Commande des systèmes à événements discrets par Automate Programmable Industriel (API)*

Objectif : L'objectif de cette séance consiste à réaliser le programme de commande d'une installation réelle dotée de composants industriels. La programmation se fait en utilisant les langages Grafcet et Ladder dans l'environnement Unity, également très répandu en industrie.

Pré-requis : module « Systèmes à événements discrets », filière ISA, 2ème année, premier semestre.

Description de la maquette

Le système qui doit être commandé représente un banc de tri des pièces qui seront ensuite aiguillées suivant leur type. Il est constitué des convoyeurs actionnés par des moteurs électriques, des vérins pneumatiques, d'un chariot motorisé, ainsi que des capteurs nécessaires. La partie de commande de ce système est représentée par un Automate Programmable Industriel (API) du type **TSX 57 2634 M Premium** de Schneider Electric.

Trois ordinateurs, supportant le logiciel Unity Pro, permettent le développement des programmes de commande, leur simulation, leur téléchargement dans l'API à travers une connexion réseau, ainsi que leur mise au point en simulation ou directement sur l'API.

Réalisation de la commande d'un poste de tri

L'installation permet le tri, suivant la taille ou la couleur, des pièces se présentant sur le convoyeur 1, et leur évacuation par les convoyeurs 2 et 3 vers le chariot. Les pièces sont ensuite transportées par le convoyeur 4 vers une zone de stockage permettant une reprise ultérieure lors du départ d'un nouveau cycle.

Début de cycle :

- le bouton *marche/arrêt* assure la mise en marche ;
- le bouton *initialisation* lance la procédure d'initialisation, il commande la sortie du vérin 5 de chargement (hypothèse : on considère que les pièces sont présentes dans la zone de stockage et, dans un premier temps, qu'aucune pièce n'est présente sur les convoyeurs de l'étage supérieur) ;
- le capteur 1 détecte la présence d'une pièce en début de chaîne ;
- le vérin 1 permet de pousser la pièce sur le convoyeur 1 qui doit être démarré à une vitesse initiale v_0 ;
- les capteurs 3 et 4 permettent d'identifier les pièces suivant leur taille ;
- le convoyeur 1 amène la pièce vers le poste d'aiguillage : les pièces « petites » à une vitesse v_1 et les pièces « grandes » à une vitesse v_2 ;
- les pièces « petites » sont évacuées par le convoyeur 2 ; les pièces « grandes » sont évacuées par le convoyeur 3 ;

---

* Vos remarques et commentaires sont les bienvenus.
Contact : Nicolae.Brinzei@univ-lorraine.fr, Jean-Francois.Petin@univ-lorraine.fr.

- le chariot récupère les pièces à la sortie des convoyeurs 2 ou 3 ;
- le chariot se positionne en face du capteur 5 ;
- le vérin 4 sort pour transférer la pièce sur le convoyeur 4 (capteur 6) ;
- la pièce est évacuée vers le lieu de stockage.

Fin de cycle : le bouton marche/arrêt en position arrêt.

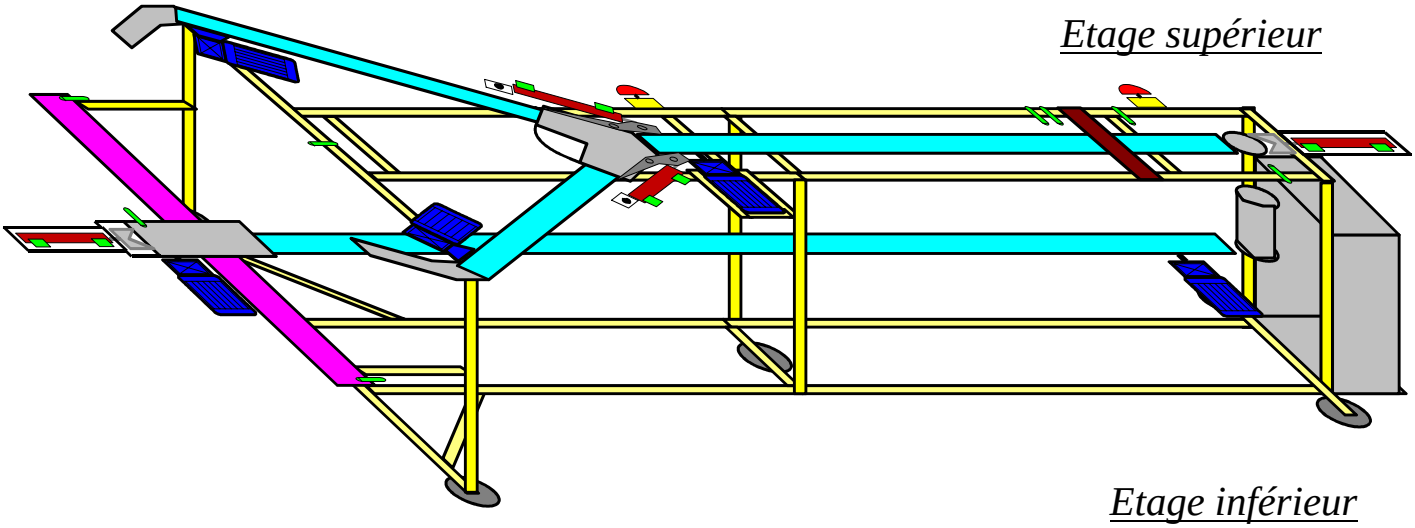
Améliorations :

- compter les pièces triées ;
- trier les pièces en fonction de leur couleur ;
- ajouter une phase d'initialisation qui permettra d'évacuer les pièces restantes sur les convoyeurs et de les mettre en stock avant de lancer le cycle de production ;
- allumer/éteindre les voyants sur le panneau de commande ;
- ajouter une partie à votre programme permettant le traitement des « arrêts d'urgence ».

Le schéma du banc de tri et d'aiguillage, ainsi que les entrées et les sorties de l'API sont présentées ci-après.

