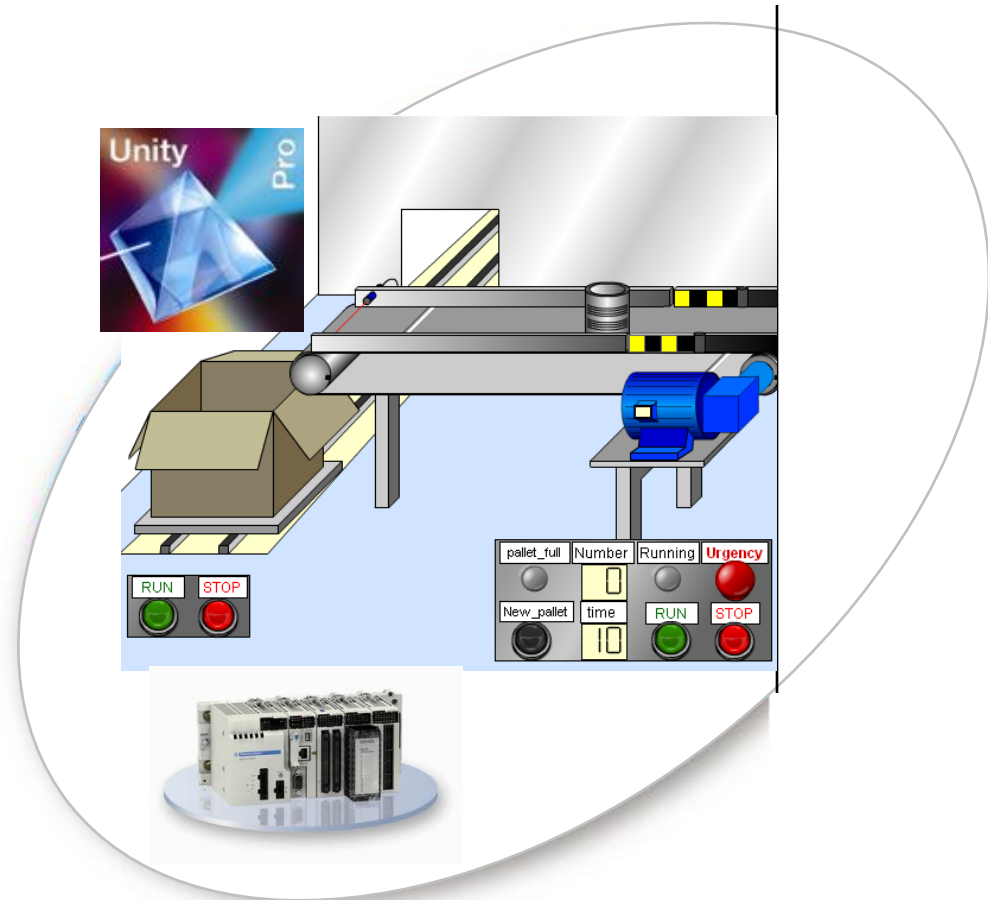


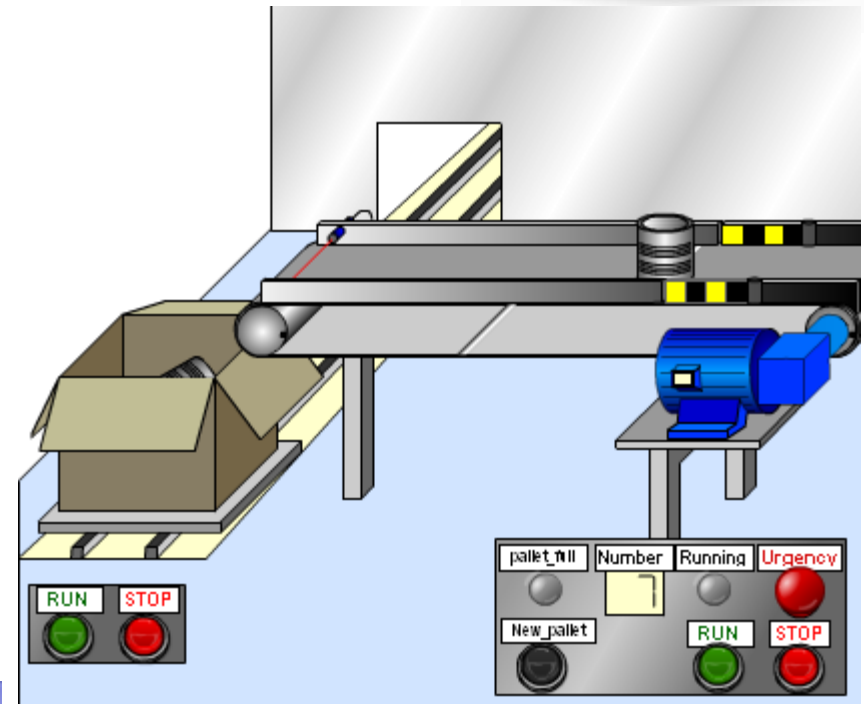
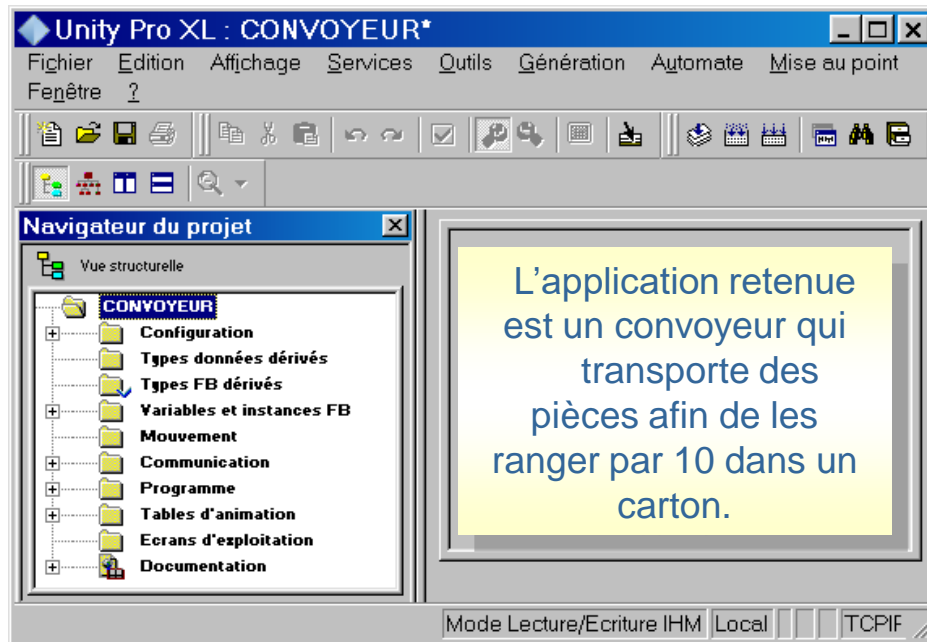
Unity Pro & Modicon M340

Prise en main Unity Pro (TPO)



Votre premier projet Unity Pro

Ce projet a pour objectif de montrer la simplicité d'usage de Unity Pro par la mise en œuvre d'une application pilotée à partir de l'automate programmable Modicon M340.



Cette première étude ne portera que sur le convoyeur.

Programmation du convoyeur en langage LD

Cahier des charges du convoyeur

Cahier des charges

Analyse

Création du projet

Déclaration des données

Programmation en LD

Génération du code

Mise au point

Personnalisation

L'installation comprend un convoyeur et un pupitre de commande.

■ Les besoins en entrées :

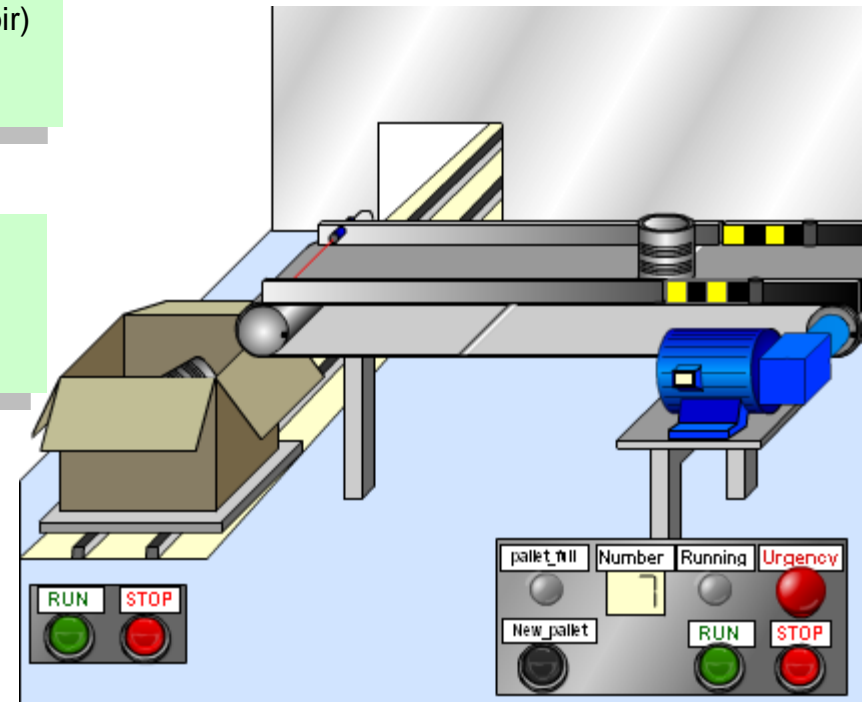
- Une entrée **RUN** de mise en marche du convoyeur (Bouton Poussoir)
- Une entrée **STOP** d'arrêt du convoyeur (Bouton Poussoir)
- Une entrée **URGENCY** d'arrêt d'urgence (Interrupteur)

■ Les besoins en sorties :

- Une sortie commande moteur **MOTOR** (Contacteur)
- Une sortie voyant **RUNNING** (Voyant Visu Mise en marche Convoyeur)

Remarque :

Dans cette première phase nous définissons les informations d'entrées/sorties nécessaires sans affecter pour l'instant des entrées/sorties réelles (pas d'adresses physiques), le projet étant testé sur le simulateur de Unity Pro.





