

Unity Pro permet de programmer les automates **Modicon M340, Premium, Atrium, Quantum**. Actuellement PL7-Pro cohabite avec Unity sur la gamme des automates Premium, il est toutefois possible de faire migrer certains processeurs utilisant PL7-Pro vers des processeurs Unity par une mise à jour du système d'exploitation.

Attention peuvent subir la migration tous les processeurs TSXP572 sauf les TSXP572823 et tous les processeurs TSXP573, il faut en plus que le dernier chiffre des références se terminent par 3.

Unity Pro propose les langages suivants pour la création du programme utilisateur :

- Langage à blocs fonctions (FBD : Function Bloc Diagram)
- Langage Ladder (LD)
- Langage List (IL)
- Langage Littéral Structuré (ST)
- Diagramme fonctionnel en séquence (SFC : Sequential Fonctionnal Chart, grafcet)

Lancement du logiciel : le logiciel se lance par un double clic sur l'icône Unity sous le bureau ou depuis le menu démarrer de Windows → répertoire Schneider Electric, puis Unity Pro et clic sur le lien Unity.

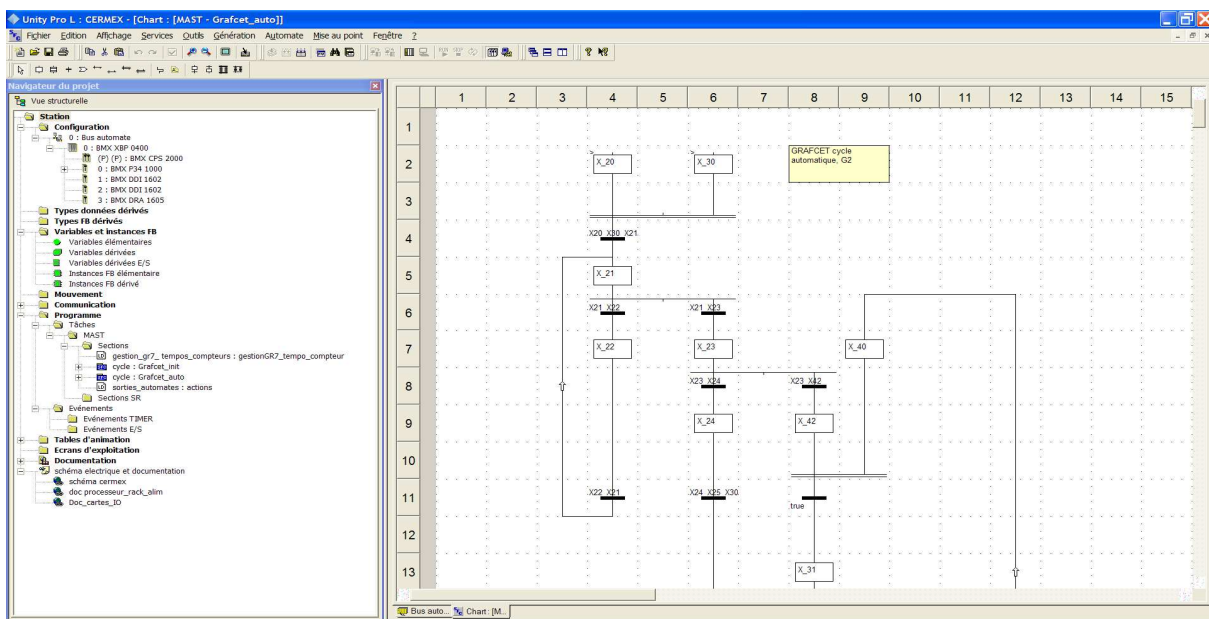
### Interface Utilisateur

The screenshot shows the Unity Pro software interface with the following callout boxes:

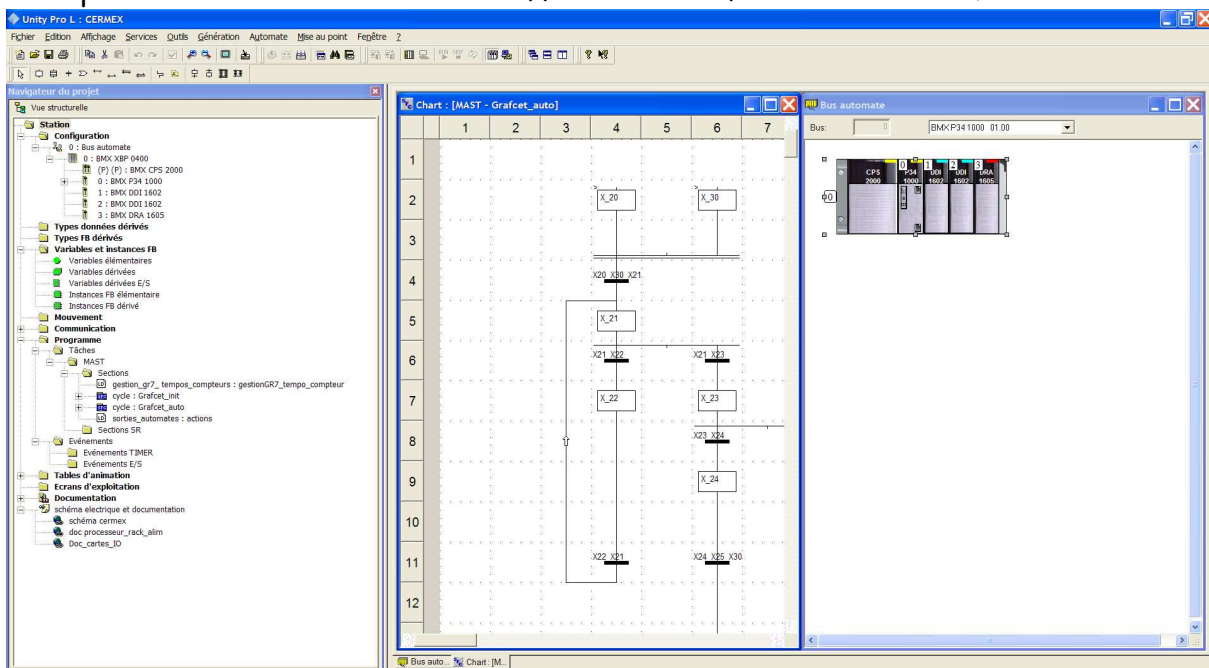
- Menus et Barres d'outils**: Points to the top menu bar and toolbar.
- Toutes les fonctionnalités sont accessibles depuis les menus ou les barres d'outils.**: A general note about accessibility.
- Navigateur de projet, donne accès à tous les éléments de l'application Unity**: Points to the project tree on the left.
- Fenêtre actuellement ouverte sous le format classeur identique à Excel.**: Points to the main workspace showing a ladder logic diagram.
- Barre d'informations, donne des informations sur les erreurs survenues, le suivi des signaux, les fonctions d'importation, etc.)**: Points to the bottom status bar.
- Barre de statut, donne des informations sur l'état du projet et sur le driver utilisé pour la connexion automate**: Points to the bottom status bar.

**Remarque** : avec Unity beaucoup de fonctionnalités sont également accessibles depuis **des menus contextuels** (accès par sélection avec la souris de l'élément en surbrillance, puis clic sur le bouton droit pour faire apparaître le menu contextuel lié à l'élément sélectionné). Si lors de l'utilisation du logiciel on se retrouve perdu dans la navigation entre les différentes fenêtres ouvertes, toujours revenir à la fenêtre du navigateur de projet. Pour avoir accès à cette fenêtre (si cette dernière est fermée) soit on utilise l'icône situé dans la barre d'outils supérieure (icône à gauche des jumelles) ou depuis la commande navigateur de projet du menu Outils.

Tous les éditeurs peuvent être basculés du mode normal au mode plein écran ou directement agrandi par utilisation de la souris.

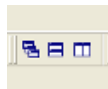


Sur cette représentation, on peut voir le navigateur d'application qui contient toutes les éléments de l'application Unity et deux éditeurs d'ouverts, on peut réorganiser l'espace de travail avec une vision différentes des fenêtres ouvertes.



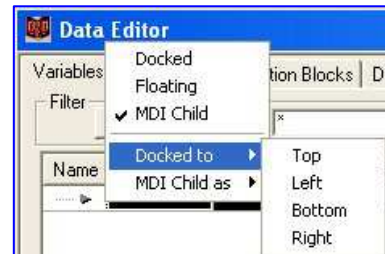
## Support à l'utilisation du logiciel Unity Pro

Pour réorganiser l'espace de travail, il suffit de sélectionner les icônes ci-dessous de la barre d'outils :



Certaines fenêtres comme l'éditeur de données et les tables d'animations peuvent être :

- DOCKED : positionne la fenêtre à l'extérieur de la fenêtre application
- FLOATING : la fenêtre sera toujours visible au Premier plan

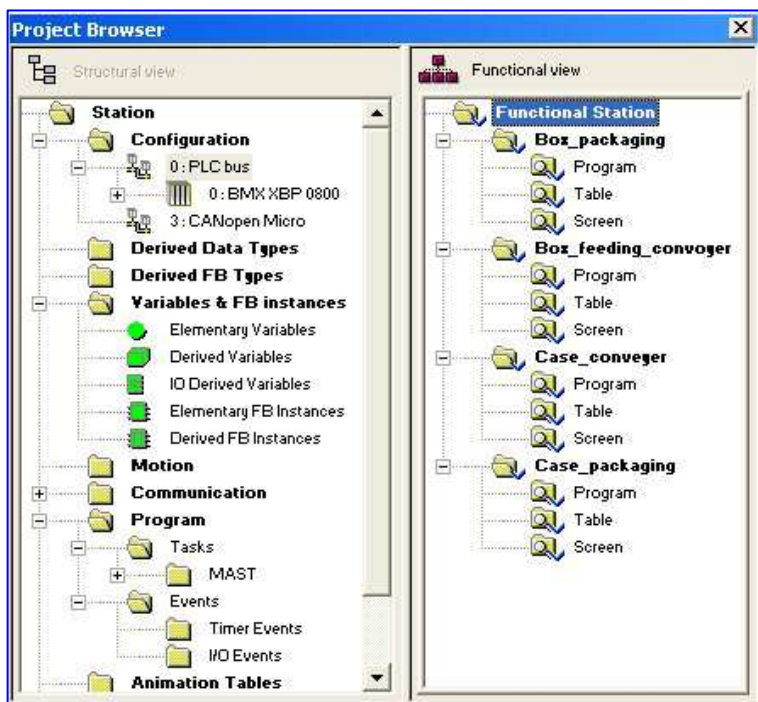


Pour avoir accès à ces attributs, il faut positionner la souris dans le bandeau bleu et effectuer un clic de souris gauche, le menu contextuel de l'illustration ci-dessus apparaît.

