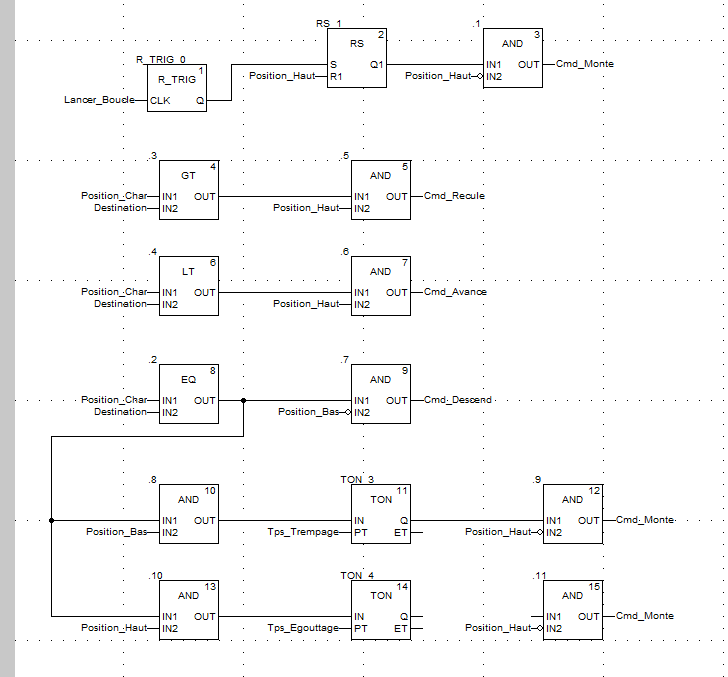


*Remarque: on notera que l'action 1 n'est pas toujours utile (notamment lors de l'exécution de deux boucles élémentaires consécutives).*

Nous avons conservé la même philosophie de programmation en essayant définir les actions en commande uniquement les sorties par des opérations logiques.

On se rend vite compte qu'il est indispensable d'utiliser un système de jeton (ou un système analogue pour faire du séquentiel) grâce, par exemple, à un système de bascule RS.

La capture d'écran suivante montre comment nous avons écrit le FDB élémentaire dont nous parlons plus haut.

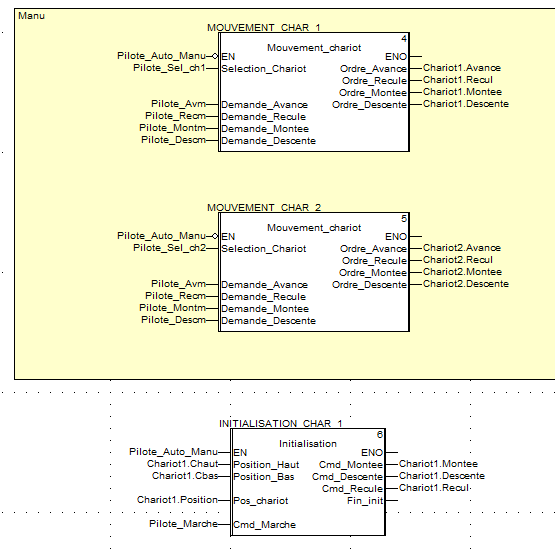


*Figure : Contenu du FBD de bloc élémentaire*

Avec une analyse simple des sorties on peut se rendre compte qu'il y a facilement un conflit entre plusieurs blocs. Il y a par exemple un problème avec la commande de remontée du bras qui se déclenche dans des cas non voulu.

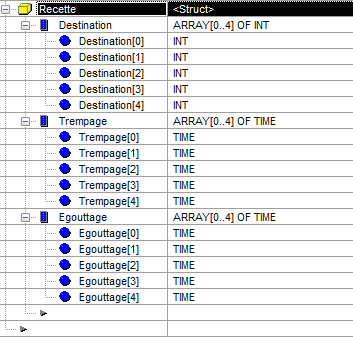
Il n'est plus possible de commander les sorties avec des cas particuliers définit uniquement par les capteurs, dès que le système se complexifie un peu.

## Annexe 1: test de la phase d'initialisation sur le chariot 1

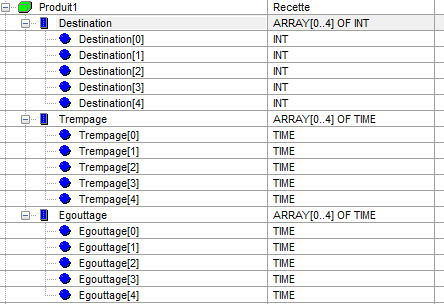


*Figure : implémentation dans le MAIN du composant d'initialisation*

## Annexe 2: Exemple de structure de données



*Figure: Définition du tableau « recette » en 3tableaux \*5 lignes (Défini dans « types DDT »)*



*Définition de la variable « Produit 1 » de type « Recette »*