Convoyeur

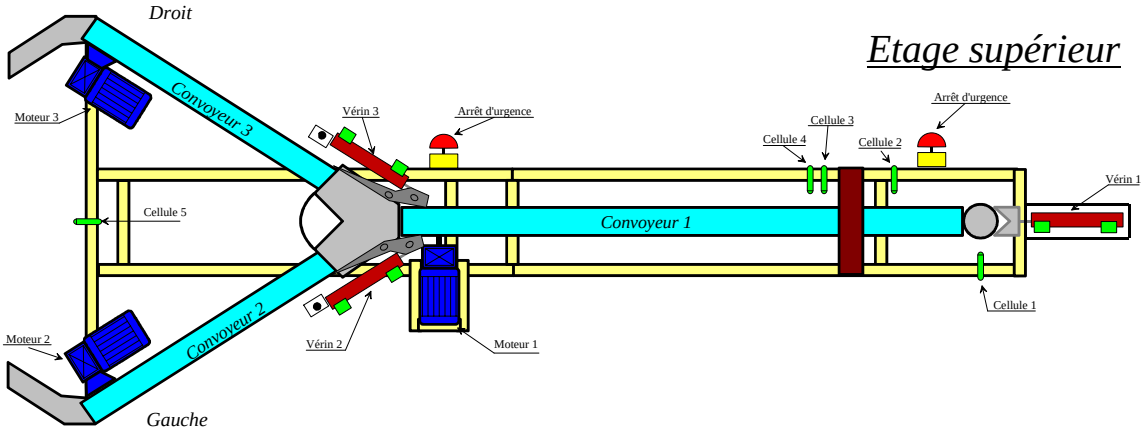
Etage supérieur



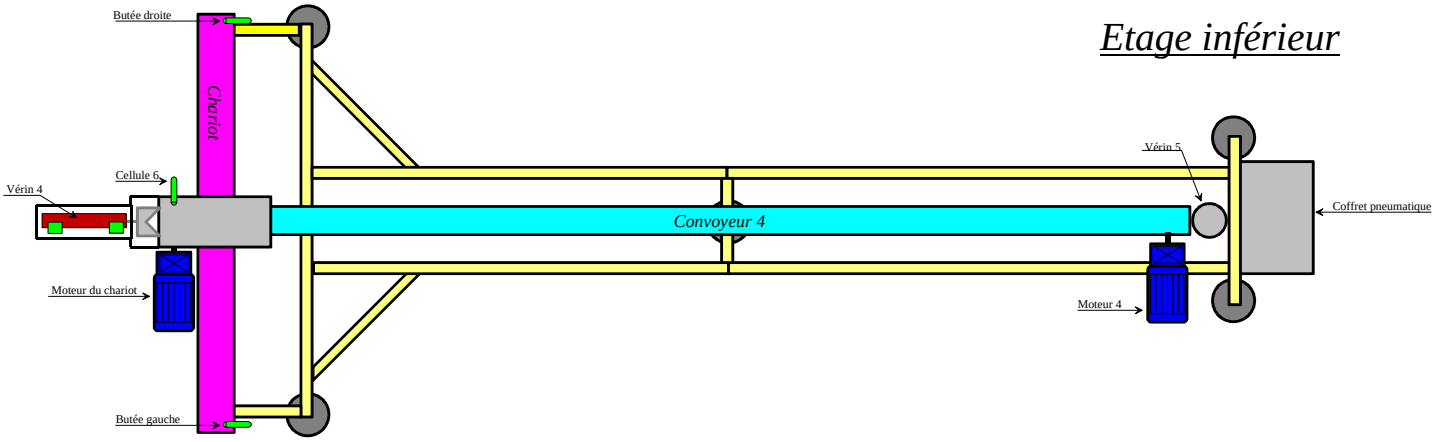
Etage inférieur

Convoyeur

Etage supérieur



Etage inférieur



Entrées TOR de l'API					
%I0.2.0	Bouton auto/manu	**%I0.2.16**	Le chariot va vers la droite	**%I0.2.32**	Pièce rouge
%I0.2.1	Bouton pas à pas	**%I0.2.17**	Le chariot va vers la gauche	**%I0.2.33**	Pièce jaune
%I0.2.2	Fonctionnement du moteur 1	**%I0.2.18**	Vérin V1 rentré	**%I0.2.34**	Pièce bleue
%I0.2.3	Fonctionnement du moteur 2	**%I0.2.19**	Vérin V1 sorti	**%I0.2.35**	Pièce noire
%I0.2.4	Fonctionnement du moteur 3	**%I0.2.20**	Vérin V2 sorti		
%I0.2.5	Fonctionnement du moteur 4	**%I0.2.21**	Vérin V2 rentré		
%I0.2.6	Ce capteur est à 1 quand le vérin 5 a monté une pièce	**%I0.2.22**	Vérin V3 sorti		
%I0.2.7	Arrivée sur le tapis du haut	**%I0.2.23**	Vérin V3 rentré		
%I0.2.8	Mesure petite hauteur	**%I0.2.24**	Vérin V4 rentré		
%I0.2.9	Mesure grande hauteur	**%I0.2.25**	Vérin V4 sorti		
%I0.2.10	Position centrale du chariot	**%I0.2.26**	Vérin V5 sorti		
%I0.2.11	Butée droite du chariot	**%I0.2.27**	Vérin V5 rentré		
%I0.2.12	Butée gauche du chariot	**%I0.2.28**			
%I0.2.13	Présence d'une pièce sur le chariot	**%I0.2.29**			
%I0.2.14	Bouton marche/arrêt	**%I0.2.30**			
%I0.2.15	Bouton initialisation	**%I0.2.31**			

Sorties TOR de l'API			
%Q0.3.0	Voyant défaut	**%Q0.3.16**	Marche chariot vers le droite
%Q0.3.1	Voyant auto/manu	**%Q0.3.17**	Marche chariot vers la gauche
%Q0.3.2	Voyant pas à pas	**%Q0.3.18**	Voyant de marche du chariot
%Q0.3.3		**%Q0.3.19**	Marche vérin V1
%Q0.3.4	Marche moteur 1 (convoyeur du haut)	**%Q0.3.20**	Marche vérin V2
%Q0.3.5	Marche moteur 2 (convoyeur de gauche)	**%Q0.3.21**	Marche vérin V3
%Q0.3.6	Marche moteur 3 (convoyeur de droite)	**%Q0.3.22**	Marche vérin V4
%Q0.3.7	Marche moteur 4 (convoyeur du bas)	**%Q0.3.23**	Marche vérin V5
%Q0.3.8	Voyant moteur 1	**%Q0.3.24**	
%Q0.3.9	Voyant moteur 2	**%Q0.3.25**	
%Q0.3.10	Voyant moteur 3	**%Q0.3.26**	
%Q0.3.11	Voyant moteur 4	**%Q0.3.27**	
%Q0.3.12		**%Q0.3.28**	
%Q0.3.13		**%Q0.3.29**	
%Q0.3.14		**%Q0.3.30**	
%Q0.3.15		**%Q0.3.31**	

Sorties analogiques de l'API	
%QW0.4.0	Variateur de vitesse du convoyeur 1 dans la plage [0, 10000] mV
%QW0.4.1	
%QW0.4.2	
%QW0.4.3	

Annexe 1 : utilisation de Unity

I. Configuration de l'automate sous le logiciel Unity

Avant de commencer la programmation proprement dite sous Unity, il faut d'abord configurer le type d'automate dans le logiciel.
Au démarrage de Unity, il faut créer un nouveau projet. Cliquez sur *Fichier* puis *Nouveau* puis choisissez un automate : une fenêtre identique à celle de la figure 1 s'ouvrira. Choisir l'API que l'on va utiliser (**TSX P57 2634M Premium**).