Votre premier projet Unity Pro

Méthodologie de développement

Cahier des charges

Analyse

Création du projet

Déclaration des données

Programmation en LD

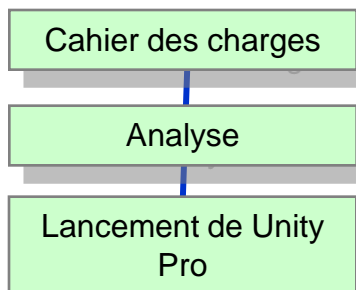
Génération du code

Mise au point

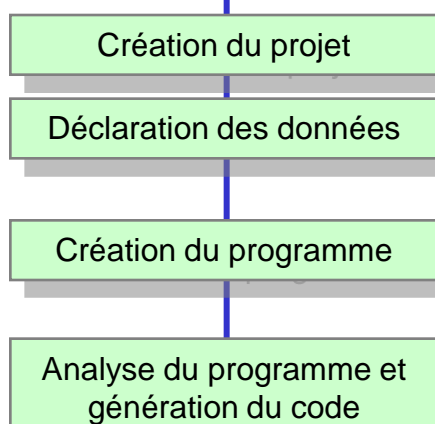
Configuration API

La chronologie de développement est la suivante :

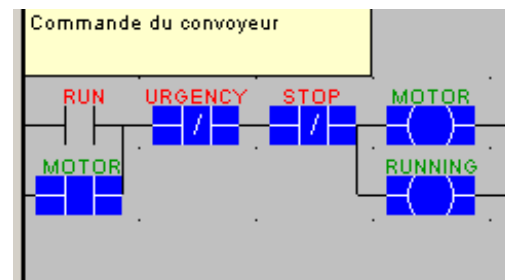
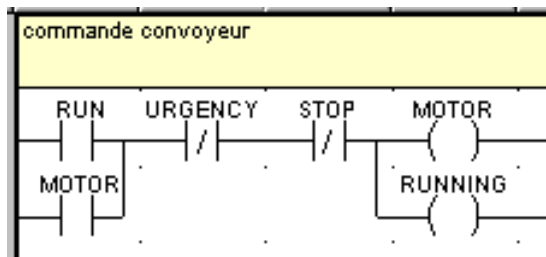
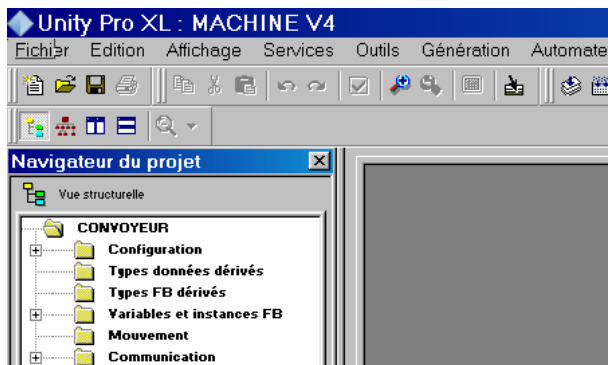
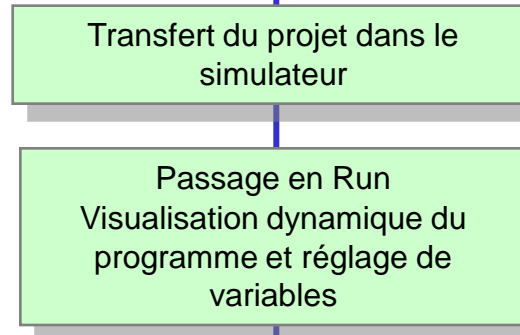
Prise en charge de l'application



Réalisation du projet



Mise au point du projet



Programmation du convoyeur en langage LD

Analyse du cahier des charges du Convoyeur

Cahier des charges

Analyse

Création du projet

Déclaration des données

Programmation en LD

Génération du code

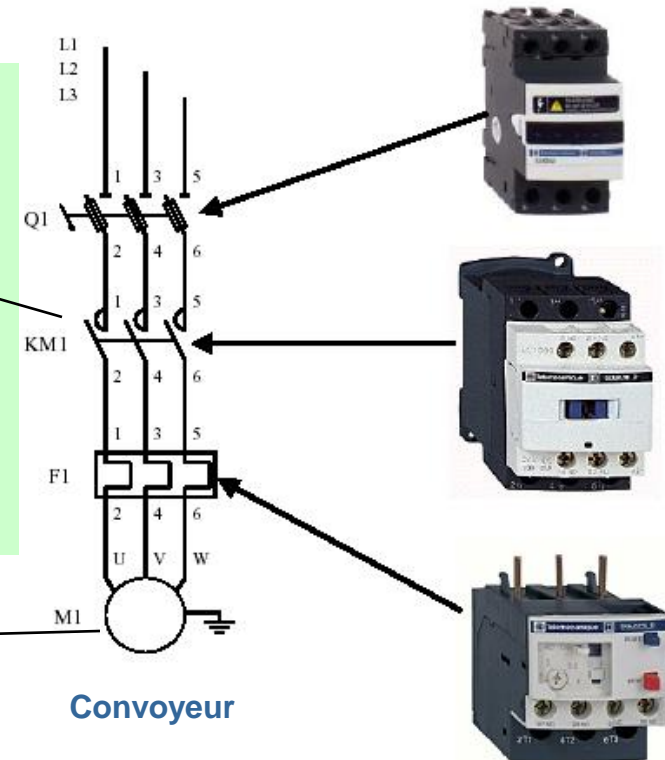
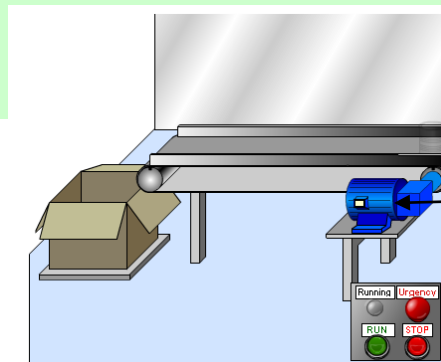
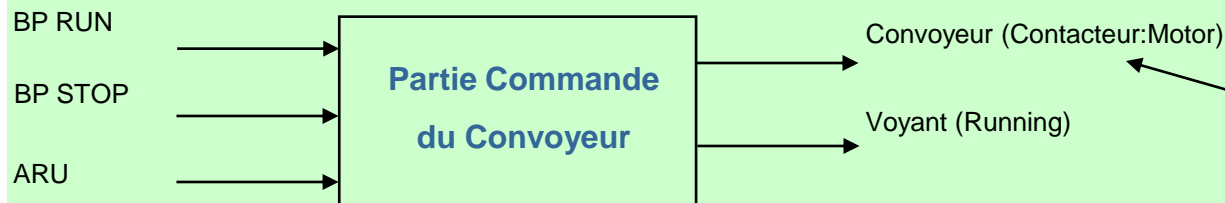
Mise au point

Configuration API

▶ Cette phase consiste à déterminer la Partie Commande du convoyeur c'est-à-dire sa logique de commande.

ENTREES

SORTIES



Programmation du convoyeur en langage LD

Analyse du cahier des charges du convoyeur

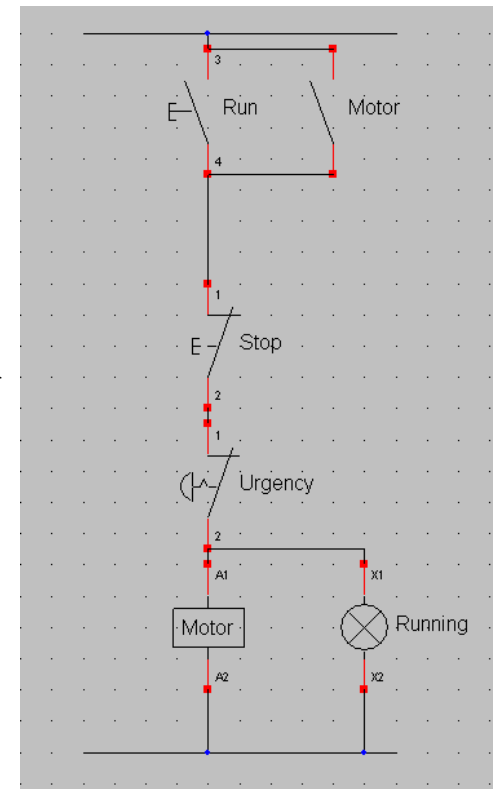
La solution est réalisée en schéma à contacts.