### Les différents éditeurs

#### Navigateur de projets

C'est l'éditeur qui permet de se rendre dans toutes les parties de l'application Unity



Le navigateur de projet propose 2 vues différentes pour présenter et structurer votre projet dans une arborescence de type arbre :

- Vue structurelle
- Vue fonctionnelle

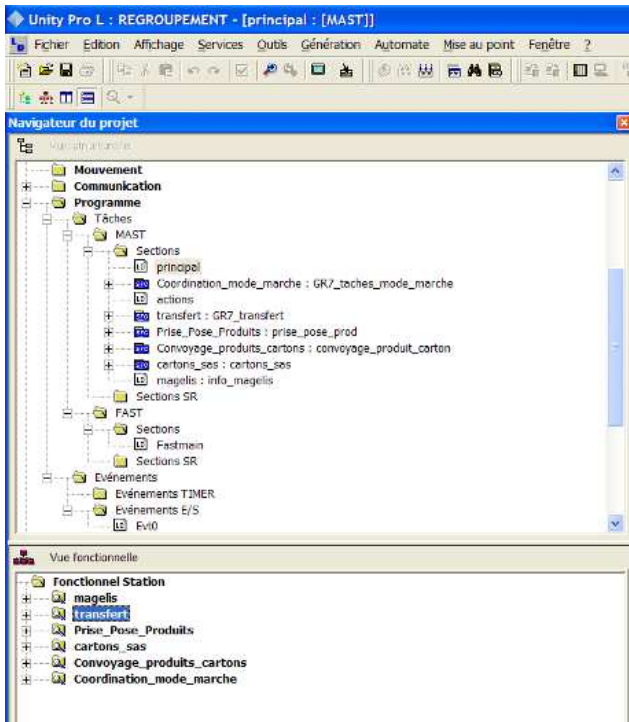
Dans la vue structurelle, l'utilisateur peut avoir accès et modifier les différents éléments de l'application (configuration, variable, programmes, documentations, tables d'animations...)

La vue fonctionnelle permet à l'utilisateur de structurer son application en modules

fonctionnels. Cette vue est très intéressante lorsque l'on arrive sur un système que l'on ne connaît pas pour voir la décomposition de l'application.

Dans tous les projets, la vue structurelle existe, par contre la vue fonctionnelle ne sera présente que si le concepteur l'a créée. On retrouvera toutes les parties de programme dans la vue structurelle et pas nécessairement dans la vue fonctionnelle en fonction du découpage effectuée par le développeur.

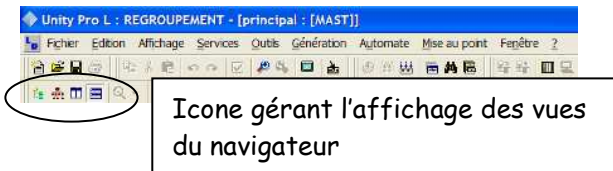
Exemple de décomposition



Ci-contre la vue structurelle d'une application qui contient toutes les parties de programme et la vue fonctionnelle où l'on retrouvera dans chaque module une ou plusieurs sections de programme de la vue structurelle.

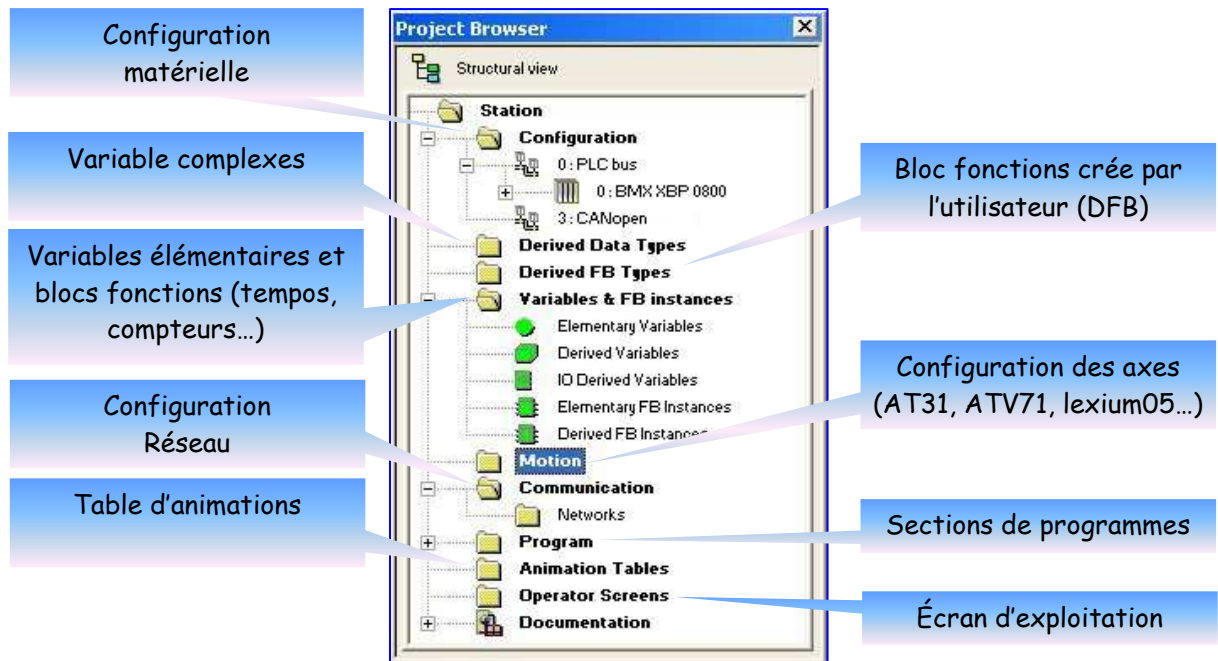
Attention dans la vue fonctionnelle, ne se trouve pas toutes les parties de programme, sur l'exemple ci-contre la section principale de la vue structurelle ne se retrouve dans aucun module de la vue fonctionnelle.

Il est fortement conseillé de travaillé avec les deux vues simultanément.



On peut faire afficher avec ces icônes soit une vue soit l'autre, voire les deux

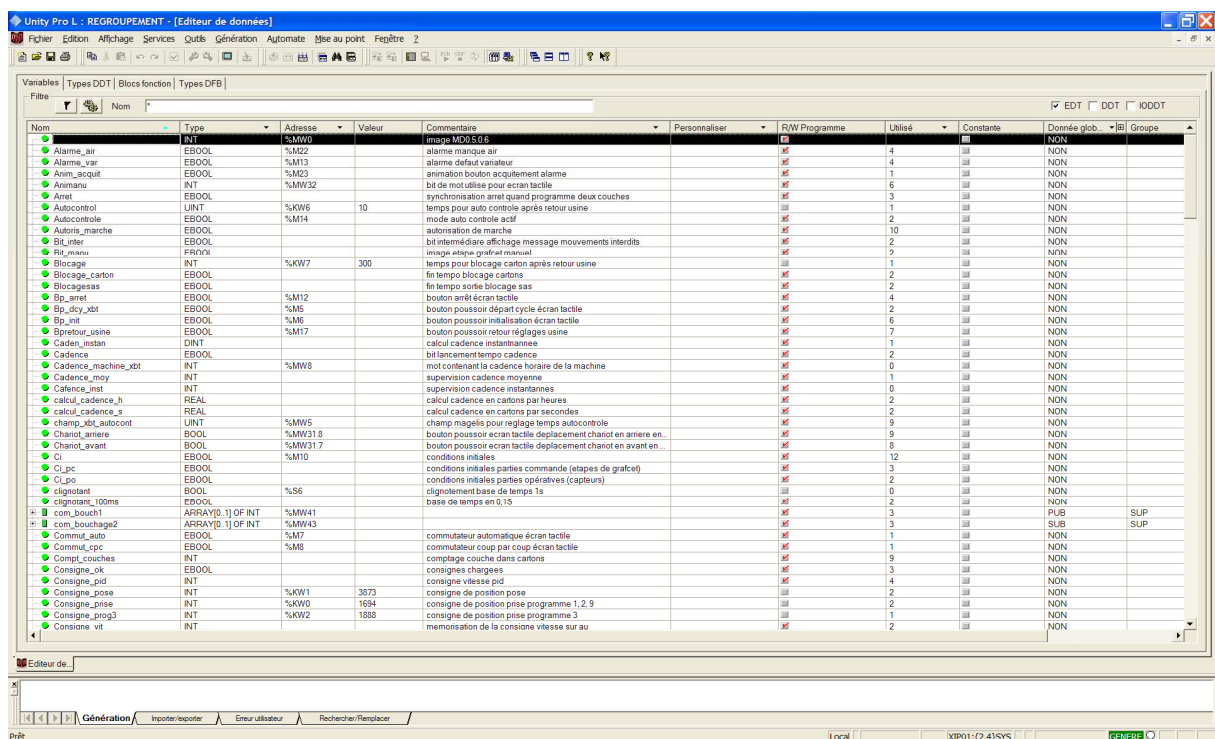
Détail de la vue structurelle :



### Editeur de données

L'accès à l'éditeur de donnée s'effectue le plus simplement par le navigateur de projet dans le répertoire Variables et instances FB puis en double cliquant sur l'un des cinq choix possibles : variables élémentaire, variables dérivées, variables dérivées d'entrées-sorties (IODT), Instance de FB élémentaire, Instance de bloc DFB (bloc créés par l'utilisateur). L'éditeur ci-dessous s'ouvre. L'éditeur de données contient toutes les variables de l'application Unity, nous verrons par la suite que l'on peut le renseigner à chaque fois que l'on crée une variable ou bien au fil de l'eau lors de l'utilisation d'une nouvelle variable dans le programme automate.

Dans cet éditeur on retrouve 4 onglets qui permettent d'aller déclarer les 5 types de variables décrit ci-dessus.