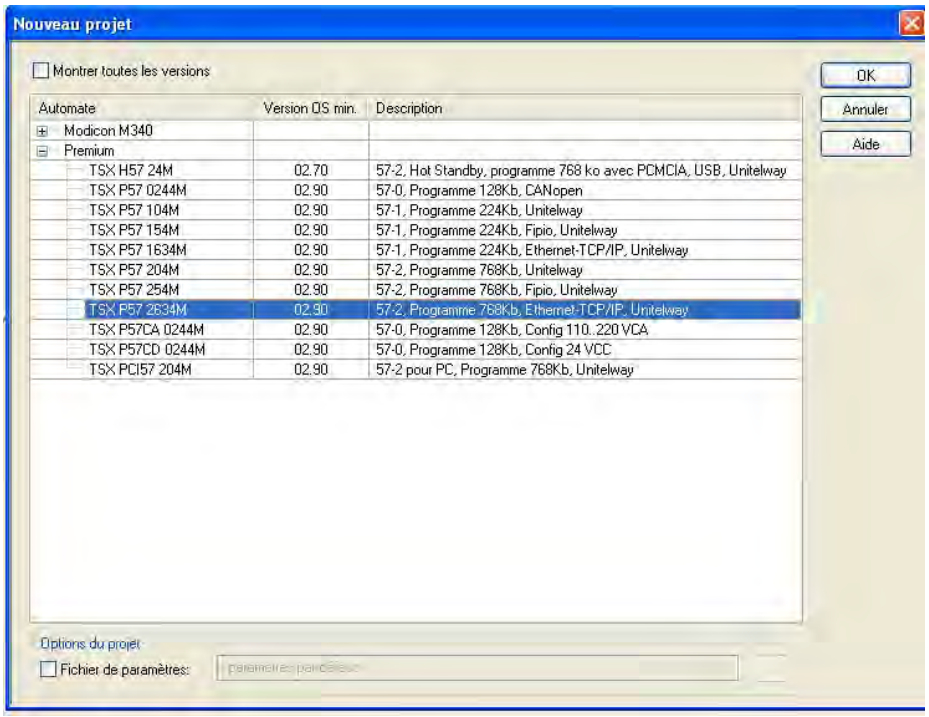


Figure 1. Création d'un projet

Validez par *OK*.

Il est nécessaire d'ajouter à la configuration matérielle de l'API les modules d'entrées-sorties dont il dispose :

- un module 64 entrées TOR (TSX DEY 64D2K) dans le rack numéro 2 ;
- un module 32 sorties TOR (TSX DSY 32T2K) dans le rack numéro 3 ;
- un module 4 sorties analogiques (TSX ASY 410) dans le rack numéro 4.

Procédez à la configuration matérielle de ces modules comme indiqué figure 2 :
Dans le *navigateur de projet* (visible sur la partie gauche de la fenêtre), dans *Projet -> Configuration -> 0:Bus X -> 0:TSX RKY 6EX*, il faut choisir un emplacement, par exemple 2, et double-cliquer dessus ; ajouter le module TSX DEY 64D2K. Procédez de la même manière pour ajouter les autres modules.

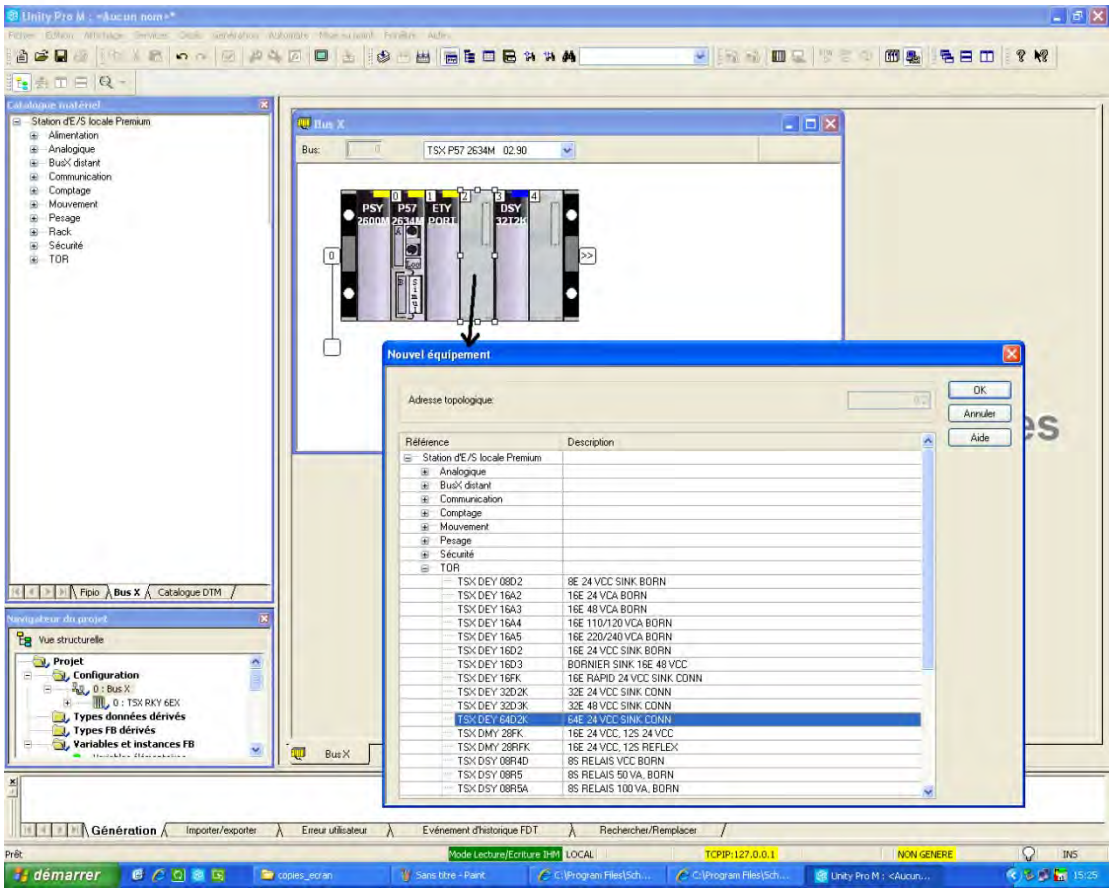


Figure 2. Configuration matérielle de l'API

II. Configuration de la connexion Ethernet

Afin de pouvoir utiliser la connexion Ethernet, il est nécessaire d'abord de la configurer.
Allez dans *Projet -> Communication -> Réseaux ->* click droit *Nouveau Réseau*, une fenêtre *Ajouter réseau* s'ouvre : dans le menu déroulant *Liste des réseaux disponibles*, choisissez *Ethernet*. Nommez-le *reseau_InTouch*. Confirmez par *OK*.
Ouvrez le réseau que vous venez de créer (*reseau_InTouch*) :