Structure du fonctionnement :

Le convoyeur démarre lorsque l'utilisateur appuie sur le bouton poussoir RUN et s'il n'y a pas d'arrêt d'urgence.

Le convoyeur s'arrête lorsque l'utilisateur appuie sur le bouton poussoir STOP ou sur l'arrêt d'urgence.

L'état du convoyeur sera signalé par un voyant Running.

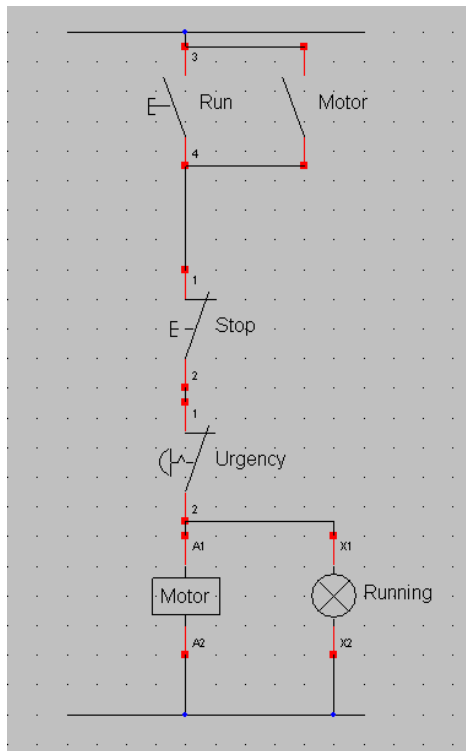


Programmation du convoyeur en langage LD

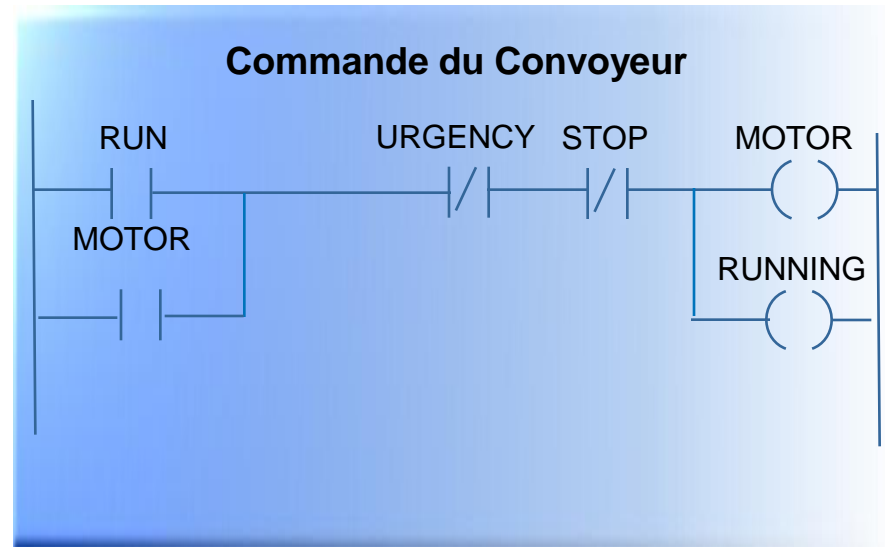
Analyse du cahier des charges du convoyeur

Le programme sera réalisé en langage Ladder (LD), qui est un langage graphique développé pour les électriciens proche du langage à contacts.

Schéma à contacts



Structure du Programme en LD

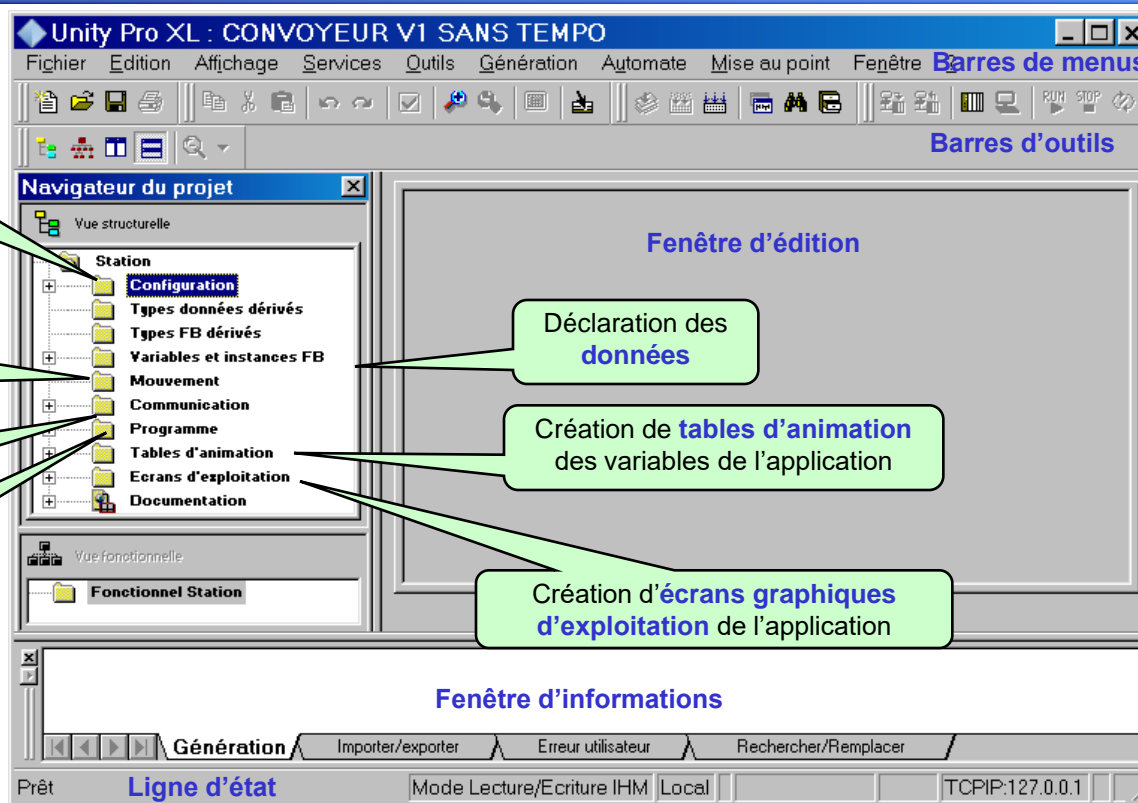


Votre premier projet Unity Pro

Introduction à Unity Pro

Programmer en LD

Unity Pro permet de programmer les automates Modicon M340, Premium, Atrium, Quantum. Le navigateur application visualise l'organisation du projet et donne l'accès aux éditeurs. Les fenêtres sont repositionnables sur l'écran et peuvent être affichées selon plusieurs modes (pleine page, réduit)



Programmation du convoyeur en langage LD

Création du projet (1/3)

Cahier des charges

Analyse

Création du projet

Déclaration des données

Programmation en LD

Génération du code

Mise au point

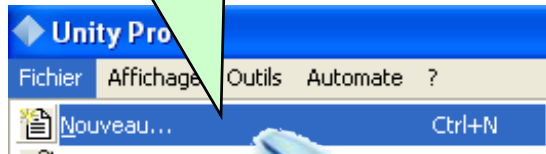
Configuration API

A l'aide du logiciel Unity Pro nous allons maintenant :

- Créer un nouveau projet intitulé « Machine »
- Définir la base de l'automate

Sélectionner le menu
Fichier/Nouveau.

1

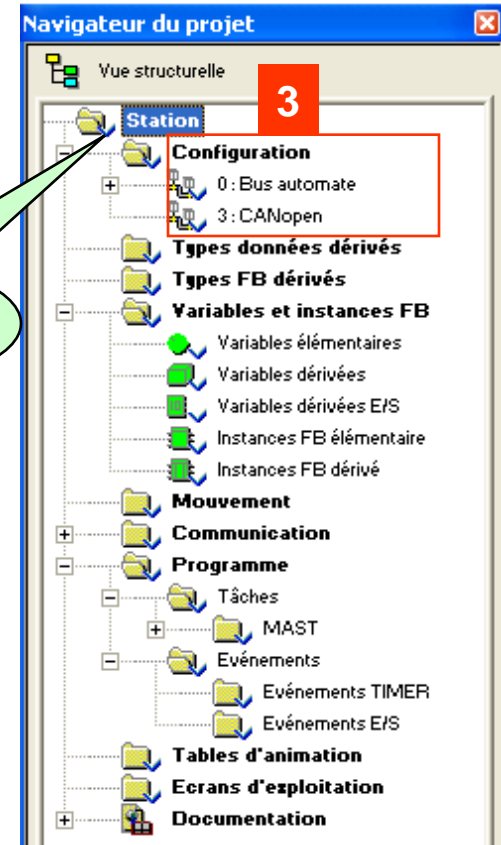


2

Automate	on OS min.	Description
Modicon M340		
BMX P34 1000	01.00	CPU 340-10 Modbus
BMX P34 2010	01.00	CPU 340-20 Modbus CANopen
BMX P34 2020	01.00	CPU 340-20 Modbus Ethernet
BMX P34 2030	01.00	CPU 340-20 Ethernet CANopen
Premium		

Sélectionner la base automate :
BMX P34 2010 (par exemple) et valider par OK.
Le navigateur présente la structure d'une application.

La structure du projet est créée.