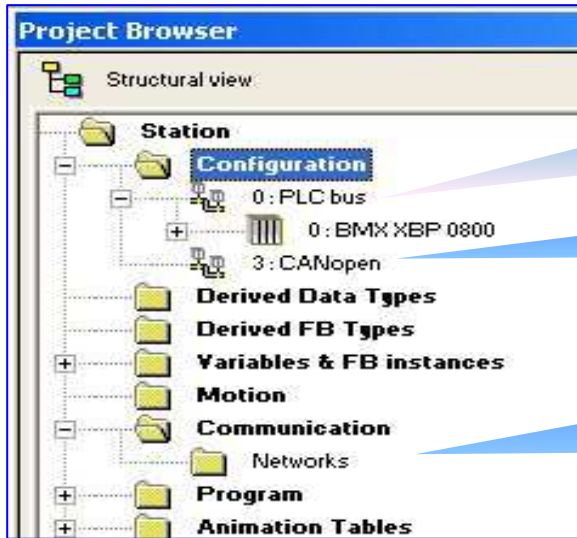


Il faut éviter d'utiliser une variable topologique (adressage automate par exemple %M2 ou %MWO), par contre on la déclare et on lui donne un nom (mnémonique).

### Editeur de configuration

Comme sous PL7, la configuration matérielle et logicielle s'effectue dans cet éditeur. La configuration matérielle permet de renseigner le logiciel sur le type d'automate programmé ainsi que les cartes utilisées, cette configuration est obligatoire.

Par contre en fonction des applications cette configuration peut être étoffée ce celle des réseaux, essentiellement CANOPEN et ETHERNET, si ces derniers sont présents.



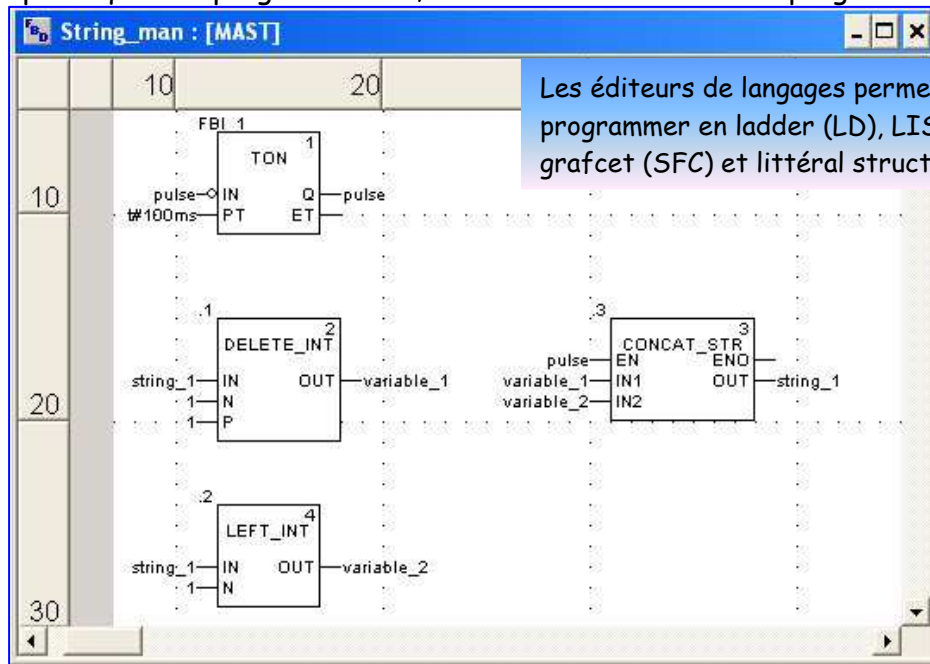
Configure le processeur et les cartes insérées dans le rack automate (obligatoire)

Configure le bus CANOPEN . Optionnel cela dépend si le bus est présent dans l'application.

Configuration physique et logiciel du réseau Ethernet et de ses services. Optionnel dépend si le réseau est présent dans l'application.

### Editeur de programmation

Les éditeurs de langage sont propres à chaque langage et contiennent des outils spécifiques de programmation, ci-dessous une section de programme en langage FBD.



Les éditeurs de langages permettent de programmer en ladder (LD), LIST (IL), FBD, grafcet (SFC) et littéral structure (ST).

### Editeur d'informations

Situé dans la partie basse de l'écran, cette fenêtre donne des informations précieuses sur les erreurs de programmation lors de l'analyse, ou lors de l'import export, ou bien lors de remplacement de variables. Il est recommandé de vérifier fréquemment son contenu.



**Architecture mémoire et règles de recouvrement des données à adressage direct**

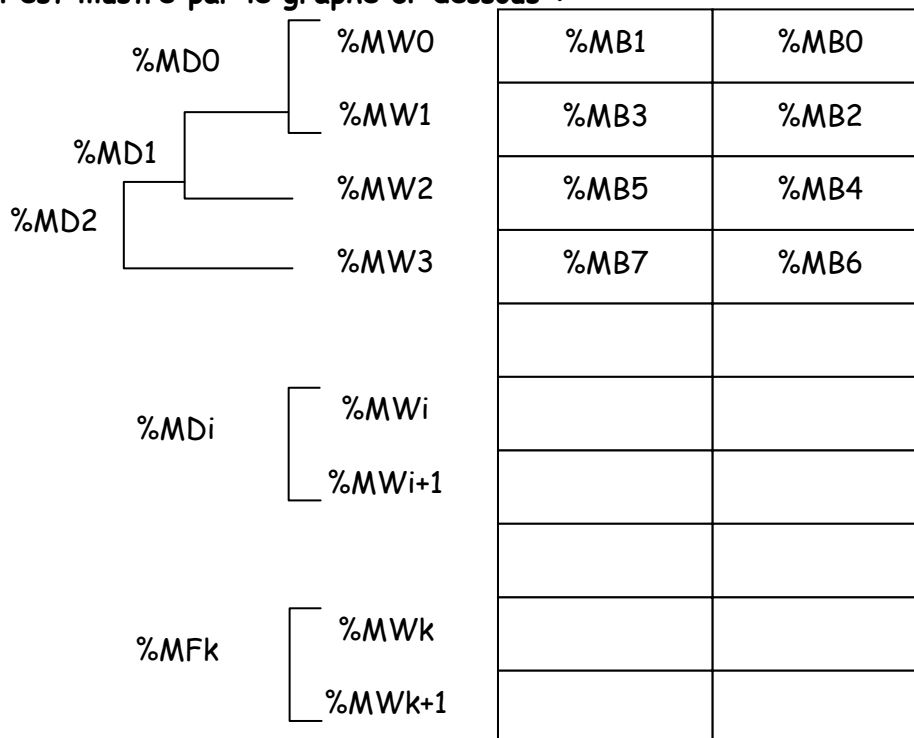
Les instances de données à adressage direct possèdent un emplacement prédéfini dans la mémoire automate ou dans un module métier et cet emplacement est connu de l'utilisateur.

Les octets (8 bits), mots simples (16 bits), double longueur (32 bits) et flottant (32 bits, permet le stockage de chiffre à virgule) sont rangés à l'intérieur de l'espace donné dans une même zone mémoire.

Ainsi, il y a recouvrement (chevauchement des zones) entre :

- le mot double longueur **%MDi** et les mots simple longueur **%MWi** et **%MWi+1** (le mot **%MWi** renfermant les poids faibles et le mot **%MWi+1** les poids forts du mot **%MDi**)
- le mot simple longueur **%MWi** et les octets **%MBj** et **%MBj+1** (avec  $j=2 \times i$ )
- le flottant **%MFk** et les mots simple longueur **%MWk** et **%MWk+1**.

C'est ce qui est illustré par le graphe ci-dessous :



Prenons un exemple, on décide dans le programme de venir stocker dans **%MW0** la valeur 3238, dans **%MB0** la valeur 243 :

Le codage de 3228 en binaire sur 16 bits donne :

%MB1							%MW0								%MB0	
32768	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1	
$2^{15}$	$2^{14}$	$2^{13}$	$2^{12}$	$2^{11}$	$2^{10}$	$2^9$	$2^8$	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$	
0	0	0	0	1	1	0	0	1	0	0	1	1	1	0	0	

**Exemple :**

On s'aperçoit qu'en stockant la valeur 3238 dans le mot simple %MWO, l'octet %MBO contient la valeur 156 (10011100 en binaire). Si à la suite de l'instruction de stockage de la valeur 3238 on stocke 243 (11110011) dans %MBO, c'est alors la valeur de %MWO qui change : 00001100 11110011 en binaire soit : 3315.

**Remarque :** le nombre de mot simple, doubles, et bits à adressage direct se configurent dans l'éditeur de configuration matériel du processeur et ensuite dans l'éditeur de données on leur donnera le format de données que l'on souhaite (cf. p 25).

La syntaxe d'une instance de données à adressage direct **est définie par le symbole % suivi d'un préfixe de localisation mémoire et dans certains cas d'informations supplémentaires.**

Le préfixe de localisation mémoire peut être :

- **M** pour les variables internes
- **K** pour les constantes
- **S** pour les variables systèmes
- **N** pour les variables systèmes
- **I** pour les variables d'entrées
- **Q** pour les variables de sorties

<b>Variables internes %M</b>
------------------------------

	Syntaxe	Format	Exemple	Droit d'accès du programme
Bit	%Mi	1 bits	%M0	R/W
Mot	%MWi	16 bits	%MWO	R/W
Bit extrait de mot	%MWi.j	1 bits	%MWO.1 (bit 1 du mot 0)	R/W
Double mot	%MDi	32 bits	%MD8	R/W
Réel (flottant)	%MFi	32 bits	%MF15	R/W

**Légende :**

R : variable accessible en lecture

W : variable accessible en écriture

**Remarque :** dans l'automate M340, l'adressage direct des zones mémoires de type double (mot double) ou flottant (réel) n'existe pas. **Pour les déclarer il faut les placer dans une zone de type mot (%MW). L'index i %MW doit être pair.**



## Constantes %K

	Syntaxe	Format	Exemple	Droit d'accès du programme
Mot	%KW <sub>i</sub>	16 bits	%KW0	R
Double mot	%KD <sub>i</sub>	32 bits	%KD8	R
Réel (flottant)	%KF <sub>i</sub>	32 bits	%KF15	R

Les variables %KD<sub>i</sub> et %KF<sub>i</sub> ne sont pas disponibles sur l'automate M340. Pour les déclarer il faut les placer dans une zone de type mot (%KW). L'index <sub>i</sub> %KW doit être pair. Les constantes sont des variables que l'on ne peut que lire, on fixera la valeur de ces variables dans l'éditeur de données.

## Variables systèmes %S

	Syntaxe	Format	Exemple	Droit d'accès du programme
Bit	%S <sub>i</sub>	1 bit	%S6	R/W ou R
Mot	%SW <sub>i</sub>	16 bits	%SW0	R/W ou R

Les variables systèmes sont des fonctions prédéterminées embarquées par le constructeur en mémoire automate. Pour plus de détails, consulter l'aide du logiciel.