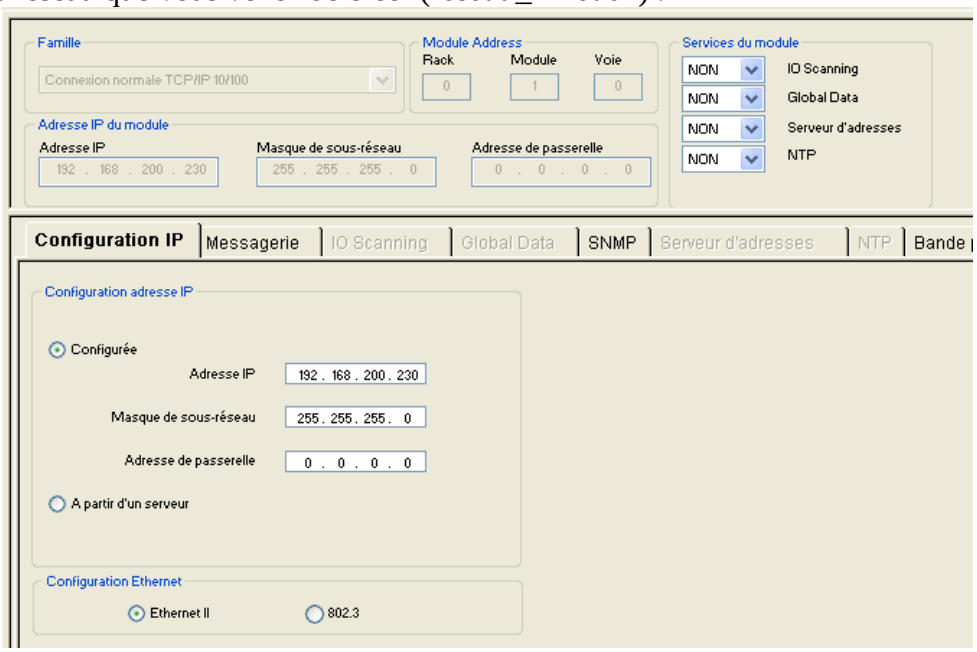


Figure 3. Configuration de la connexion

Entrez :

- l'adresse IP : 192.168.200.230
- le masque du sous-réseau : 255.255.255.0

Fermez la fenêtre en validant les modifications.

Dans *Projet -> Configuration -> 0 : Bus X* , double-cliquez sur le module **1 : TSX ETY PORT** dans le rack 1.
En sélectionnant *Voie 0* dans le menu déroulant *Fonction*, choisissez *ETH TCP IP*, puis dans *Lien réseau*, choisissez le réseau que vous venez de configurer (*reseau_InTouch*).

Dans le menu *Automate -> Définir l'adresse*, Choisir le *Média* **TCPIP** dans le menu déroulant et donner l'adresse **192.168.200.230** comme indiqué sur la figure :

Figure 4. Définition des adresses

III. Définition des variables

Une fois que vous avez accepté la configuration, il faut déclarer les variables (cf. Figure 5).

Dans le *Navigateur du projet* \ *Variables et instances FB*, double-cliquez sur *Variables élémentaires*.
Dans la fenêtre *Editeur de données* sélectionnez la case dans la colonne *Nom* puis entrez le nom de votre première variable (les accents ne sont pas acceptés), par exemple « rentre » pour une variable correspondante au capteur d'un vérin en position « rentré ».
Ensuite, sélectionnez le *Type* de cette variable, par exemple EBOOL (pour un booléen), ou INT (pour les sorties analogiques QW), et entrez l'adresse correspondante, par exemple %I0.2.18 pour la variable correspondante à « Vérin V1 rentré ». Procédez de la même manière pour définir des variables pour toutes les entrées et toutes les sorties.
Lorsque toutes vos variables sont déclarées, vous pouvez fermer la fenêtre.

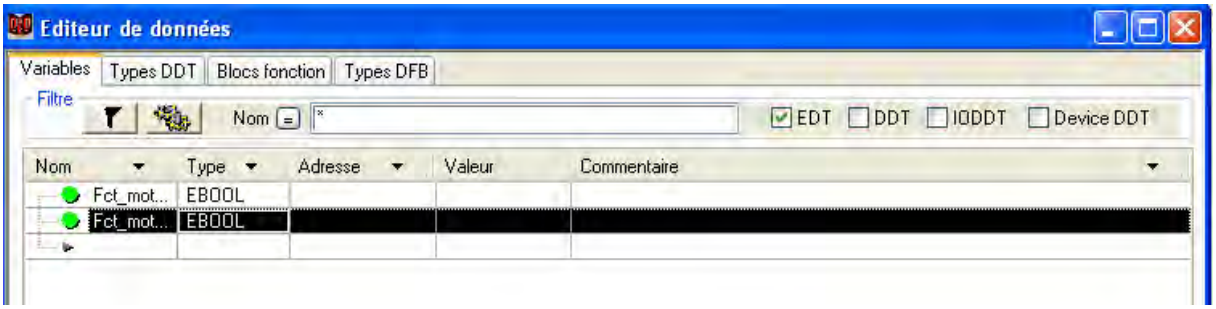


Figure 5. Déclaration des variables

IV. Création du Grafcet sous Unity

On peut ensuite créer le dossier qui contiendra le programme de commande :
Dans le *Navigateur de projet**Programme**Tâches**MAST*, faîtes un clic droit sur *Sections* puis choisissez *Nouvelle section*.
Donnez un nom à votre section puis sélectionnez le langage *SFC* (Grafcet), cliquez sur *OK*.

Un dossier (qui a le nom que vous avez choisi) sera créé contenant 4 éléments : le *Chart* (programme principal) et trois autres éléments *Macros non utilisées*, *Actions* et *Transitions* (qui peuvent être programmés en langage LD).

Ouvrez le *Chart*, les outils d'édition du Grafcet apparaissent alors dans la fenêtre, vous pouvez ainsi créer votre Grafcet (cliquez sur l'élément qui vous intéresse, par exemple une étape ou une transition, puis placez-la dans l'éditeur) :

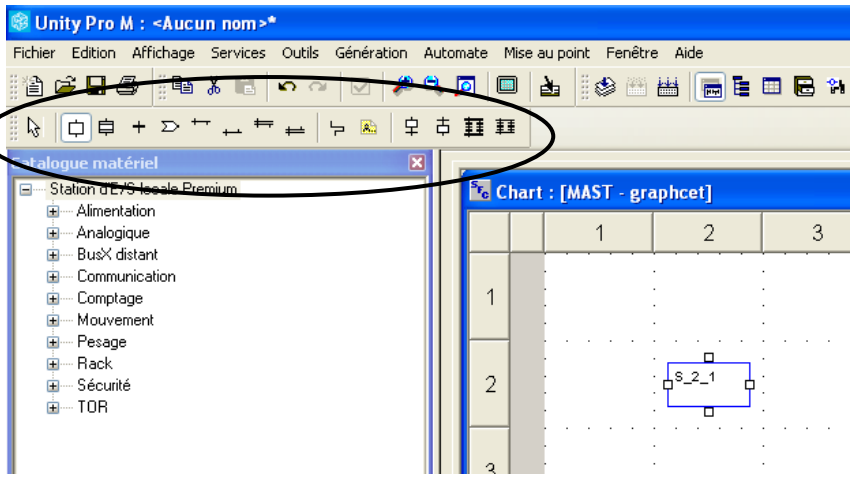


Figure 6. Construction du Grafcet

Placez alternativement les étapes et les transitions.
Pour les étapes initiales : placer une étape puis sélectionnez *Etape Initiale* dans *Propriétés* (dans le menu du clic droit sur l'étape choisie).
Pour boucler le Grafcet placez un *Saut* (cf. Figure 5) et indiquez le nom de l'étape à laquelle vous voulez le relier dans *click droit > Propriétés > Nom d'étape*.