Programmation du convoyeur en langage LD

Création du projet (2/3)

Cahier des charges

Analyse

Création du projet

Déclaration des données

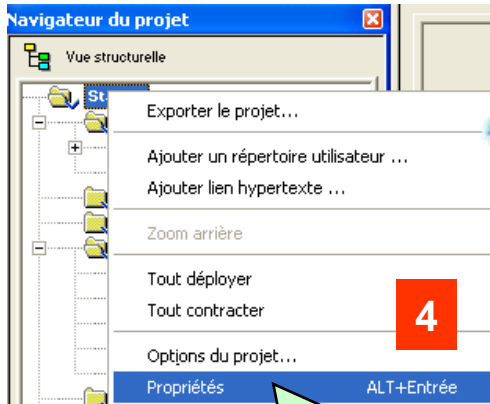
Programmation en LD

Génération du code

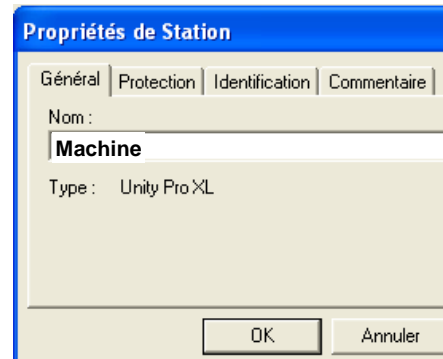
Mise au point

Configuration API

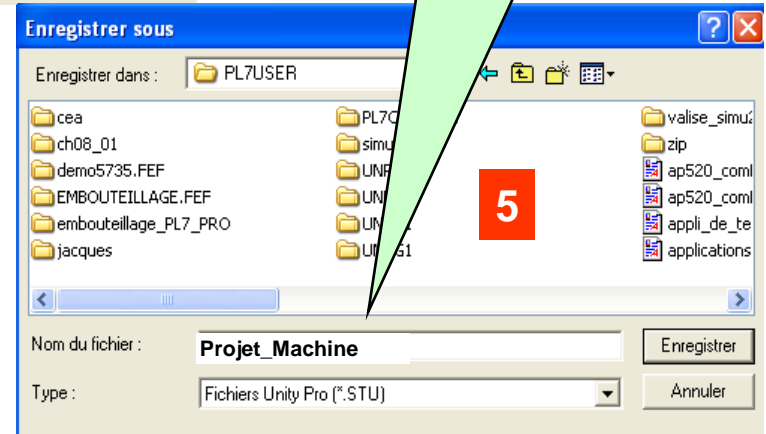
Donner un nom et enregistrement du projet :



Effectuer un clic droit sur **Projet** et sélectionner le menu **Propriétés**, afin d'indiquer le **nom** et saisir le **commentaire** du projet.



Sauvegarder le projet à l'aide du menu **Fichier/Enregistrer** (Fichier *.STU).



Programmation du convoyeur en langage LD

Création du projet (3/3)

Cahier des charges

Analyse

Création du projet

Déclaration des données

Programmation en LD

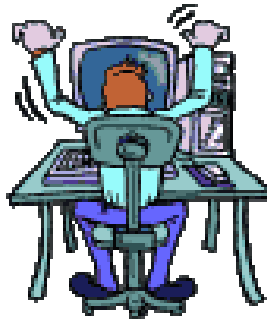
Génération du code

Mise au point

Configuration API

■ A vous de jouer...

Lancez le logiciel Unity Pro (Control Expert aujourd'hui) et réalisez les opérations de création du projet.



Programmation du convoyeur en langage LD

Déclaration des données (1/2)

Cahier des charges

Analyse

Création du projet

Déclaration des données

Programmation en LD

Génération du code

Mise au point

Configuration API

La déclaration des données peut se faire :

- soit à partir de l'éditeur de données,
- soit au fil de l'eau, lors de la saisie du programme.

Nous allons déclarer les données d'entrées relatives au programme Convoyeur dans l'**éditeur de données**.
Les données de sortie seront déclarées au fil de l'eau lors de la création du programme.

Entrées

Nom	Type	Commentaire
RUN	EBOOL	Départ convoyeur
STOP	EBOOL	Arrêt convoyeur
URGENCY	EBOOL	Arrêt d'urgence

Sorties

Nom	Type	Commentaire
MOTOR	EBOOL	Commande Moteur convoyeur
RUNNING	EBOOL	Voyant Marche/Arrêt du moteur



Programmation du convoyeur en langage LD

Déclaration des données (2/2)

Cahier des charges

Analyse

Création du projet

Déclaration des données

Programmation en LD

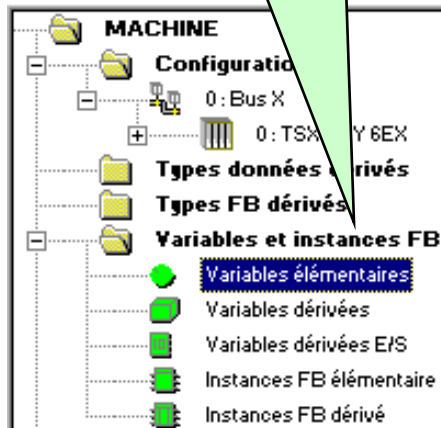
Génération du code

Mise au point

Configuration API

Sur l'atelier Unity Pro, vous pouvez utiliser des variables en déclarant uniquement le nom et le type mais sans déclarer d'adresse : ces variables sont non localisées, c'est le système qui attribue de manière interne ces adresses.

Effectuer un double clic sur **Variables élémentaires** pour accéder à l'éditeur de données.



1

Nom	Type	Adresse	Valeur	Commentaire
RUN	BOOL			

2

Indiquer :
Le **nom de la variable**
Le **type de la variable** : EBOOL
Le **commentaire** de la variable.

3

Déclarer toutes les variables suivantes

Nom	Type	Adresse	Valeur	Commentaire
RUN	EBOOL			Départ convoyeur
STOP	EBOOL			Arrêt convoyeur
URGENCY	EBOOL			Arrêt d'urgence

Remarque : les autres données seront déclarées au fil de l'eau lors de la création du programme LD

Programmation du convoyeur en langage LD

Création de la section convoyeur (1/3)

Cahier des charges

Analyse

Création du projet

Déclaration des données

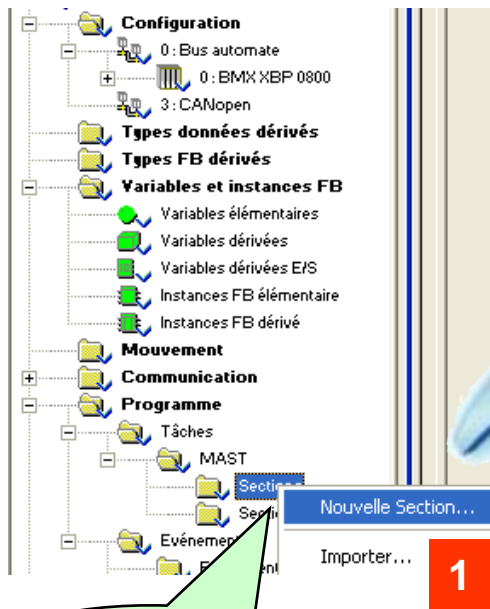
Programmation en LD

Génération du code

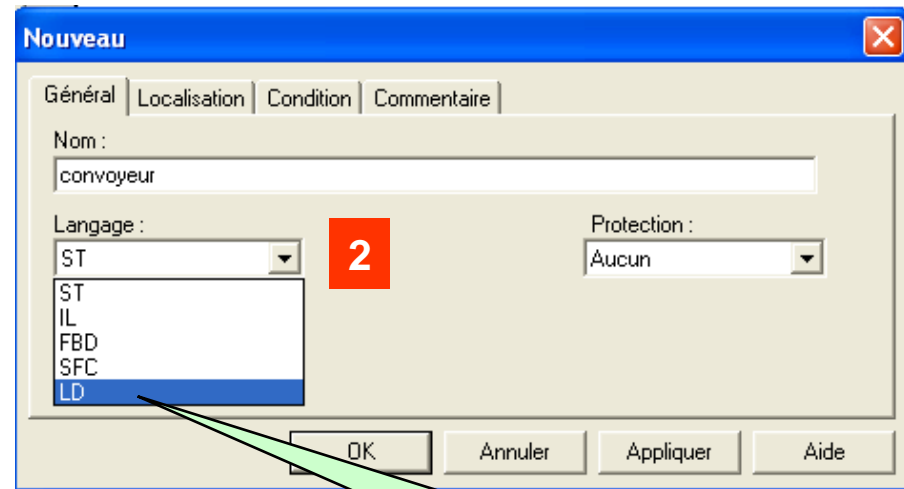
Mise au point

Configuration API

Une projet Unity Pro peut comporter plusieurs tâches (tâche maître créée par défaut et qui représente la tâche principale, des tâches événementielles, ...). Les tâches sont composées de sections et sous-programmes. L'ordre des sections détermine l'ordre de scrutation du programme.



Effectuer un clic droit sur **Section** et sélectionner le menu **Nouvelle section**.



Saisir le **Nom de la section** et sélectionner le **langage LD** puis valider par OK.