## Adressage des variables d'entrées sorties

L'accès aux entrées-sorties des modules métiers, dépend de plusieurs éléments décrits dans le tableau ci-dessous.

	Syntaxe	Exemple	Droit d'accès du programme
<b>Entrées</b>			
Bit de voie	%I\b.e\r.m.c.d	%I4.2.3.0	R
Bit d'erreur module	%I\b.e\r.m.MOD.ERR	%I4.2.MOD.ERR	R
Bit d'erreur de voie	%I\b.e\r.m.c.ERR	%I4.2.3.ERR	R
Mot de type INT	%IW\b.e\r.m.c.d	%IW4.2.3.1	R
Double mot de type DINT	%ID\b.e\r.m.c.d	%ID4.2.3.0	R
Reel de type REAL	%IF\b.e\r.m.c.d	%IF4.2.3.0	R

*Support à l'utilisation du logiciel Unity Pro*

	Syntaxe	Exemple	Droit d'accès du programme
<b>Sorties</b>			
Bit de voie	%Q\b.e\r.m.c.d	%Q4.2.3.0	R/W
Bit d'erreur module	%Q\b.e\r.m.MOD.ERR	%Q4.2.MOD.ERR	R/W
Bit d'erreur de voie	%Q\b.e\r.m.c.ERR	%Q4.2.3.ERR	R/W
Mot de type INT	%QW\b.e\r.m.c.d	%QW4.2.3.1	R/W
Double mot de type DINT	%QD\b.e\r.m.c.d	%QD4.2.3.0	R/W
Réel de type REAL	%QF\b.e\r.m.c.d	%QF4.2.3.0	R/W

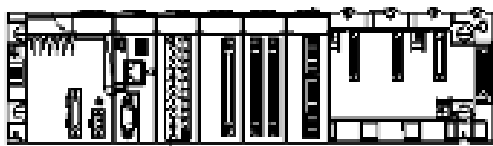
**Entrées-Sorties avec utilisation des IODDT**

Structure d'IO	%CH\b.e\r.m.c	%CH4.2.3	R
<b>Mots cartes métier (T.O.R, analogique, réseau...)</b>			
Mot de type INT	%MW\b.e\r.m.c.d	%MW4.2.3.1	R/W
Double mot de type DINT	%MD\b.e\r.m.c.d	%MD4.2.3.0	R/W
Réel de type REAL	%MF\b.e\r.m.c.d	%MF4.2.3.0	R/W

- b** : numéro de bus (omis si station locale)
- e** : numéro du point de connexion de l'équipement (omis si station locale)
- r** : numéro du rack
- m** : emplacement du module sur le rack
- c** : numéro de voie
- d** : numéro de données de 0 à 999, facultatif si valeur à 0

**Station locale  
(sans bus de communication)**

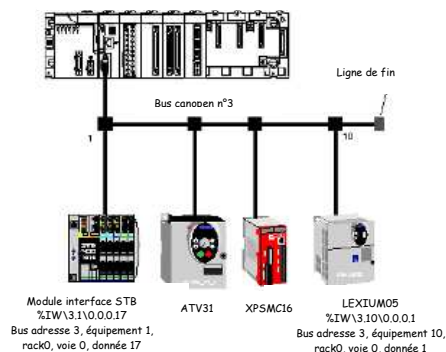
Configuration M340



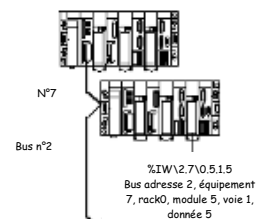
%IO1.5  
Rack0, module 1, voie 5

**Station sur bus**

Configuration M340



Configuration Premium



Dans le cas de l'application en mode locale, les termes b et e peuvent être omis on retrouve donc uniquement la syntaxe %I.r.m.c soit dans notre cas %IO.1.5. Dans le cas d'adressage direct de la voie d'une entrée ou d'une sortie le terme d (n° de données est le plus souvent égale à 0).

On pourra trouver très facilement un numéro de données différent de 0 dans les mots métiers comme par exemple : %MW4.0.1.3.4 (dans ce cas la donnée n°4 correspond au bit n°4 du mot %MW4 de la voie 3 du module 1 monté sur le rack 0)

**Remarque** : les numéros de modules, de rack, de voie et de bus peuvent être visualisés sous Unity dans l'éditeur de configuration matérielle

### Types de variables utilisées dans Unity

Le logiciel Unity travaille avec deux types de données différentes :

- EDT (Elementary Data Type : données de types élémentaires)
- DDT (Derivated Data Type : données complexes dérivées)

#### Type de données EDT utilisé par Unity

**BOOL** : qui contient uniquement la valeur 0 ou 1 sans détection de fronts

**EBOOL** : qui contient la valeur 0 ou 1 avec détection des fronts

**BYTE** : chaîne de 8 bits

**INT** : nombre entier de 16 bits signés compris entre **-32768 et +32767**

**UINT** : nombre entier de 16 bits non signés valeur comprise entre **0 et 65535**

**DINT** : nombre entier de 32 bits non signés, valeur comprise entre **-2147483648 et +2147483647**

**UDINT** : nombre entier de 32 bits signés, valeur comprise entre **0 et 4294967295**

**REAL** : nombre réel (chiffre à virgule), codé sur 32 bits, valeur comprise entre **-3.402824e+38 et +3.402824e+38**

**WORD ou DWORD** : chaîne de 16 ou de 32 bits. La particularité de ce format est que l'ensemble des bits qui le compose ne représente pas une valeur numérique, mais une combinaison de bits séparés. Très utilisé pour stocker des grandeurs de type BCD. Les données appartenant aux types de ce format peuvent être représentées sous trois

*Support à l'utilisation du logiciel Unity Pro*

bases qui sont : l'hexadécimal (16#valeur en hexa), l'octal (8# valeur en octal) et le binaire (2#valeur en binaire).

**TIME** : codé sur 32 bits contient l'heure codés en millisecondes, soit une durée maximale d'environ 49 jours. La syntaxe est T# ou TIME# suivi de la valeur pouvant être exprimée en plusieurs unités

- T# 4294967295MS                    valeur en millisecondes
- T#4294967S\_295MS                valeur en secondes et millisecondes
- T# 71582M\_47S\_295MS            valeur en minutes, secondes, ms
- T# 1193H\_2M\_47S\_295MS        valeur en heures, minutes, s, ms
- T# 49J\_17H\_2M\_47S\_295MS     valeur en jour, heures, minutes, s, ms

**T.O.D** : Time of day codé sur 32 (syntaxe : **TOD#1:10:3**, 1 heure, 10 minutes, 3 secondes). Chaque grandeur est codée au format BCD, les 8 bits de poids faibles sont inutilisées.

Heure (13)	Minutes (25)	Secondes (47)	Octet de poids faible
0001 0011	0010 0101	0100 0111	Inutilisés

**DATE** : format de 32 bits contenant la date, syntaxe D#1994-3-15 (15 mars 2005). L'année, le mois et le jour sont respectivement codés en BCD, les 8 bits de poids faibles sont inutilisées.

Année (2001)	Mois (09)	Jour (20)
0010 0000 0000 0001	0000 1001	0010 0000

**DT** : date and Time codé sur 64 bits, syntaxe : DT#2000-1-10-0:40:0 (10 janvier 2000, 0h40mn0s). Chaque grandeur est codée en BCD, les 8 bits de poids faibles sont inutilisées.

Année (2000)	Mois (09)	Jour (20)	Heure (13)	Minute(25)	Secondes (47)	Octet de poids faible
0010 0000 0000 0000	0000 1001	0010 0000	0001 0011	0010 0101	0100 0111	Inutilisés

**STRING** : le format chaîne de caractères permet de représenter une chaîne de caractères ASCII (lettre, chiffre, virgule...), chaque caractère étant codé sur un format de 8 bits. La taille maximale d'une chaîne est de 65 534 caractères. La taille par défaut d'une variable ayant le format **STRING** est de 16 caractères. Par exemple soit la variable toto à qui on définit un format STRING, lors de la saisie : **toto := 'il fait beau'**

Cette variable peut contenir 16 caractères, dans notre cas elle en contient 12 (l'espace compte comme un caractère).

On peut également définir le nombre de caractère que contiendra la variable STRING. Lors de l'affectation de l'attribut il suffit de saisir STRING[12]. Dans ce cas la variable à laquelle on aura affectée cet attribut pourra contenir au maximum 12 caractères. Le nombre de caractères devra être inférieur 65534.

### Type de données DDT (Derivated Data Type) utilisé par Unity

Les données de types dérivées (DDT) comprennent des données complexes de deux natures :

- Tableaux
- Structures

**Tableaux** : Un tableau est un élément de données incluant un ensemble de données de type identique.

- De données élémentaires (EDT) par exemple :
  - Un groupe de mots INT
  - Un groupe de mots REAL

L'avantage du tableau réside dans le fait d'éviter de déclarer x variable élémentaire. Prenons un exemple si sur une machine nous avons besoin de 10 mots de type REAL pour traiter une partie de programme, **au lieu de déclarer 10 variable EDT de type REAL, on créera une donnée de type DDT à laquelle on attribuera l'attribut de tableau, ainsi on crée une variable et non 10.**

**Syntaxe : ARRAY[X..Y] OF TYPE**

X : numéro de départ de la donnée du tableau

Y : numéro de fin de la donnée du tableau

TYPE : format des données contenu dans le tableau (BOOL, EBOOL, INT.....)

Exemple : ARRAY[0..9] OF REAL

Tableau contenant 10 mots (0 à 9) de type REAL (données de types EDT, 32 bits)

**Structures** : c'est une donnée qui contient un ensemble de données de type différent telles que :

- Un ensemble de BOOL, WORD, UINT, etc (structure EDT)
- Un ensemble de tableaux (structures DDT)
- un ensemble de REAL, DWORD, tableaux (structure EDT et DDT)

Les structures sont très utilisées dans les programmes d'automatismes pour créer des recettes. En effet les machines de production peuvent en général fabriquer plusieurs

produits avec des conditions de fabrication bien différentes. Chacune des recettes contiendra toutes les variables permettant la gestion respective d'un produit, encore une fois cela structure le programme, supposons que l'on fabrique deux produits et que l'on est besoin dans le programme de 10 variables pour gérer la fabrication de chacun de ses produits, au lieu de déclarer 20 variables de types EDT, on déclare 2 variables auxquelles on attribue le type structures, on sein des ses structures on gèrera le type de variables que l'on a besoin.