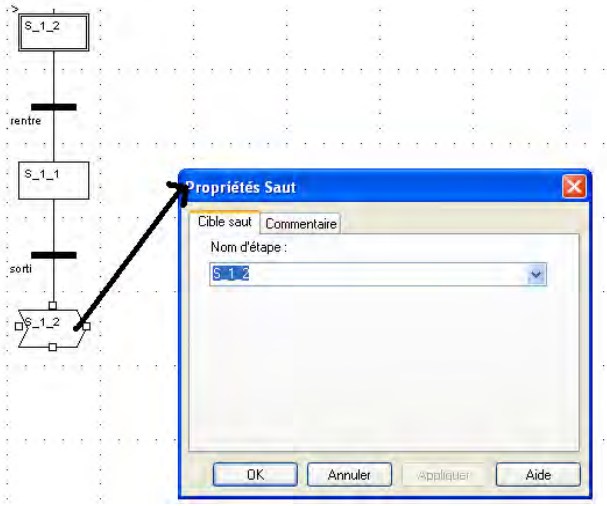


Figure 7. Saut

Vous pouvez associer à chaque transition et à chaque étape, au choix, un programme en langage LD ou une variable. Pour cela :

- pour les transitions : allez dans *Propriétés* (obtenu par click droit sur la transition) et choisissez
    - *section TRANSITION* puis choisissez un nom pour la transition et cliquez sur *Editer*, puis choisissez *LD* dans la fenêtre *Spécifier type de langage* et confirmez par *OK*.
      La fenêtre du programme Ladder de la transition s'ouvre alors : compléter-la (cf. Annexe 2).
    - OU *Variable* puis allez chercher la variable désirée dans « … », puis confirmez par *OK*.
- Pour les étapes : allez dans *Propriétés* (obtenu par click droit sur l'étape) puis dans l'onglet *Actions*, choisissez

- *Section* puis choisissez un nom pour l'étape et cliquez sur *Nouveau*, puis *Editer la section d'action*, choisissez *LD* dans la fenêtre *Spécifier type de langage* et confirmez par *OK*.
  La fenêtre du programme Ladder de l'étape s'ouvre alors : compléter-la (cf. Annexe 2).
- OU *Variable* puis allez chercher la variable désirée dans « … », cliquez sur *Nouveau*, puis confirmez par *OK*.

Il est possible de faire un « Set » ou « Reset » en choisissant le qualificatif respectivement *S* ou *R*.

L'utilisation du langage Ladder permet de créer une seule transition (ou étape) utilisant une combinaison de plusieurs variables pour être validée (ou effectuant plusieurs actions).
Le principe d'écriture de ce langage est présenté en Annexe 2.
Il faut utiliser les objets de la barre d'outils suivante :

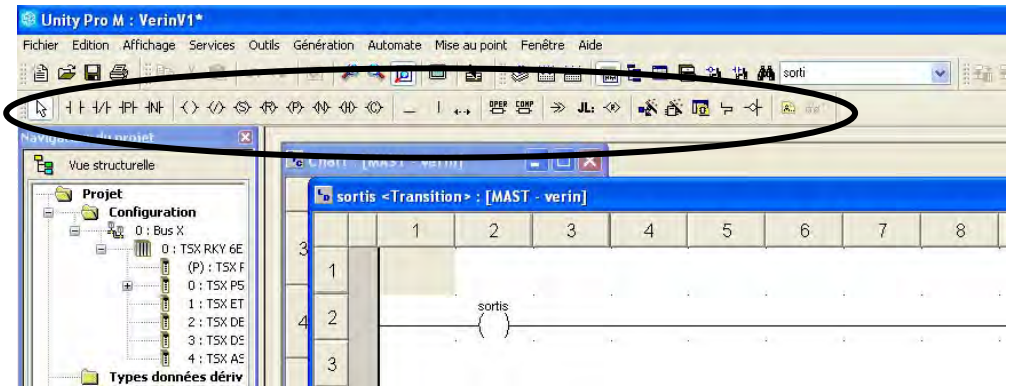


Figure 8. Ladder

V. Simulation du Grafcet

Pour pouvoir modifier les valeurs des variables, il faut d'abord créer une table d'animation :
Dans *Projet* → *Tables d'animation* → click droit, *Nouvelle table d'animation*, choisissez un nom puis valider par *OK*.

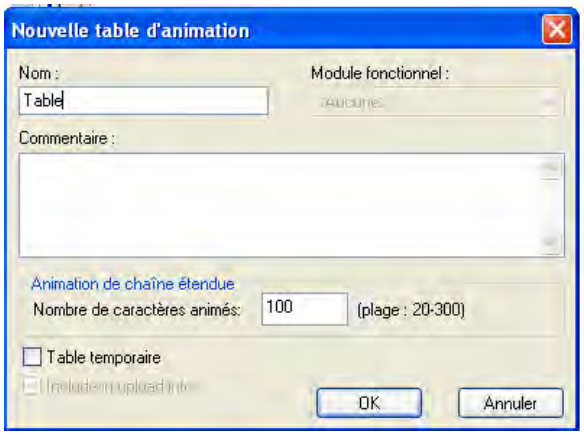


Figure 9. Création de la table d'animation

Une table comportant le nom que vous avez choisi est alors créée. Ouvrez-la et entrez les variables (précédemment crées) que vous voulez pouvoir contrôler en cliquant sur le bouton « **...** ». Dans l'exemple précédent, on inscrit les variables « rentre » et « sorti ».

Pour simuler le Grafcet, enregistrez votre projet puis faites *Génération -> Analyser le projet*.

Si votre projet ne contient pas d'erreurs, vous pouvez faire :

Automate -> Mode simulation
Génération -> Régénérer tout le projet
Automate -> Connexion
Automate -> Transférer le projet vers l'automate -> Transférer
Automate -> Exécuter -> OK

Vous pouvez alors suivre l'exécution du Grafcet et l'état de l'automate virtuel (accessible dans la barre des tâches :

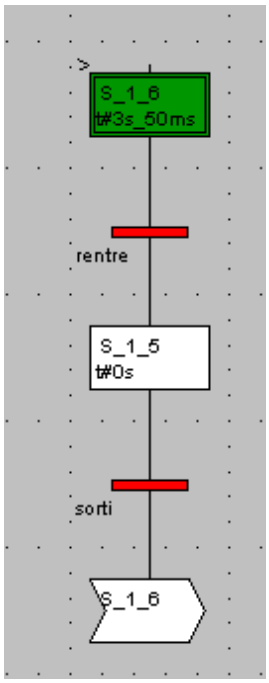


Figure 10. Grafcet en exécution

Figure 11. Etat de l'automate virtuel

Pour modifier les valeurs des variables, allez dans la table d'animation et utilisez le bouton *Forcer*, ainsi que les boutons *Force la valeur 0* et *Force la valeur 1*, … :

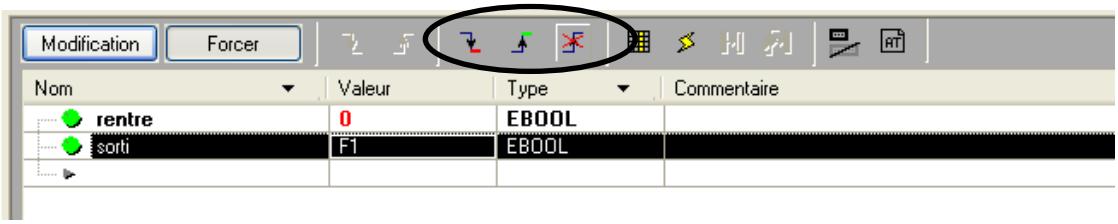


Figure 12. Forçage des variables dans la table d'animation

Vous pouvez suivre les effets de ces changements sur votre Grafcet dans le *Chart*.
Vous pouvez arrêter et reprendre la simulation, en cliquant sur *Stop* et *Run*, et déconnecter l'automate virtuel dans *Automate -> Déconnexion*.