Programmation du convoyeur en langage LD

Création de la section convoyeur (2/3)

Cahier des charges

Analyse

Création du projet

Déclaration des données

Programmation en LD

Génération du code

Mise au point

Configuration API

Construction du réseau de contacts en utilisant les variables déjà déclarées

1 Sélection du **type d'objet**.

2 Poser l'**objet** sur la cellule désirée.

3 Effectuer un **double clic** pour renseigner le contact

4 Cliquer sur **...** pour faire apparaître la **liste des variables déjà déclarées**

5 Sélectionner l'**objet** dans la liste en double cliquant (Veiller à ce que la case "dans structure" ne soit pas cochée).

Editeur LD : Sélection d'instance

RUN

Variables | Blocs fonction

Nom	Type	Commentaire
<input checked="" type="checkbox"/> RUN	EBOOL	Départ convoyeur
<input checked="" type="checkbox"/> STOP	EBOOL	Arrêt convoyeur
<input checked="" type="checkbox"/> URGENCY	EBOOL	Arrêt d'urgence

Remarque : L'écran de saisie est divisé en cellules recevant les différents objets. Un survol avec la souris donne la signification de l'objet.

Programmation du convoyeur en langage LD

Création de la section convoyeur (3/3)

Cahier des charges

Analyse

Création du projet

Déclaration des données

Programmation en LD

Génération du code

Mise au point

Configuration API

Création du réseau de contacts avec déclaration des variables au fil de l'eau

Effectuer un **double clic** pour renseigner l'objet

Renseigner l'objet et valider par OK.

Sélectionner le type d'objet et le positionner.

Indiquer le **type d'objet** et valider.

1

2

3

4

Remarque : Le type d'objet proposé est toujours en cohérence avec l'objet sélectionné.

Programmation du convoyeur en langage LD

Analyse et première génération du projet (1/3)

Cahier des charges

Analyse

Création du projet

Déclaration des données

Programmation en LD

Génération du code

Mise au point

Configuration API

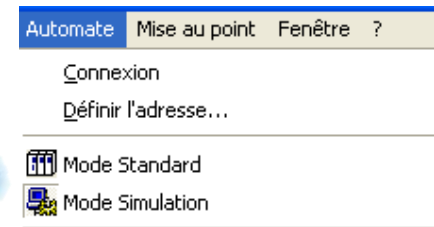
La saisie d'un programme étant terminée, nous allons effectuer l' **Analyse du projet** (signaler les erreurs et avertissement dans le projet) puis la **Regénération du projet** (indispensable la première fois).

Exécution du programme sur le simulateur

Le projet peut être exécuté sur :

- L'automate et il faut dans ce cas définir la configuration.
- Un simulateur de l'automate et dans ce cas la définition de la configuration n'est pas nécessaire.

Lors de l'analyse et de la génération de code, Unity Pro tient compte de la cible automate ou simulateur



Remarque

Le simulateur d'automate permet de simuler un projet dans son ensemble avec toutes les tâches utilisateur associées. Cependant, la dynamique de comportement d'exécution du simulateur ne peut pas être comparée à celle d'un automate réel.

Programmation du convoyeur en langage LD

Analyse et première génération du projet (2/3)

Cahier des charges

Analyse

Création du projet

Déclaration des données

Programmation en LD

Génération du code

Mise au point

Configuration API

Analyse du projet.

Affichage des erreurs et avertissements avec navigation vers la cause d'origine.

1 Lancer l'analyse du projet.

Unity Pro XL : CONVOYEUR*

Fichier Edition Affichage Services Outils Génération Automate Mise au point

Analyser Ctrl+Maj+B

Analyser le projet

Générer le projet Ctrl+B

Regénérer tout le projet

Navigateur du projet

Vue structurelle

Configuration

2 Affichage du résultat de l'analyse du projet.

Analyse en cours...
{CONVOYEUR : [MAST]}
{CONVOYEUR : [MAST]}
Processus réussi : 0 Erreur(s) , 1 Avertissement(s)

[! : 3, c: 4] W1072 variable attendue
0 erreur(s), 1 avertissement(s)

Un **avertissement** signale un élément pouvant poser problème mais n'empêche pas le transfert dans le simulateur ou l'automate. Une **erreur** bloque tout transfert.

3 Effectuer un double clic sur l'élément signalé en **bleu** ou en **rouge**, Unity Pro se positionne automatiquement sur l'élément en cause.

Programmation du convoyeur en langage LD

Analyse et première génération du projet (3/3)

Cahier des charges

Analyse

Création du projet

Déclaration des données

Programmation en LD

Génération du code

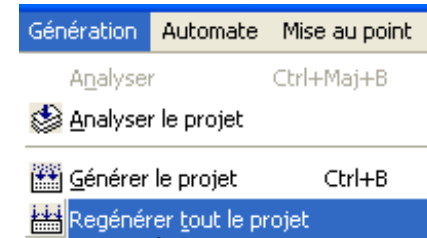
Mise au point

Configuration API

La première génération nécessite une régénération complète du projet. Par la suite, il est nécessaire de générer uniquement les modifications.



Sélectionner la **cible d'exécution** du programme par le menu **Automate / Mode simulation**.



Sélectionner le menu **Génération / Regénérer le projet**.

```
Analyse en cours...
{a : [MAST]} : 0 erreur(s), 0 avertissement(s)
Génération en cours...
{Sous-équipement [0.0:C3.M3] CANopen comm head}
{Sous-équipement [0.0:C3.M3] CANopen comm head}
{Sous-équipement [0.0:C3.M3] CANopen comm head}
{Sous-équipement [0.0:C3.M3] CANopen comm head}
Edition des liens en cours...
Processus réussi : 0 Erreur(s) , 4 Avertissement(s)
```

3

Le nombre de %MW réservés en e
Le nombre de %MW réservés en s
Le nombre de %M réservés en ent
Le nombre de %M réservés en sor

Affichage des **avertissements ou erreurs** éventuelles

Remarque : Les avertissements sont dus au fait que le bus CAN Open n'est pas configuré

Programmation du convoyeur en langage LD

Mise au point du projet (1/6)

Cahier des charges

Analyse

Création du projet

Déclaration des données

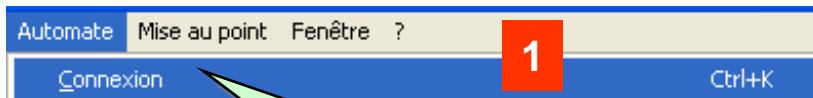
Programmation en LD

Génération du code

Mise au point

Configuration API

L'analyse étant correcte, nous allons mettre au point l'application à l'aide du simulateur automate en nous connectant dans un premier temps avec celui-ci.

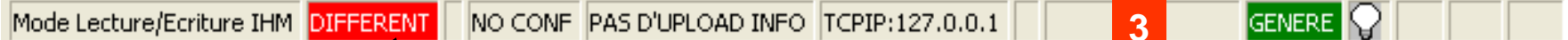


Sélectionner le menu **Automate / Connexion**
Le bandeau inférieur affiche l'état du simulateur.

2



Le simulateur se lance



Dans le bandeau, il est indiqué que **le projet ouvert dans Unity Pro et celui dans le simulateur sont différents.**

Remarque : Le ? dans la barre des tâches signale que le simulateur est lancé **sans projet valide.**

Programmation du convoyeur en langage LD

Mise au point du projet (2/7)

Cahier des charges

Analyse

Création du projet

Déclaration des données

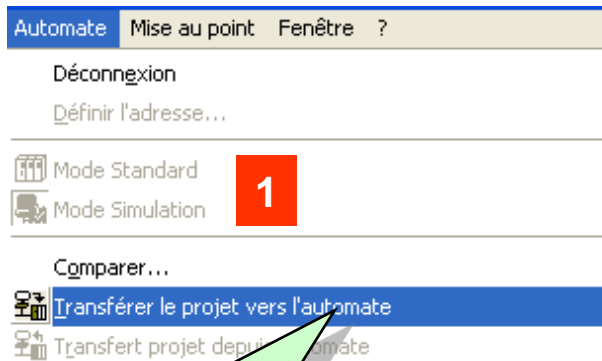
Programmation en LD

Génération du code

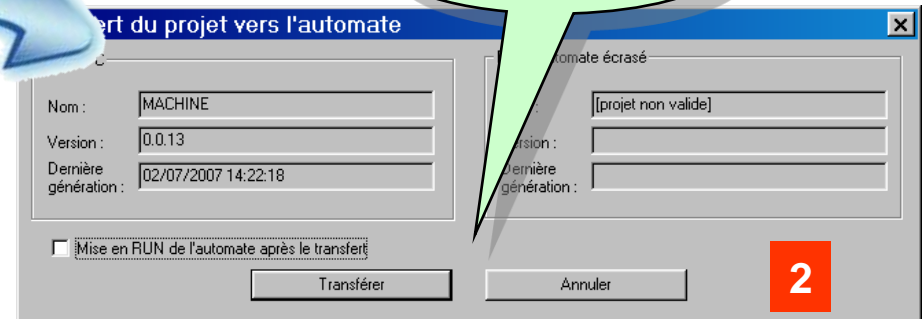
Mise au point

Configuration API

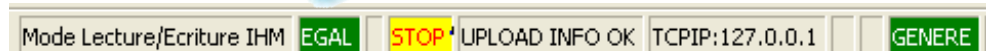
La connexion étant réalisée, nous pouvons transférer le projet dans le simulateur.