**Remarque** : toutes les variables EDT et DDT seront déclarés dans l'éditeur de données

### **Cas particulier IODDT (données dérivées d'entrées sorties)**

Le terme IODDT désigne un type de données structuré représentant un module ou une voie d'un module automate. Chaque module expert présente ses propres IODDT.

**Les types de données dérivés d'entrées\sorties IODDT (Input Output Derivated Data Type) sont prédéfinis par le constructeur, ils contiennent des objets langage de la famille EDT appartenant à la voie d'un module métier.**

Les types IODDT sont des structures dont la taille (nombre d'éléments qui les composent) dépend de la voie ou du module d'entrées\sorties qu'elles représentent.

Les IODDT constituent une des grosses différences entre la programmation PL7 et la programmation Unity. En effet sous Unity on peut adresser directement une entrée ou une sortie avec son adresse topologique (cf. adressage p 9 et 10), on peut voir que cet adressage peut être vite relativement fastidieux, grâce à l'utilisation des IODDT, le logiciel va directement nous faire correspondre l'adresse topologique avec le nom que l'on a donnée à l'IODDT, avec en plus l'avantage de remonter tout un tas d'information supplémentaire (cf. p 27). **Sous Unity il faut éviter de travailler en adressage direct (topologique).**

### **Cas particulier des données appartenant à un grafcet**

La famille des types de données diagrammes fonctionnels en séquence SFC (Sequential Function Chart) regroupe des types de données dits composés **tels que des structures restituant les propriétés et l'état du graphe (Chart) et des actions le composant.**

Chaque étape est représentée par deux structures qui sont :

- la structure **SFCSTEP\_STATE**
- la structure **SFCSTEP\_TIMES**

### **Définition de la structure de type SFCSTEP STATE :**

- **x**: donnée élémentaire (EDT) de type BOOL contenant la valeur TRUE quand l'étape est active

*Support à l'utilisation du logiciel Unity Pro*

- **t**: donnée élémentaire (EDT) de type TIME contenant le temps d'activité de l'étape (valeur en ms). Lorsqu'il est désactivé, la valeur de l'étape est maintenue jusqu'à la prochaine activation.
- **tminErr**: donnée élémentaire (EDT) de type BOOL contenant la valeur TRUE si le temps d'activité de l'étape est inférieur au temps d'activité minimal programmé.
- **tmaxErr**: donnée élémentaire (EDT) de type BOOL contenant la valeur TRUE si le temps d'activité de l'étape est supérieur au temps d'activité maximal programmé.

Ces données sont accessibles à partir de l'application en lecture seule.

**Définition de la structure de type SFCSTEP TIMES**

Cette structure rassemble les données liées aux paramétrages du temps d'exécution de l'étape ou de la Macro-étape.

Ces données sont:

- **delay**: donnée élémentaire (EDT) de type TIME définissant le temps de retard de scrutation de la transition situé en aval de l'étape active.
- **tmin**: donnée élémentaire (EDT) de type TIME contenant la valeur minimale durant laquelle l'étape doit être exécutée, Si cette valeur n'est pas respectée, **les données tminErr prennent la valeur TRUE.**
- **tmax**: donnée élémentaire (EDT) de type TIM contenant la valeur maximale durant laquelle l'étape doit être exécutée. Si cette valeur n'est pas respectée, **les données tmaxErr prennent la valeur TRUE.**

Ces données sont accessibles uniquement à partir de l'éditeur du SFC et pas depuis le programme.

**Syntaxe d'accès à des données de la structure SFCSTEP\_STATE**

Syntaxe	Commentaire
Nom_Etape.x	Permet de connaitre l'état de l'étape (active\inactive)
Nom_Etape.t	Permet de connaitre le temps d'activation en cours ou total de l'étape
Nom_Etape. tminErr	Permet de connaitre si le temps minimal d'activation de l'étape est inférieur au temps programmé dans Nom-Etape.tmin

*Support à l'utilisation du logiciel Unity Pro*

Nom_Etape.tmaxErr	Permet de connaître si le temps maximal d'activation de l'étape est supérieur au temps programmé dans Nom_Etape.tmax
-------------------	--

**Types de blocs fonctions utilisés dans Unity**

Les familles de type de données blocs fonction sont :

- la famille de type Blocs fonction élémentaire (EFB : Elementary Fonction Block)
- la famille de type Blocs fonction utilisateur (DFB : Derivated Fonction Block)

Les types blocs fonctions élémentaires (EFB) sont fournis par le constructeur, et sont stockés dans les bibliothèques de Unity, **c'est dans ces blocs que l'on trouvera les temporisations, compteurs et autres blocs fonctions prédéfinis que l'on se servait dans PL7-Pro**

Ils sont programmés en langage C. L'utilisateur peut créer ses EFB, pour cela il doit disposer d'un outil logiciel optionnel "SDKC".

Un type EFB peut avoir une ou plusieurs instances, chaque instance est référencée par un nom (symbole) et possède les données du type de l'EFB. **C'est à dire que l'on peut utiliser plusieurs fois le même bloc temporisation ou compteur, il suffit simplement de lui donner un nom différent (instancier).**

Les types blocs fonction utilisateur (blocs fonction dérivés) sont développés par l'utilisateur avec un ou plusieurs langages (suivant le nombre de sections). Ces langages sont les suivants :

- ladder
- FBD
- List Instruction
- Litteral Structure

L'utilisateur peut ensuite enregistrer ces blocs dans la bibliothèque pour les utiliser dans d'autres applications. Un type DFB peut avoir une ou plusieurs instances, chaque instance est référencée par un nom, c'est-à-dire que l'on peut utiliser plusieurs fois le même bloc, il suffit de lui donner un nom différent.

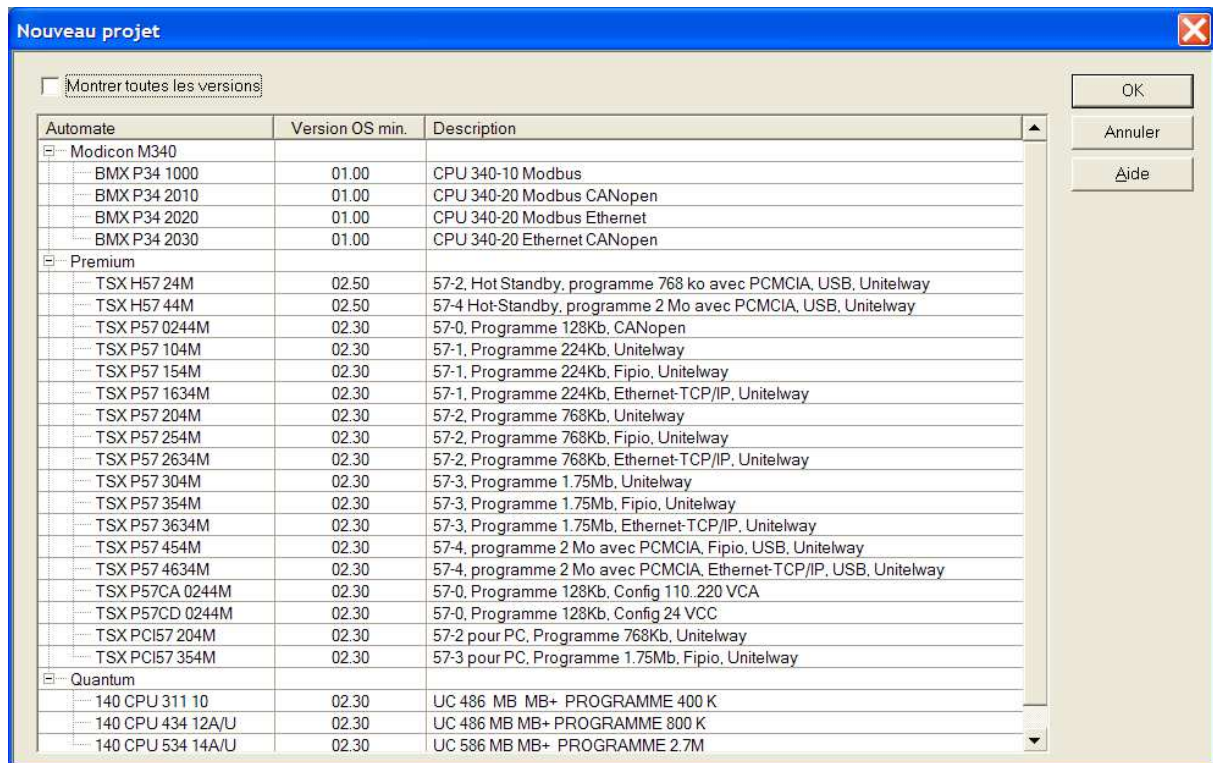


**METHODOLOGIE POUR LE DEVELOPPEMENT D'UNE NOUVELLE  
APPLICATION**

- **Définir les propriétés :**
  - de la station de travail accessible via le Menu OUTILS/OPTIONS
  - du projet via le Menu OUTILS/OPTIONS DU PROJET
- **Configuration matériel (hardware, rack, module) et des réseaux le cas échéant avec l'éditeur de configuration**
- **Définir et éditer les variables** (variables élémentaires, arrays, structures, instance de bloc fonctions) avec l'éditeur de variables
- **Création de la structure de l'application :**
  - Taches (maitre, rapides, événementielle) avec navigateur de projet
  - sections de programme avec navigateur de projet
  - modules fonctionnels avec navigateur de projet
- **Ecrire les programmes dans les sections (LADDER, SFC, FBD....)**
- **Générer le code accessible par le menu GENENER, GENERER LE PROJET**
- **Choisir la cible automate ou simulateur et transférer l'application**
- **Mise au point de l'application avec les outils de communication en ligne avec l'automate**

## Création d'une nouvelle application

Sélectionner la commande fichier nouveau, la fenêtre ci-dessous apparaît, choisir alors son type de processeur et la référence installée ainsi que le choix de la version du système d'exploitation du processeur (ce système peut être mis à jour avec l'outil OSLOADER, logiciel fournit avec Unity) puis valider par OK. Si la version de votre processeur n'apparaît pas cocher la case montrer toutes les versions.



En fonction de la version du logiciel Unity S, M, L ou XL le type et le nombre d'automate pouvant être programmé pourra être plus ou moins important.