Pour une exécution en mode « Pas à pas » :
Faites un click droit sur une étape et cliquer sur Animation -> Afficher Commande de l'animation. La fenêtre suivante apparaît :

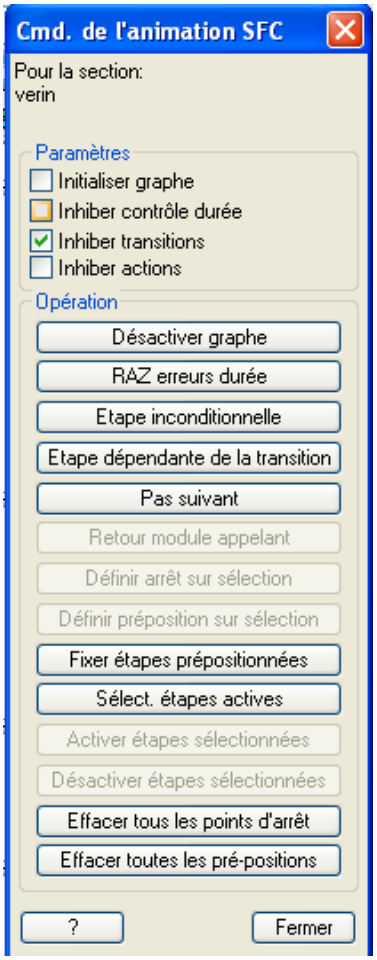


Figure13. Commande de l'animation

Cochez la case *Inhiber transitions*, le bouton *Pas suivant* apparaît, il vous permet de faire évoluer progressivement (en pas à pas) votre Grafcet (les valeurs des variables sont toujours modifiées dans la table d'animation).

Si vous voulez faire évoluer votre Grafcet jusqu'à un point choisi (sans faire du pas à pas), il faut insérer des « points d'arrêt » sur les étapes choisies pour l'arrêt : sélectionnez les étapes de votre choix (vous pouvez en sélectionner plusieurs en même temps en appuyant sur la touche *Ctrl* du clavier), puis faites un *click droit -> Animation -> Insérer/Enlever point d'arrêt*. (Cette manipulation doit être effectuée lorsque la connexion avec l'automate est établie).

VI. Exécution sur l'automate

Pour exécuter le Grafcet sur l'automate (il faut d'abord se déconnecter du mode Simulation, mettre l'automate en marche via le pupitre et vérifier que les vannes de l'air comprimé sont bien ouvertes), enregistrez votre projet puis faites *Génération -> Analyser le projet*.

Si votre projet ne contient pas d'erreurs, vous pouvez faire :

Automate -> Mode Standard
Génération -> Regénérer tout le projet
Automate -> Connexion
Automate -> Transférer le projet vers l'automate -> Transférer
Automate -> Exécuter -> OK

Vous pouvez alors suivre simultanément l'exécution du Grafcet dans le *Chart* et ses effets sur la partie opérative « Convoyeur et trieur de pièces ».

Annexe 2 : Langage à contact (Ladder)

- **Structure d'un réseau de contacts**

Un réseau s'inscrit entre deux barres de potentiel. Le sens de circulation du courant se fait de la barre de potentiel gauche à la barre de potentiel droite.

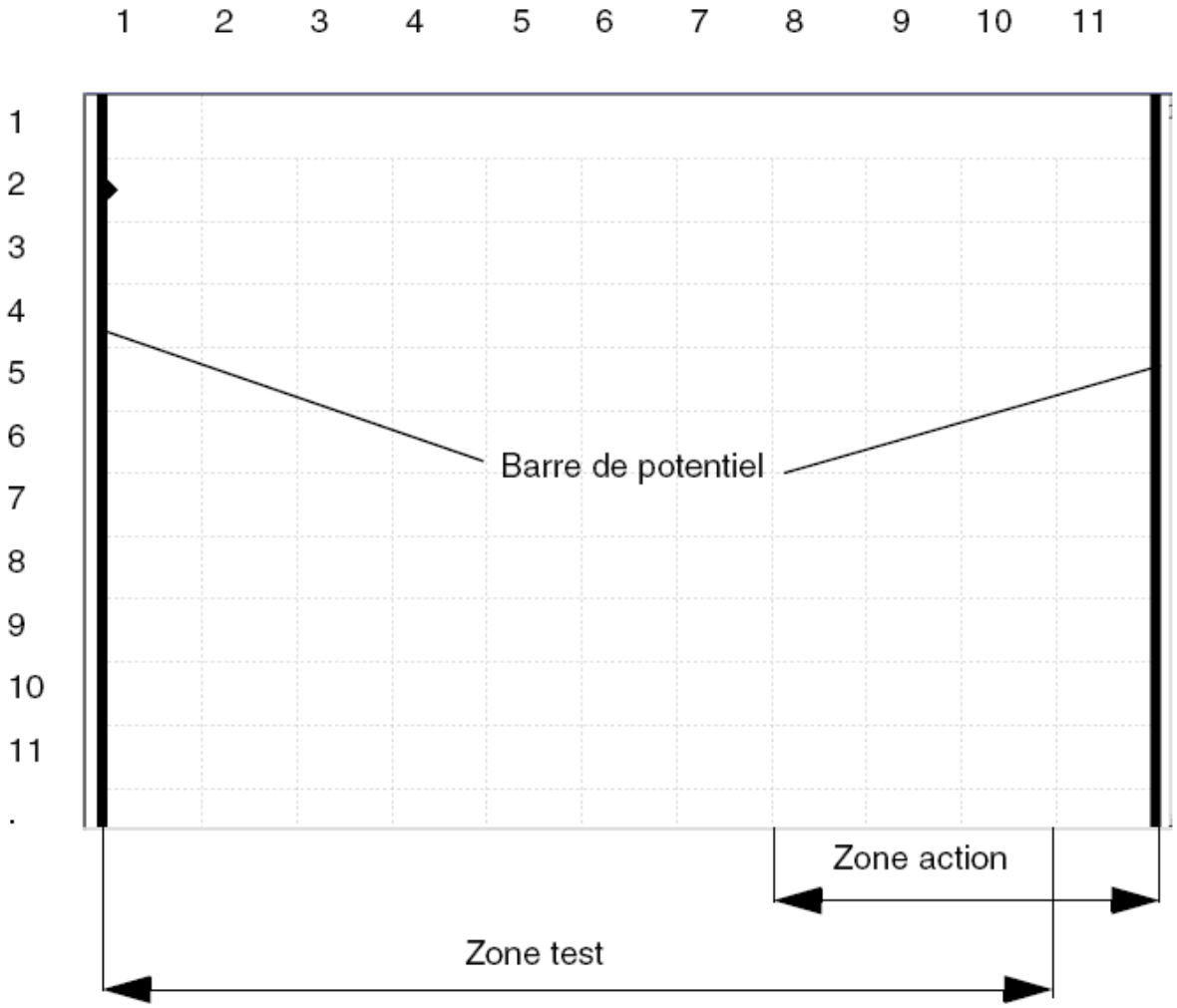


Figure 14 : structure d'un réseau de contacts

Un réseau de contacts est composé d'un ensemble d'éléments graphiques disposés sur une grille de 100 lignes maximum et 11 colonnes.
Il est réparti en 2 zones :

- la zone test dans laquelle figurent les conditions nécessaires à une action ;
- la zone action qui applique le résultat consécutif à un enchaînement de test.