Sélectionner le menu
**Automate / Transférer le
projet vers l'automate.**



3



Le bandeau indique que **les
programmes sont identiques**
mais que l'automate est en
STOP.

Programmation du convoyeur en langage LD

Mise au point du projet (3/7)

Cahier des charges

Analyse

Création du projet

Déclaration des données

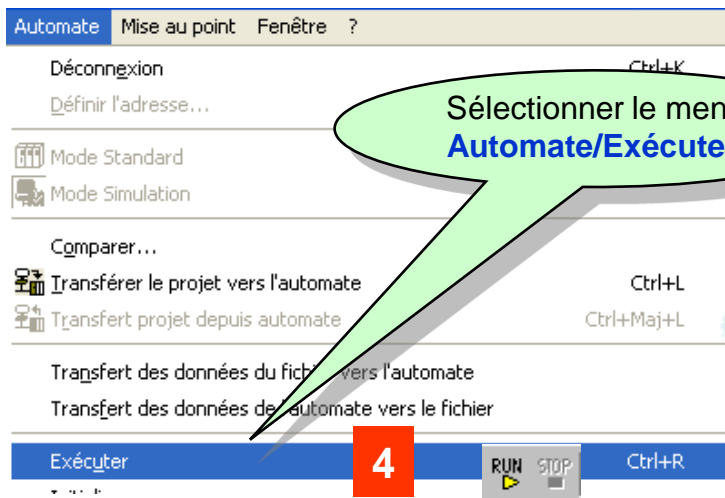
Programmation en LD

Génération du code

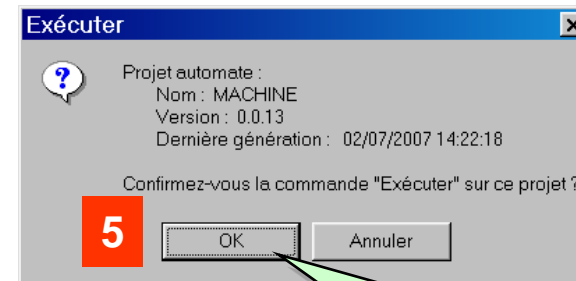
Mise au point

Configuration API

La connexion étant réalisée, nous pouvons exécuter le programme convoyeur dans le simulateur.



Sélectionner le menu **Automate/Exécuter.**



Cliquer sur Ok pour passer l'automate en RUN.

6

Mode Lecture/Ecriture IHM EGAL **RUN** UPLOAD INFO OK TCP/IP:127.0.0.1 GENERE

Le bandeau nous indique que l'automate est en **RUN**.

Programmation du convoyeur en langage LD

Mise au point du projet (4/7)

Cahier des charges

Analyse

Création du projet

Déclaration des données

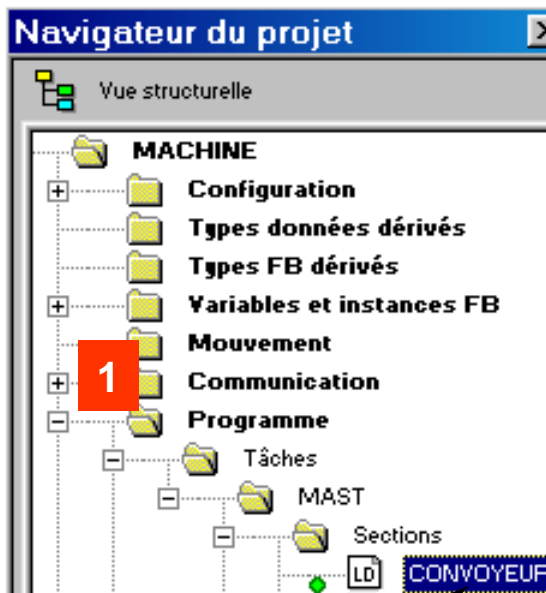
Programmation en LD

Génération du code

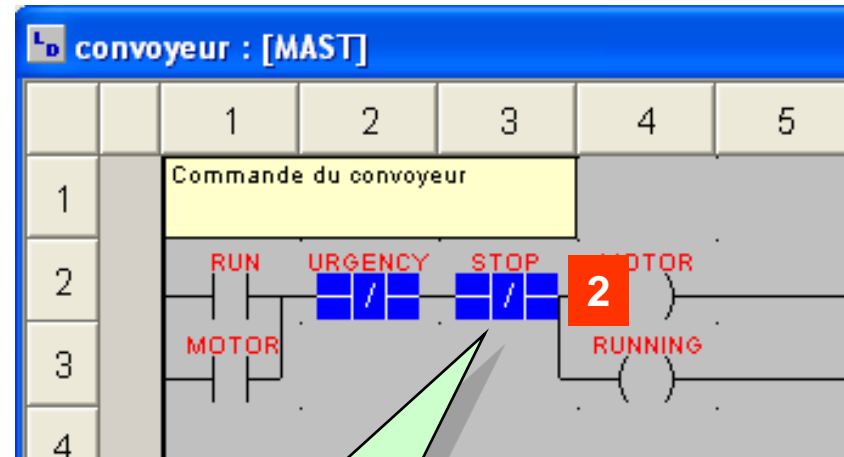
Mise au point

Configuration API

► Nous allons pouvoir visualiser le programme en dynamique et modifier les variables pour simuler le fonctionnement du convoyeur.



Effectuer un double clic sur la **section Convoyeur**



La section Convoyeur apparaît **en dynamique**.
Les contacts passants sont **en vidéo inverse** (sur fond bleu).

Programmation du convoyeur en langage LD

Mise au point du projet (5/7)

Cahier des charges

Analyse

Création du projet

Déclaration des données

Programmation en LD

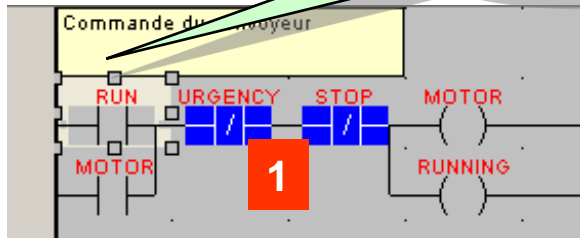
Génération du code

Mise au point

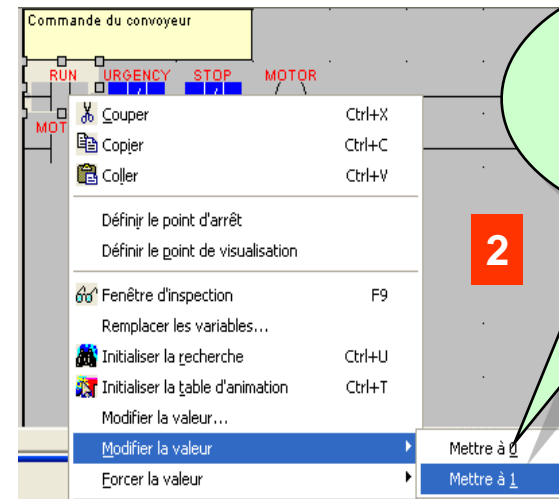
Configuration API

Modification des variables depuis l'écran de visualisation du schéma en ladder.

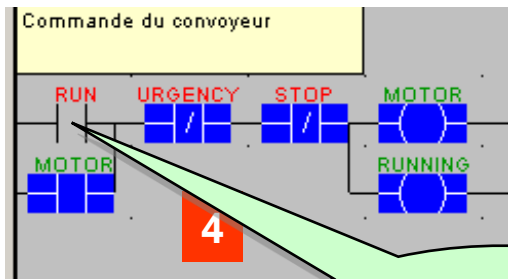
Sélectionner la variable RUN.



Effectuer un clic droit puis sélectionner le menu **Modifier la valeur** et Mettre la valeur à 1.



Remettre à 0 la commande RUN (En réitérant 1 et 2)



Le moteur a démarré et le voyant de marche est allumé.



Programmation du convoyeur en langage LD

Mise au point du projet (6/7)

Cahier des charges

Analyse

Création du projet

Déclaration des données

Programmation en LD

Génération du code

Mise au point

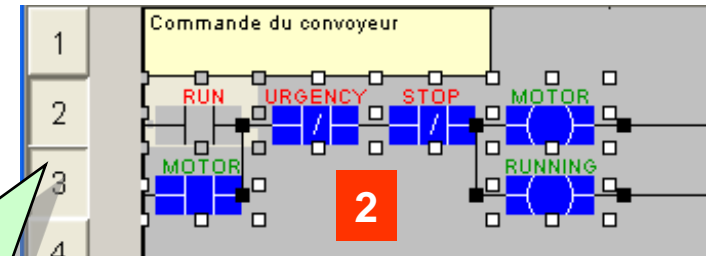
Configuration API

Initialisation d'une table d'animation pour visualiser l'état des variables de la section convoyeur.

Sélectionner la case 2.



Sélectionner la case 3 par sélection multiple (Shift) ou par le raccourci CRT A.