**Unity S** : uniquement les automates M340

**Unity M** : M340 et tous les automates Premium jusqu'au TSXP571634

**Unity L** : M340, tous les Premium et une partie des automates Quantum

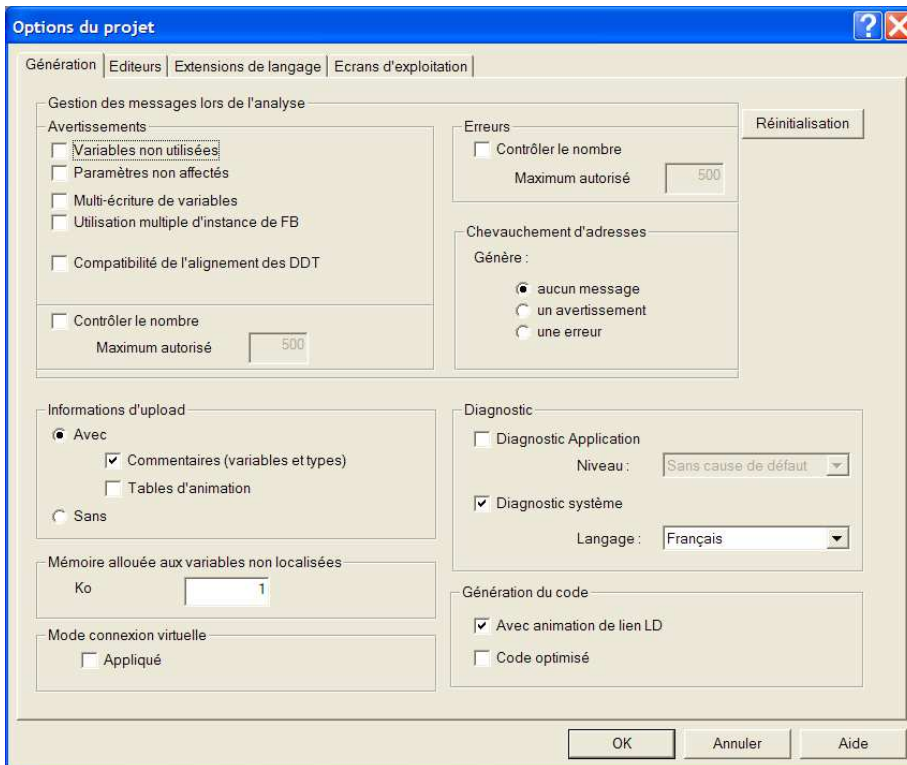
**Unity XL** : M340, Premium, Quantum, Atrium

**ATTENTION QUAND LE TYPE D'AUTOMATE A ETE SELECTIONNE ET VALIDER, ON NE PEUT PLUS REVENIR EN ARRIERE**

Sélectionner ensuite la commande **fichier** → **enregistrer sous** pour le stockage de votre programme sur le PC à l'emplacement de votre choix.

## Définition des propriétés du projet

Après la création de l'application, on doit sélectionner les propriétés de la station de travail et les propriétés du projet par les commandes respectives du menu **OUTIL/OPTIONS** et **OPTIONS DU PROJET**.



On peut faire remonter tout un tas d'avertissement lors de l'analyse du projet, ce choix est à l'initiative du développeur. Il peut être intéressant de faire cocher la case multi-écritures de variables et le chevauchement d'adresse génère un avertissement. En effet dans le premier cas un message nous informera lors de l'analyse si une variable est pilotée plusieurs fois dans le

projet (intéressant pour éviter la multi-écriture de la même sortie dans un programme). Pour le chevauchement d'adresse, cela permet d'éviter des erreurs de recouvrement, par exemple si dans le programme on utilise %MWO et %MDO, le logiciel nous avertira que ces deux zones mémoires se superposent.

La partie la plus importante de cet éditeur correspond aux informations d'UPLOAD. Les informations d'upload sont constituées de :

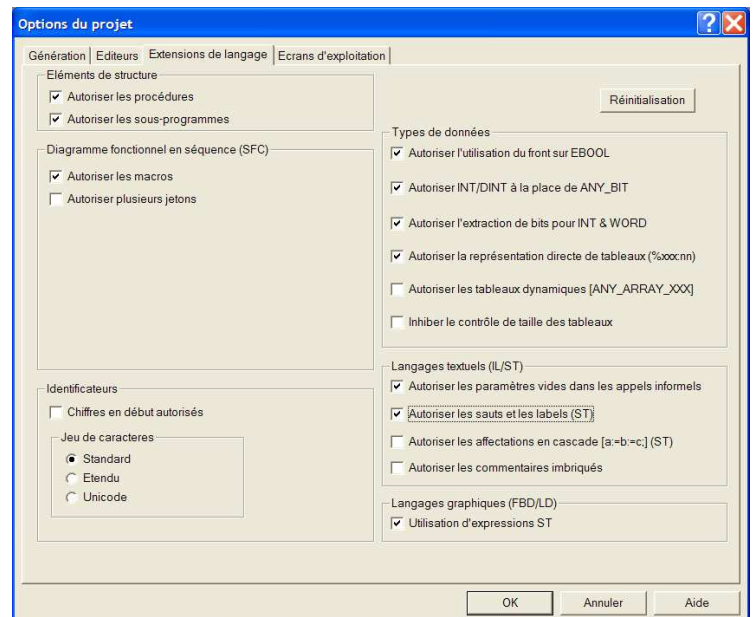
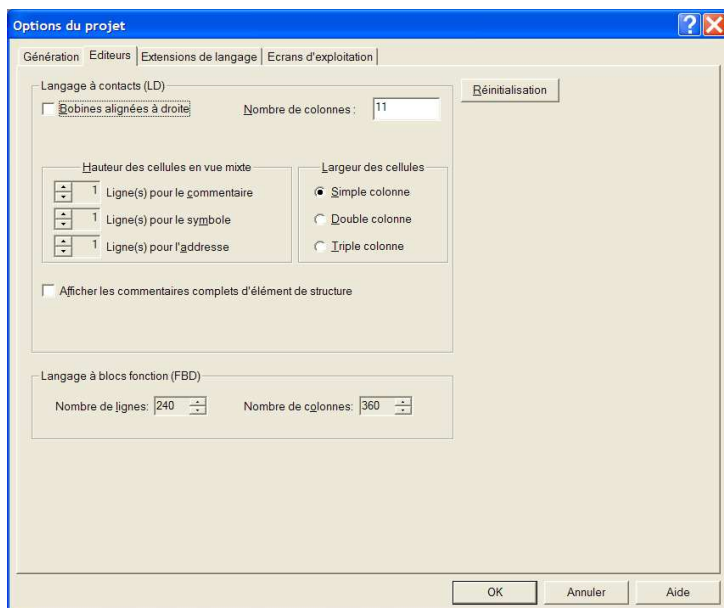
- informations pour déchargement de projet : code graphique des langages à contacts LD et diagramme de blocs fonction FBD, symboles des variables localisées et non localisées
- commentaires : des variables, section de code
- tables d'animation

Les informations d'Upload ne doivent être chargées dans l'automate que dans les cas indispensables afin d'optimiser la taille mémoire disponible et de la réserver pour le code exécutable, ainsi que d'améliorer les performances des modifications en connecté.

En phase maintenance et exploitation, ces informations devront être incluses dans l'automate s'il y a nécessité de restituer l'application sur des terminaux vierges (n'ayant pas à disposition le fichier d'origine du projet). Par contre outre le fait du gain de place mémoire, l'absence des informations d'Upload constitue une protection contre la lecture et l'écriture

En conclusion il est fortement recommandé d'activer cette option pour permettre un accès complet au programme de l'automate. Si cette option est décochée et que l'on ne dispose pas du programme d'origine, on peut seulement passer l'automate en STOP ou en RUN. Si la taille mémoire de l'automate le permet, il est intéressant d'embarquer les commentaires,

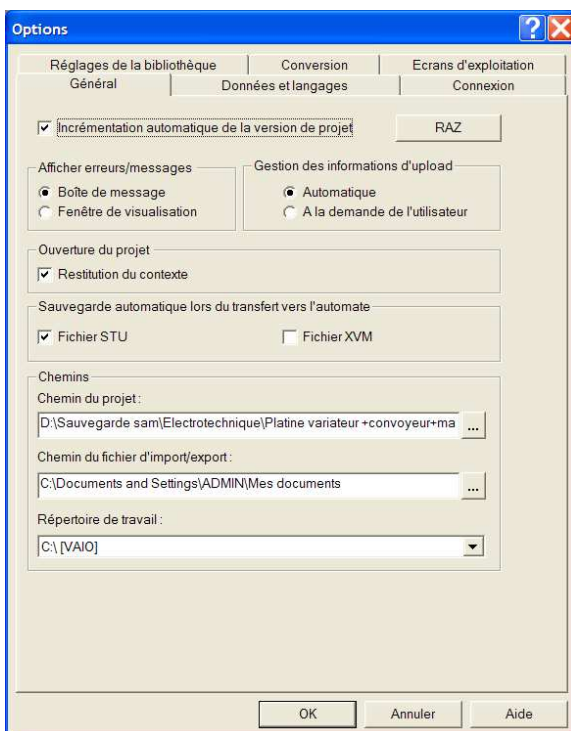
Il est également conseillé d'activer l'onglet diagnostic application et diagnostic système ainsi que de sélectionner la langue français pour l'utilisation des ces informations. En effet en mode connecté, Unity possède un viewer de diagnostic qui si ces options sont cochées nous permettra de remonter tout un tas de messages sur les erreurs matériels présents dans l'installation, ainsi que sur les erreurs de programmation, nous verrons cet éditeur par la suite (cf. p 58).



L'onglet éditeur, permet de configurer l'ergonomie des sections ladder (nombre de colonnes, commentaires...). Par contre l'éditeur extension de langage permet de pouvoir accéder à davantage de fonctions dans la programmation, les différents paramètres les plus intéressants ont été sélectionnés dans la capture d'écran ci-dessus.

**Remarque :** dans le langage SFC, la case à cocher autoriser plusieurs jetons doit être cochée si on veut représenter sur une même page plusieurs grafset. Si dans un éditeur de programmation, on saisie une ligne de programmation et que l'on détecte une erreur lors de l'analyse, il est possible que certaines options n'ont pas été sélectionnées.