- **Éléments graphiques du langage à contact**
  - **contacts**

Les éléments graphiques des contacts se programment en zone test et occupent une cellule (1 ligne de hauteur et 1 colonne de largeur).

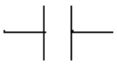
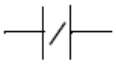
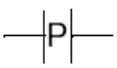
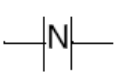
Désignation	Graphisme	Fonctions
Contact à fermeture		Contact passant quand l'objet bit qui le pilote est à l'état 1.
Contact à ouverture		Contact passant quand l'objet bit qui le pilote est à l'état 0.
Contact à détection de front montant		Front montant : détection du passage de 0 à 1 de l'objet bit qui le pilote.
Contact à détection de front descendant		Front descendant : détection du passage de 1 à 0 de l'objet bit qui le pilote.

Figure 15 : tableau des contacts

- **bobines**

Les éléments graphiques des bobines se programment en zone action et occupent une cellule (1 ligne de hauteur et 1 colonne de largeur).

Désignation	Graphisme	Fonctions
Bobine directe		L'objet bit associé prend la valeur du résultat de la zone test.
Bobine inverse		L'objet bit associé prend la valeur inverse du résultat de la zone test.
Bobine d'enclenchement		L'objet bit associé est mis à 1 lorsque le résultat de la zone test est à 1.
Bobine de déclenchement		L'objet bit associé est mis à 0 lorsque le résultat de la zone test est à 1.

Figure 16 : tableau des bobines

- **temporisateurs**

Il existe trois grands types de temporisateurs sous Unity :

- TON : permet de gérer des retards à l'activation ;
- TOF : permet de gérer des retards à la désactivation ;
- TP : permet de générer une impulsion de durée précise.

Les retards ou durées d'impulsion sont programmables.

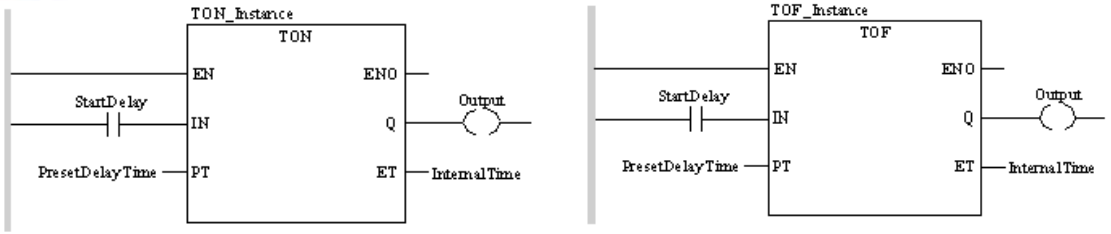


Figure 17 : Représentation graphique des temporisateurs TON et TOF

Paramètre	Type de données	Signification
IN	BOOL	Déclenchement de la temporisation
PT	TIME	Présélection du temps de retard

Figure 18 : Description des paramètres d'entrée des temporisateurs TON et TOF

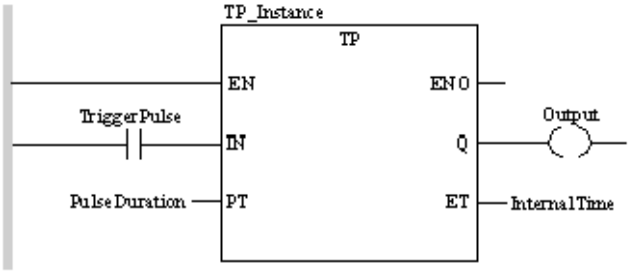


Figure 19 : Représentation graphique du temporisateur TP

Paramètre	Type de données	Signification
IN	BOOL	Déclenchement d'impulsion
PT	TIME	Présélection de la durée d'impulsion

Figure 20 : Description des paramètres d'entrée du temporisateur TP

Paramètre	Type de données	Signification
Q	BOOL	Sortie
ET	TIME	Horloge interne

Figure 21 : Description des paramètres de sortie des 3 temporisateurs

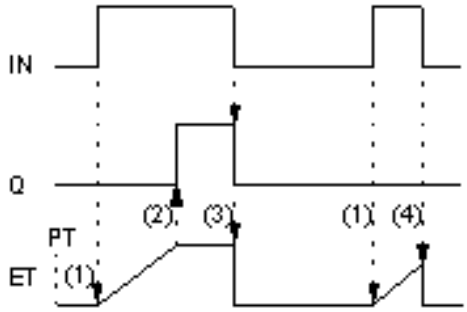
Les paramètres supplémentaires EN et ENO peuvent être configurés.

EN correspond à ENable (activer) ; il s'agit d'une entrée de bloc facultative. Lorsque l'entrée EN est activée, une sortie ENO est automatiquement établie. Si EN = 0, le bloc n'est pas activé, son programme interne n'est pas exécuté et ENO est réglé sur 0. Si EN = 1, le programme interne du bloc est exécuté et ENO est réglé sur 1. Si une erreur survient, ENO reprend la valeur 0. Si l'entrée EN n'est pas connectée, elle est automatiquement réglée sur 1.

ENO correspond à Error NOtification (notification d'erreur) ; il s'agit de la sortie associée à l'entrée facultative EN. Si ENO est réglé sur 0 (car EN = 0 ou en cas d'erreur d'exécution) :

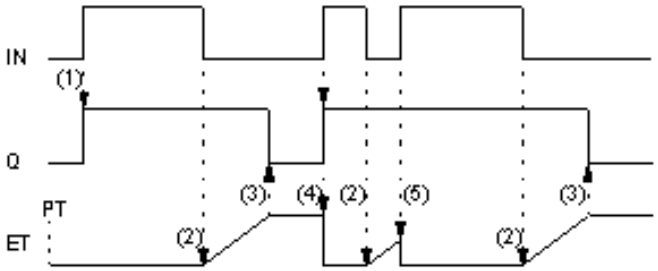
- l'état des sorties de blocs fonction reste identique à celui dans lequel elles étaient lors du dernier cycle de scrutation exécuté correctement ;
- la ou les sorties de la fonction, ainsi que les procédures, sont réglées sur 0.