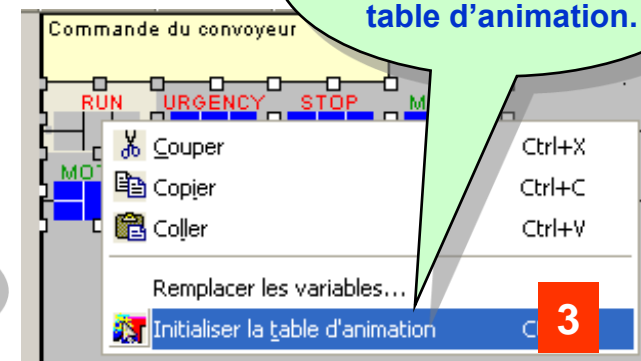


Effectuer un clic droit puis sélectionner le menu Initialiser la table d'animation.

Table[Editeur LD - convoyeur : [MAST]]

Nom	Valeur	Type	Commentaire
URGENCY	0	EBOOL	Arrêt d'urgence
STOP	0	EBOOL	Arrêt convoyeur
MOTOR	1	EBOOL	Commande du moteur
RUNNING	1	EBOOL	
RUN	0	EBOOL	

La table d'animation apparaît avec les variables sélectionnées.



Programmation du convoyeur en langage LD

Mise au point du projet (7/7)

Cahier des charges

Analyse

Création du projet

Déclaration des données

Programmation en LD

Génération du code

Mise au point

Configuration API

Modification des variables depuis la table d'animation.

Nom	Valeur
URGENCY	0
STOP	0
MOTOR	1
RUNNING	1
RUN	0

Sélectionner le bouton Modification.

Nom	Valeur	Type	Commentaire
URGENCY	0	EBOOL	Arrêt d'urgence
STOP	0	EBOOL	Arrêt convoyeur
MOTOR	1	EBOOL	Commande du moteur
RUNNING	1	EBOOL	Voyant marche moteur
RUN	0	EBOOL	Départ convoyeur

Sélectionner la variable STOP.

Nom	Valeur
URGENCY	0
STOP	1
MOTOR	0
RUNNING	0
RUN	0

La variable STOP est à 1 et le moteur s'arrête.

Nom	Valeur
URGENCY	0
STOP	0
MOTOR	1
RUNNING	1
RUN	0

Cliquer sur l'icône de mise à 1.

Sauvegarder le Projet.

Fin de la phase 1: Programmer en LD.

Programmation du palettiseur en langage ST

Les écrans d'exploitation (1/4)

Cahier des charges

Analyse

Déclaration des données

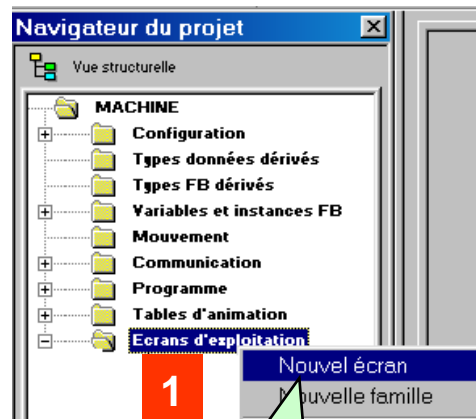
Programmation en ST

Mise au point

Les écrans d'exploitation

Configuration API

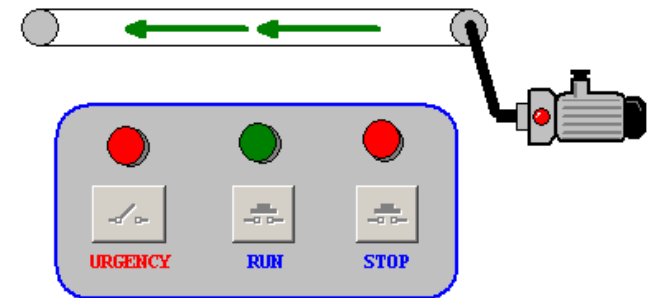
Unity Pro propose des écrans d'exploitation destinés à faciliter l'exploitation d'un procédé automatisé. Ces écrans peuvent être construits en mode local ou connecté. Nous allons créer un écran associé à la machine en mode connecté.



Sélectionner dans le navigateur **Ecrans d'exploitation** le menu **Nouvel écran**.

Saisir le nom de l'écran et valider par OK.

The 'Propriétés des écrans' dialog box is shown with the 'Général' tab selected. The 'Nom' field contains 'convoyeur' and the 'Valeur' field contains '0'. A red box with the number '2' is placed over the 'Valeur' field. The 'Emplacement' section has 'Famille' and 'Module fonctionnel' both set to '<Aucun>'. The 'OK' button is highlighted.



Programmation du palettiseur en langage ST

Les écrans d'exploitation (2/4)

Cahier des charges

Analyse

Déclaration des données

Programmation en ST

Mise au point

Les écrans d'exploitation

Configuration API

Création du contenu de l'écran Convoyeur.

L'écran de saisie propose un ensemble d'objets graphiques auxquels peuvent être associées des variables d'animation. Le principe de saisie est le suivant :

The diagram illustrates the process of creating and configuring a rectangle object in a software interface. It is divided into four numbered steps:

- 1** Sélection du type d'objet. (Selection of the object type.)
- 2** Dessiner l'objet à l'aide de la souris. (Draw the object using the mouse.)
- 3** Effectuer un double clic pour accéder aux propriétés de l'objet. (Double-click to access the object properties.)
- 4** Renseigner les propriétés de l'objet. (Specify the object properties.)

The interface shown includes a toolbar with icons for drawing shapes (Rectangle, Circle, Triangle, etc.), text formatting (Aa), and alignment (Aligner vers le haut). A 'Propriétés de l'objet : Rectangle' dialog box is open, showing the 'Dessin' (Drawing) tab with options for 'Tracé' (Stroke), 'Epaisseur' (Thickness), 'Motif' (Pattern), and 'Arrondir les coins' (Round corners). The 'Couleur du tracé' (Stroke color) and 'Couleur de fond' (Fill color) sections are highlighted, with a callout indicating 'Renseigner les propriétés de l'objet'.

Onglet Dessin : Modification de la couleur de l'objet
Onglet Type d'animation : Choix du type d'animation et condition d'affichage
Onglet Animation : Choix de la variable d'animation et condition d'affichage

Programmation du palettiseur en langage ST

Les écrans d'exploitation (3/4)

Cahier des charges

Analyse

Déclaration des données

Programmation en ST

Mise au point

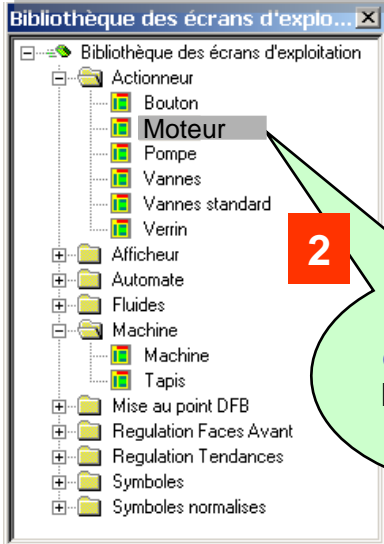
Les écrans d'exploitation

Configuration API

Création du contenu de l'écran Convoyeur : Objets issus de la bibliothèque.
Unity propose une bibliothèque d'objets prédéfinis : actionneurs, afficheurs, automates, machines.
Le principe de saisie est le suivant :

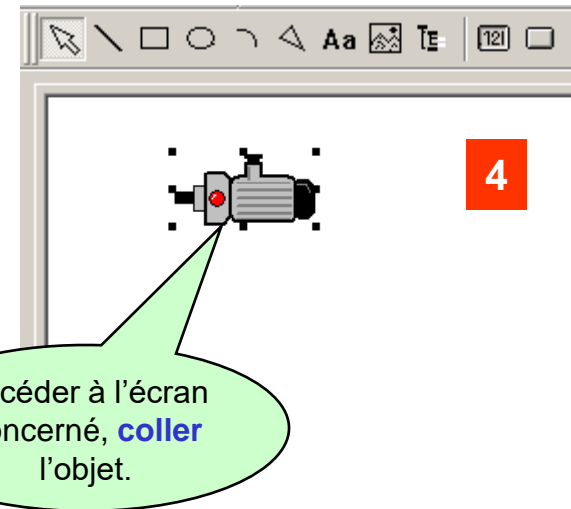
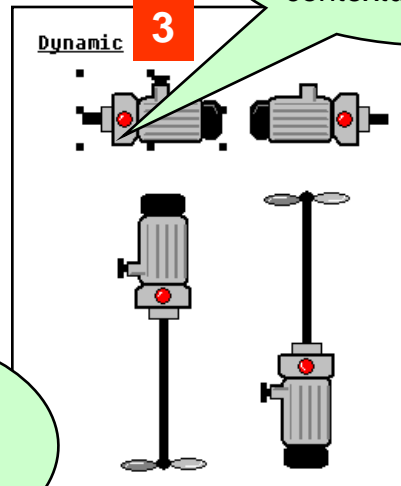
Sélectionner le menu
**Outils / Bibliothèques des
écrans d'exploitation.**

Outils
Bibliothèques d'écrans d'exploitation **1**



**2. Effectuer un
double clic** sur
le type d'objet à
insérer

Sélectionner l'objet à
insérer et le **copier** à
l'aide du menu
contextuel (**clic droit**)



Accéder à l'écran
concerné, **coller**
l'objet.