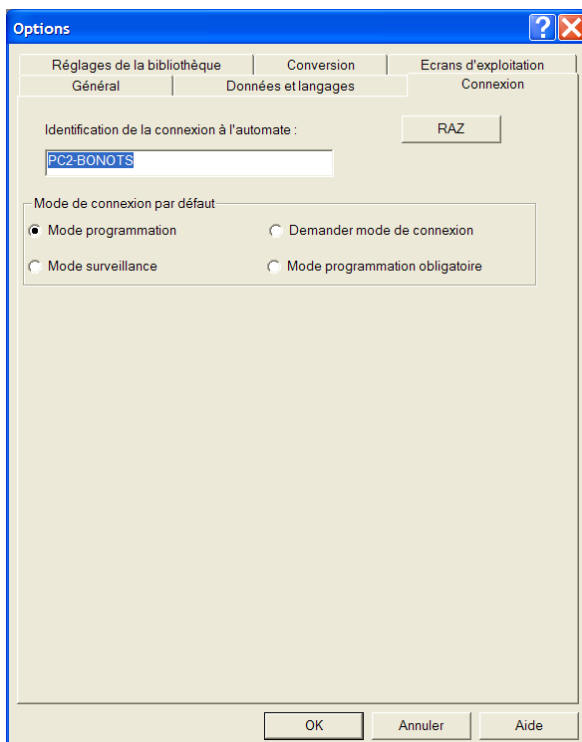
Onglet d'option de l'application, accessible menu Outils, Options



Cet onglet permet de gérer comment les infos d'UPLOAD seront mis à jour, il est vivement conseillé de sélectionner automatique.

Les autres paramètres ne sont pas fondamentaux pour l'utilisation du logiciel, mais permettent de sélectionner l'endroit où l'enregistrement par défaut des fichiers va s'effectuer.

Si la case à cocher restitution du contexte est sélectionnée, le logiciel enregistre la configuration des fenêtres au moment de la fermeture du logiciel et lors de la réouverture de ce projet on retrouvera la même configuration de fenêtre qu'au moment de la fermeture.



L'onglet connexion est important, il permet de définir la manière dont on se connectera à l'automate.

**Mode programmation : permet de lire et de modifier le programme**

**Mode surveillance : accès au programme en mode lecture**

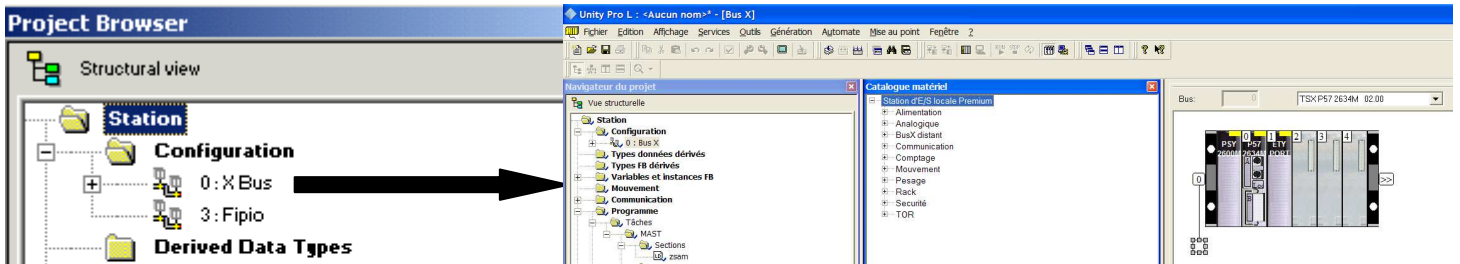
**Demander mode de connexion : le logiciel demande le mode à chaque connexion**

**Conseil : activer le mode programmation**

Configuration matériel de l'application

Depuis le navigateur de projet on pourra :

- configurer le rack de la station local : adresse 0 pour Premium et M340 et adresse 1 pour automate Quantum.
- Configurer les racks d'extension s'il y en a
- Configurer les bus de terrain
- Configurer les réseaux de communication (Ethernet, Modbus, Fipway)



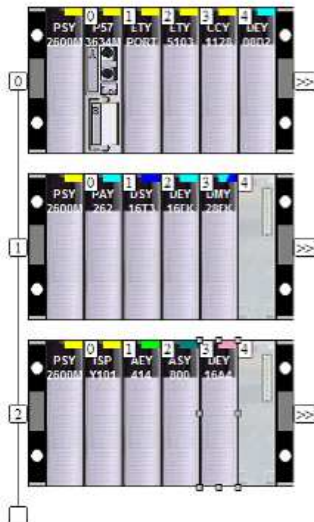
Méthodologie de configuration:

- Choisir ou remplacer le rack
- Choisir la puissance de l'alimentation, module le plus à gauche
- Remplacer le processeur si nécessaire
- Double cliquer sur les emplacements vides du rack et sélectionner le matériel dans le catalogue, ou bien faire du glisser-déposer depuis le catalogue sur l'emplacement libre

Le remplacement du rack s'effectue en sélectionnant le rack avec la souris, puis clic droit remplacer le rack, on choisit alors dans le catalogue celui qui nous convient.

La configuration matérielle logicielle doit correspondre à la configuration réelle sur le rack automate.

Il est possible de remplacer le processeur, pour cela sélectionner le processeur avec la souris, puis clic droit remplacer le processeur, on ne pourra choisir que des processeurs de la même gamme, on ne peut changer un Premium par un M340 et vis versa

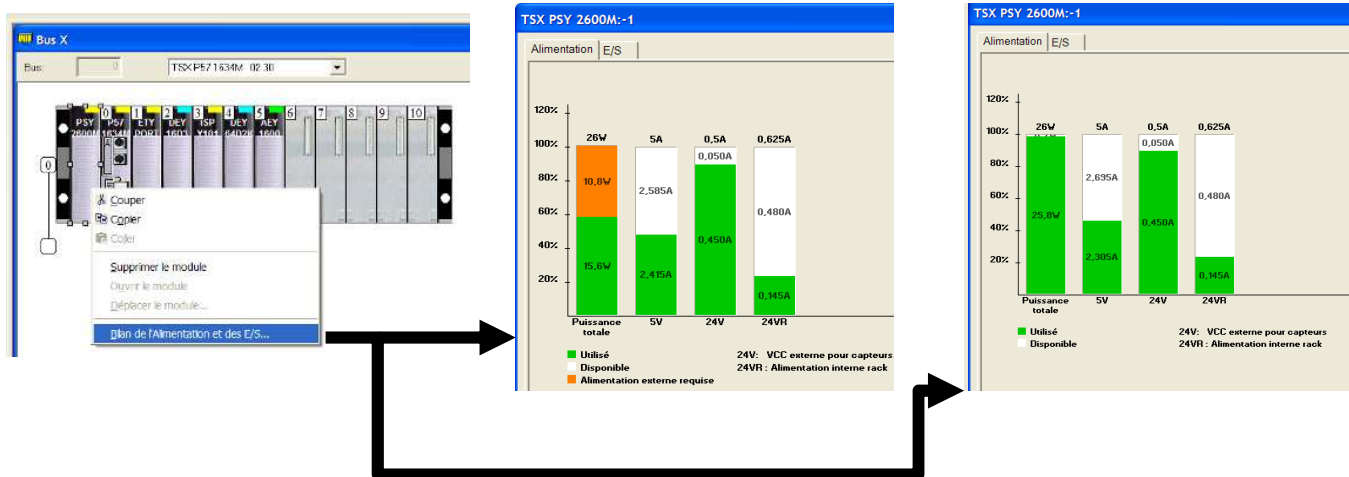


On trouvera ci-contre un exemple de configuration multi-rack, l'éditeur de configuration nous informe du numéro de rack, ici de 0 à 2 et du numéro d'emplacement des modules ici de 0 à 4 pour les 3 racks

L'emplacement module 3 du rack 3 est une carte de 16 entrées, son adressage topologique est donc :

%I2.3.0 à %I2.3.15 (cf. p 9 et 10)

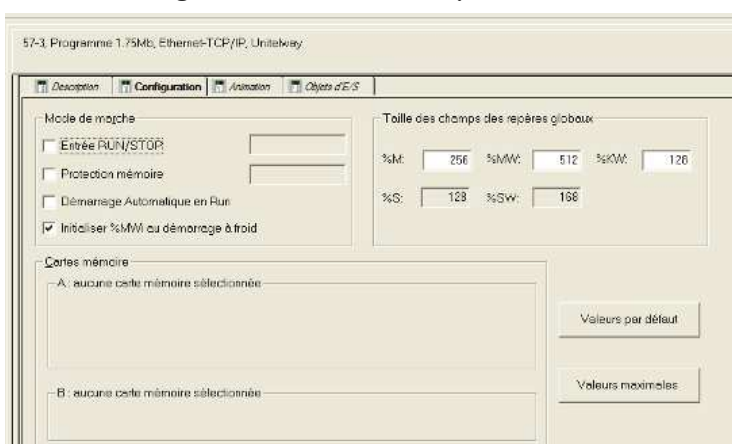
Le module d'alimentation est toujours à gauche pour Premium et M340, le logiciel permet d'établir un bilan de la consommation électrique afin de savoir si l'alimentation est bien dimensionnée, pour cela il suffit de sélectionner l'alimentation avec la souris puis clic droit et choisir bilan de l'alimentation et des E/S.