Les chronogrammes des trois temporisateurs sont donnés ci-dessous :



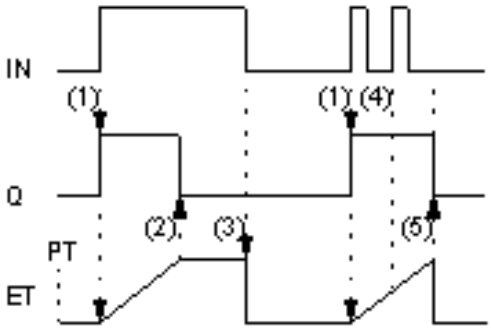
(1) Si IN passe à "1", l'horloge interne (ET) se déclenche.
(2) Si l'horloge interne atteint la valeur de PT, Q passe à "1".
(3) Si IN passe à "0", Q passe à "0" et l'horloge interne s'arrête/est remise à zéro.
(4) Si IN passe à "0" avant que l'horloge interne n'ait atteint la valeur de PT, l'horloge interne s'arrête/est remise à zéro sans que Q ne passe à "1".

Figure 22 : Chronogramme de la temporisation à l'enclenchement TON



(1) Si IN passe à "1", Q passe à "1".
(2) Si IN passe à "0", l'horloge interne (ET) se déclenche.
(3) Si l'horloge interne atteint la valeur de PT, Q passe sur "0".
(4) Si IN passe à "1", Q passe à "1" et l'horloge interne s'arrête/est remise à zéro.
(5) Si IN passe à "1" avant que l'horloge interne n'ait atteint la valeur de PT, l'horloge interne s'arrête/est remise à zéro sans que Q ne soit remis à "0".

Figure 23 : Chronogramme de la temporisation au déclenchement TOF



(1) Si IN passe à "1", Q passe à "1" et l'horloge interne (ET) se déclenche.
(2) Si l'horloge interne atteint la valeur de PT, Q est remis à "0" (indépendamment de IN).
(3) L'horloge interne s'arrête/est remise à zéro, si IN est remis à "0".
(4) Si l'horloge interne n'a pas encore atteint la valeur de PT, la présence d'un cycle d'impulsion au niveau de IN n'a aucune influence sur l'horloge interne.
(5) Si l'horloge interne a atteint la valeur de PT et que IN est à l'état "0", l'horloge interne s'arrête/est remise à zéro et Q est remis à "0".

Figure 24 : Chronogramme de l'impulsion TP

- **compteurs**

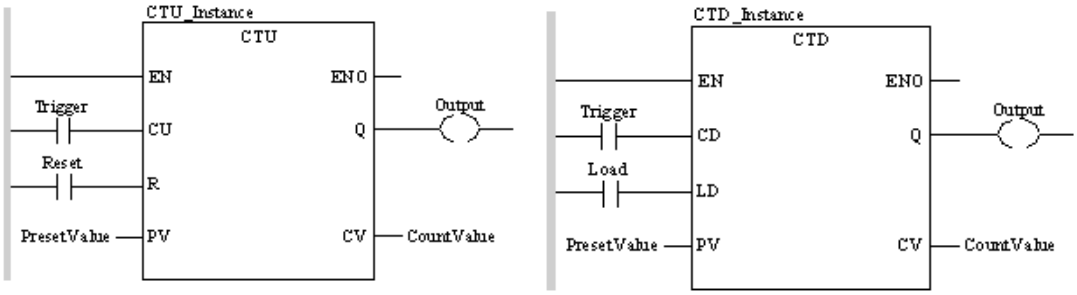


Figure 25 : Représentation graphique de CTU (compteur) et CTD (décompteur)

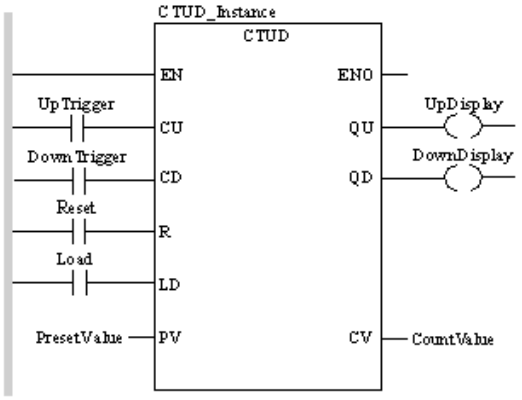


Figure 26 : Représentation graphique de CTUD (compteur-décompteur)

Paramètres	Type de données	Signification
CU	BOOL	Entrée de comptage
CD	BOOL	Entrée de comptage
R	BOOL	Reset
LD	BOOL	Chargement des données
PV	Pour CTUD : INT, Pour CTUD_*** : INT, DINT, UINT, UDINT	Valeur de présélection

Figure 27 : Paramètres d'entrée des compteurs

Paramètres	Type de données	Signification
QU	BOOL	Sortie comptage
QD	BOOL	Sortie décomptage
CV	Pour CTUD : INT Pour CTUD_*** : INT, DINT, UINT, UDINT	Valeur de comptage (valeur réelle)

Figure 28 : Paramètres de sortie des compteurs

Description du fonctionnement des compteurs :

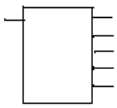
Ces blocs fonction sont utilisés pour le comptage et le décomptage.

En présence d'un signal "1" à l'entrée R, la valeur "0" est attribuée à la sortie CV. En présence d'un signal "1" à l'entrée LD, la valeur de l'entrée PV est attribuée à la sortie CV. Chaque fois que la valeur, à l'entrée CU, passe de "0" à "1", la valeur de CV est incrémentée

de 1. Chaque fois que la valeur, à l'entrée CD, passe de "0" à "1", la valeur de CV est décrémentée de 1.
En cas de présence simultanée du signal "1" aux entrées R et LD, l'entrée R est prédominante.
Si CV ≥ est égal à PV, la sortie QU passe à "1".
Si CV ≤ 0, la sortie QD passe à "1".
Note : Le compteur ne fonctionne que jusqu'aux valeurs minimum (décomptage) ou maximum (comptage) du type de données utilisé. Aucun débordement n'a lieu.
Les paramètres supplémentaires EN et ENO peuvent être configurés.

- **blocs « Opération » (Compare, Operate)**

Les éléments graphiques des blocs « opération » se programment en zone test et occupent les dimensions mentionnées ci-dessous.

Désignation	Graphisme	Fonctions
Bloc comparaison vertical		Permet la comparaison de 2 opérandes, suivant le résultat, la sortie correspondante passe à 1. Dimension : 2 colonnes/4 lignes
Bloc comparaison horizontal		Permet la comparaison de 2 opérandes, la sortie passe à 1 lorsque le résultat est vérifié (un bloc peut contenir jusqu'à 4096 caractères). Dimension : 2 colonnes/1 ligne
Bloc Opération		Réalise les opérations arithmétiques, logiques…fait appel à la syntaxe du langage littéral structuré. (Un bloc peut contenir jusqu'à 4096 caractères). Dimension : 4 colonnes/1 ligne

L'illustration suivante présente un exemple d'un réseau de contacts contenant deux blocs de comparaison et un bloc opération.