Programmation du palettiseur en langage ST

Les écrans d'exploitation (4/4)

Cahier des charges

Analyse

Déclaration des données

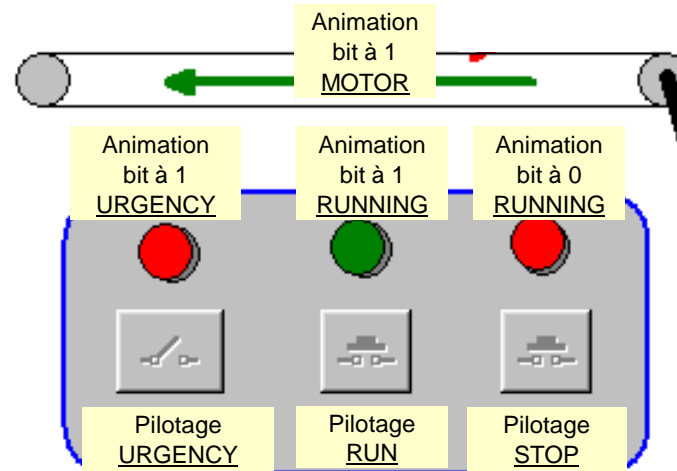
Programmation en ST

Mise au point

Les écrans d'exploitation

Configuration API

Vous allez créer le contenu de l'écran convoyeur, les textes en jaune indiquent les variables à associer aux objets (par l'onglet "animation" ou par l'onglet "pilotage" pour les propriétés de l'objet)



Pour passer en exploitation sur le simulateur et effectuer des commandes lorsque l'écran est affiché, appuyer sur la touche **F7** ou cliquer sur l'icône automate

Remarque : Après appui sur la touche F7, les boutons de commande sont actifs, et vous pouvez effectuer le pilotage du procédé

Utilisation de l'automate Modicon M340

Configuration de l'automate (1/2)

Cahier des charges

Analyse

Déclaration des données

Programmation en ST

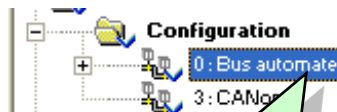
Mise au point

Les écrans d'exploitation

Configuration API

Utilisation de l'automate réel : création de la configuration physique.

Nous allons travailler avec un automate réel : se déconnecter du simulateur pour revenir en mode local et fermer éventuellement le simulateur (clic droit sur l'icône vert en bas de l'écran)



Effectuer un double clic sur **Bus automate** pour accéder à la configuration, le rack est affiché nous allons le configurer.

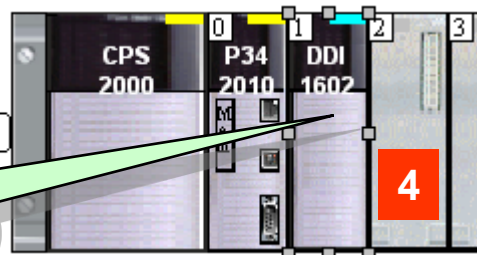


Effectuer un **double clic** sur l'emplacement 1 qui recevra le module d'entrées.

Référence	Description
Station d'E/S locale Modicon M340	
Analogique	
Communication	
Comptage	
TOR	
BMX DAI 1602	16 E num. négatives 24 VCA/24 VCC
BMX DAI 1603	Dig 16I 48 Vac
BMX DAI 1604	Dig 16I 100 to 120 Vac
BMX DAO 1605	16 S num. triac
BMX DDI 1602	Dig 16I 24 Vdc Sink
BMX DDI 1603	Dig 16I 48 Vdc Sink

3

Sélectionner dans les références TOR, le **module 16 Entrées DDM1602** et valider par OK, le module est **inséré** dans le rack.



Le **module** est **configuré**.

Utilisation de l'automate Modicon M340

Configuration de l'automate (1/2)

Cahier des charges

Analyse

Déclaration des données

Programmation en ST

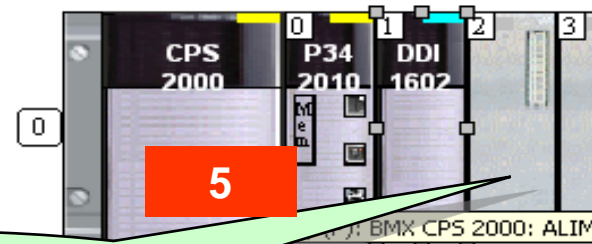
Mise au point

Les écrans d'exploitation

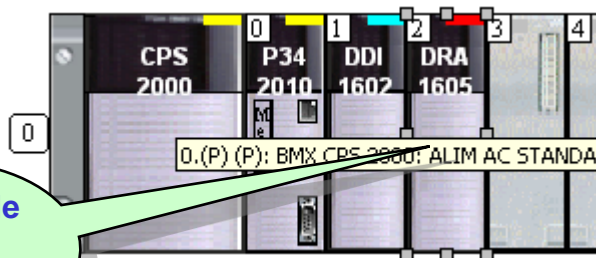
Configuration API

Utilisation de l'automate réel : création de la configuration physique.

Nous allons travailler avec un automate réel : se déconnecter du simulateur pour revenir en mode local et fermer éventuellement le simulateur (clic droit sur l'icône verte en bas de l'écran)



Effectuer un **double clic** sur l'emplacement 2 qui recevra le module de sorties.



Le **module** est configuré.

BMX DDO 1602	Dig 1602 Source 0,5A
BMX DDO 1612	16 S num. négatives statiques
BMX DDO 3202K	Dig 320 Trans Source 0.1A
BMX DDO 6402K	Dig 640 Trans Source 0.1A
BMX DRA 0805	Dig 80Q Isolated F
BMX DRA 1605	Dig 160Q Relays

Sélectionner dans les références TOR, le **module 16 Sorties** DRA1605 et valider par OK, le module est **inséré** dans le rack.

Utilisation de l'automate Modicon M340

Déclaration des données d'entrées/sorties (1/2)

Cahier des charges

Analyse

Déclaration des données

Programmation en ST

Mise au point

Les écrans d'exploitation

Configuration API

La configuration de l'automate étant définie, nous allons déclarer les adresses des entrées et des sorties.

Effectuer un double clic sur **Variables élémentaires** pour accéder à l'éditeur de données.

Sélectionner la variable **MOTOR**

Nom	Type	Adresse
MOTOR	EBOOL	
RUN	EBOOL	
RUNNING	EBOOL	
STOP	FRONT	

Nom	Type	Adresse
MOTOR	EBOOL	%Q0.2.1
RUN	EBOOL	
RUNNING	EBOOL	

Affecter une adresse réelle à la variable.

Nom	Type	Commentaire	Adresse
RUN	EBOOL	Départ convoyeur	%I0.1.1
STOP	EBOOL	Arrêt convoyeur	%I0.1.2
URGENCY	EBOOL	Arrêt d'urgence	%I0.1.3

Nom	Type	Commentaire	Adresse
MOTOR	EBOOL	Commande Moteur convoyeur	%Q0.2.1
RUNNING	EBOOL	Voyant Marche.Arrêt du moteur	%Q0.1.2

Remarques

Effectuer la même manipulation pour toutes les variables d'entrées/sorties :

Adressage : %I (entrée), %Q (Sortie) rack.emplacement.voie

Utilisation de l'automate Modicon M340

Déclaration des données d'entrées/sorties (2/2)

Cahier des charges

Analyse

Déclaration des données

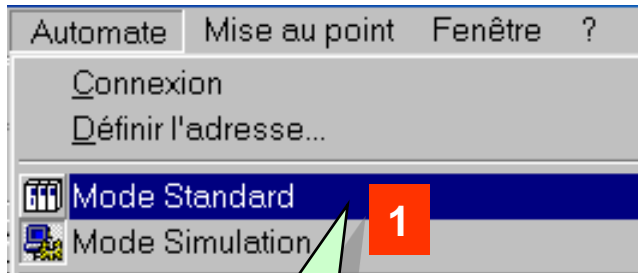
Programmation en ST

Mise au point

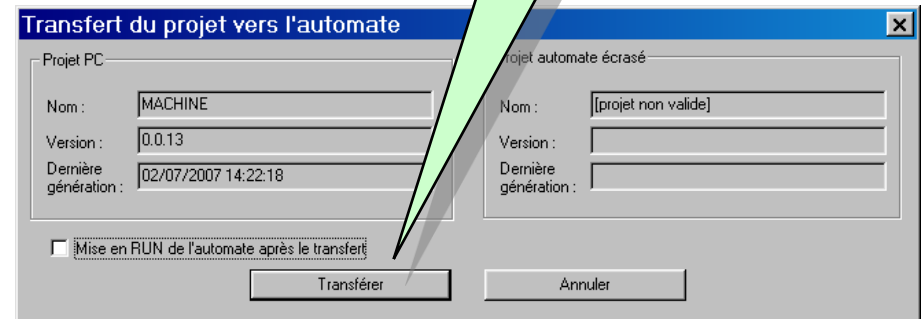
Les écrans d'exploitation

Configuration API

▶ A l'aide d'un automate Modicon M340, vous pouvez à présent tester votre programme. Effectuer l'analyse des modifications et génération du code comme dans les phases précédentes, puis transférer l'application dans l'automate et enfin passer en RUN.



Sélectionner la **cible d'exécution** du programme par le menu **Automate / Mode Standard**



Cliquer sur **Transférer.**

Remarque

Il est possible de revenir en mode simulation après avoir configuré un automate réel. Pour cela, procéder comme dans la phase 1 : Programmer en LD.

Sauvegarder le projet.
Fin de la phase 3 : Configuration API