Sur l'illustration ci-dessus, on voit deux bilans de consommations pour des configurations matérielles différentes, la seconde montre que l'alimentation est correcte, mais on ne pourra rajouter aucune carte car la puissance totale de l'alimentation est déjà quasiment à 100%, alors que dans le premier cas la consommation électrique est trop importante, il faut rajouter une alimentation extérieure. **Si non, on peut prendre une alimentation dans le catalogue plus importante (si bien sur il elle existe !!!!).**

Chaque module peut ensuite être configuré individuellement par un double clic

### Configuration du module processeur



Cet onglet est important, il permet de sélectionner le nombre de variables localisées (variables à adresses topologiques) que l'on attribue à notre application. Sur cet exemple on voit 256 bits internes, 512 mots de 16 bits et 128 constantes de 16 bits. **On rappelle que ces variables sont utilisées pour effectuer de la communication avec des équipements extérieurs (pupitre, automates...), si**

**ce n'est pas le cas il peut être intéressant de diminuer ces valeurs, ou à l'inverse si l'on n'a pas assez de variables pour communiquer de les augmenter.**

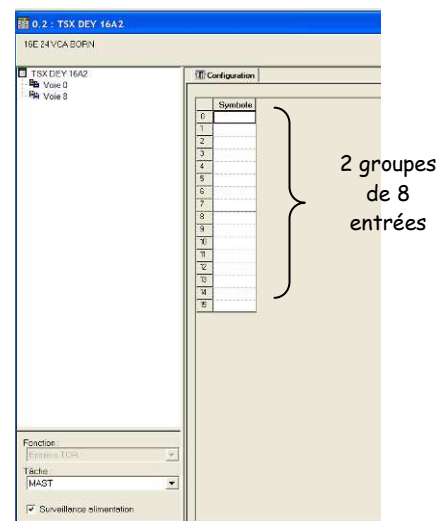
**On peut également sélectionner des cartes mémoires ci-ces dernières sont installées.**

Une option à cocher est le démarrage automatique en run sur un démarrage à froid, par défaut cette entrée n'est pas cochée

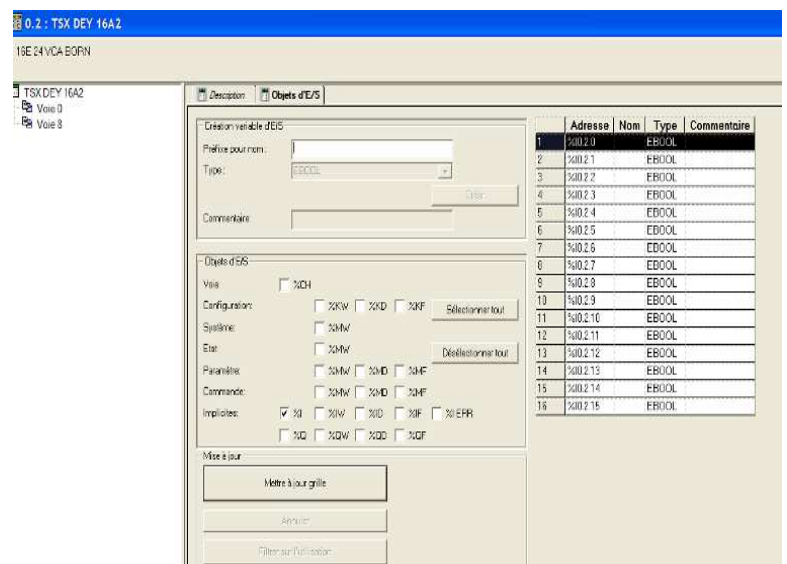
Si l'option protection mémoire est sélectionnée, la protection est activée par un bit d'entrée (à saisir en adresse topologique). Elle interdit le transfert d'un projet dans l'automate et la modification en mode connecté quelque soit la voie de communication. Les commandes d'exécution et d'arrêt sont autorisées. Entrée à 1 -> la carte mémoire et l'application interne ne sont pas protégées. Entrée 0 -> la carte mémoire et l'application interne sont protégées

### Configuration du module d'entrées-sorties T.O.R

Après un double clic sur la carte d'entrées ou de sorties depuis l'éditeur de configuration, la fenêtre à droite s'ouvre, on peut alors sélectionner dans cet écran à quelle tâche de programme seront affectés les entrées ou les sorties (tâche maître (MAST) ou rapide (FAST)). Si des mnémoniques ont été saisies, ils apparaissent dans la colonne symbole. Les entrées sorties sont regroupées par groupe de voies, sur l'exemple on voit deux voies contenant respectivement chacune 8 entrées, car nous avons à faire à une carte d'entrée. L'affectation tâche MAST ou FAST se fait pour une voie complète.



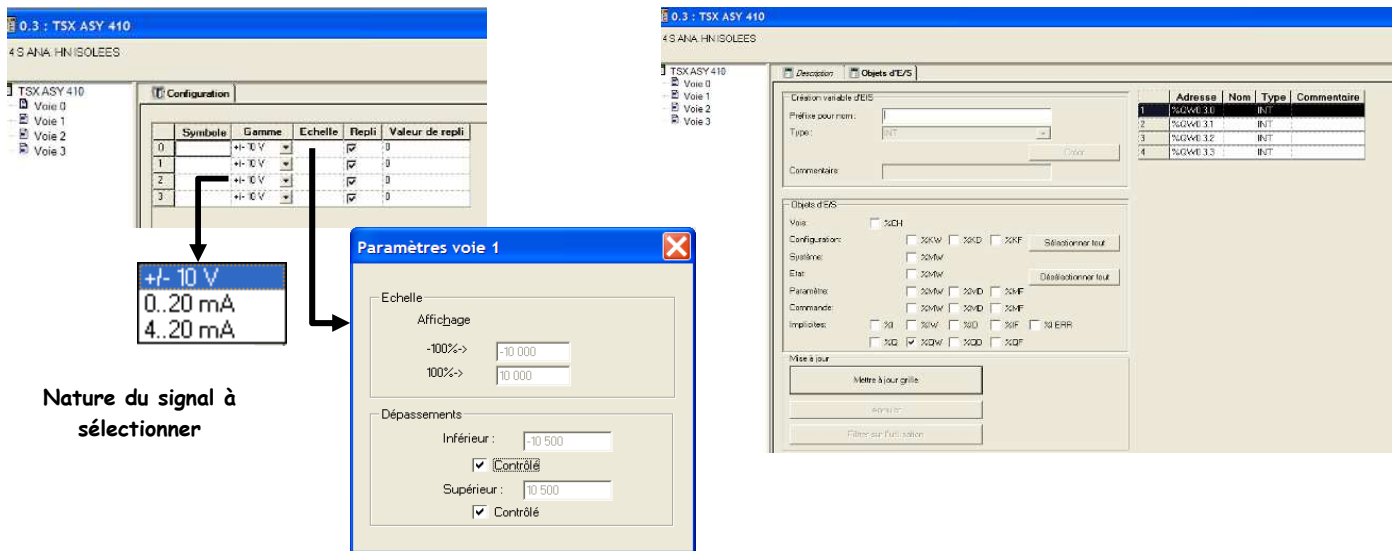
Depuis la fenêtre précédente, en plaçant le nom de la carte (ici TSXDEY16A2) en surbrillance dans l'écran en haut à gauche, on a alors accès à deux onglets : **description et onglet E/S**, le premier est un rappel du type de carte et donne également des informations sur les voyants susceptibles de s'allumer en face avant, le second est beaucoup plus intéressant, il **permet de connaître tous les objets de langage à adressage topologique de la carte**. Par exemple, dans ce cas nous avons une carte d'entrée, il suffit donc de sélectionner I (Q pour une sortie) et de cliquer sur mettre à jour la grille, **toutes les adresses topologiques utilisables sont affichées à droite, cet utilitaire est très pratique si on a un doute dans l'adressage**.





### Configuration du module d'entrées-sorties analogique

Après un double clic sur la carte dans l'éditeur de configuration, on ouvre comme dans le cas des modules T.O.R la configuration des voies analogiques, qui permet le choix du type de signal (0-20 mA, 4-20 mA..., dépend de la carte). En se rendant dans l'onglet E/S de la même façon que pour les modules T.O.R, on peut cette fois avoir accès aux adresses topologiques des voies en sélectionnant IW (entrée analogique) ou QW (sortie analogique) puis en mettant la grille à jour.



Nature du signal à sélectionner

Informations sur la valeur numérique, lies à la sélection de la grandeur électrique

**Remarque** : chaque module à son éditeur de configuration, se rendre à l'intérieur et régler les différents paramètres

### Définition et édition des variables

Une variable est une entité mémoire de types **BOOL**, **EBOOL**, **WORD**, **DWORD**, .....dont le contenu peut être modifié par le programme durant son exécution.

Une **variable localisée** est une variable qui est liée à un module d'entrées sorties ou modules métier ou associé à une zone mémoire spécifique (**INT**, **DINT**.....)

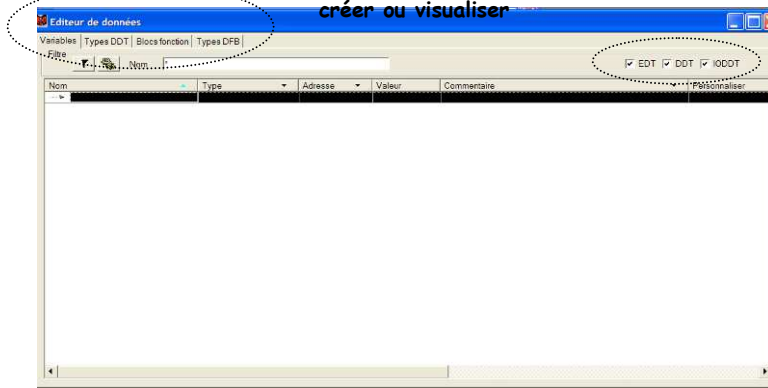
Une **variable non localisée** est une variable qui n'est liée à aucun module ou zone mémoire. Une variable qui n'a pas d'adresses est dite non localisée.

Chaque variable doit être déclarée dans l'éditeur de données.

L'accès à l'éditeur de données s'effectue depuis le navigateur de projet en double cliquant sur **variables et instances FB**.

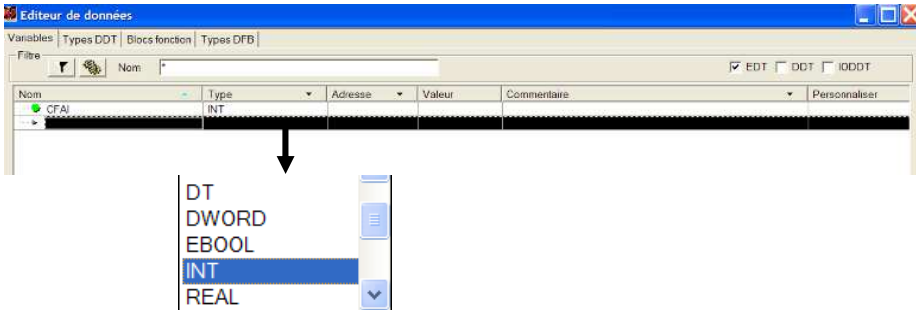
**Support à l'utilisation du logiciel Unity Pro**

**Choix du type de variables à créer ou visualiser**



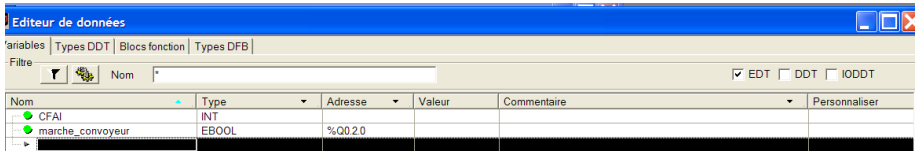
Filtre sur les variables à visualiser

**Exemple de création d'une variable non localisée CFAI de type INT :**



Sélectionner dans le filtre EDT, puis dans la colonne nom, lui donner le nom CFAI, dans la colonne type sélectionner le format INT, valider la variable non localisée INT est créée.

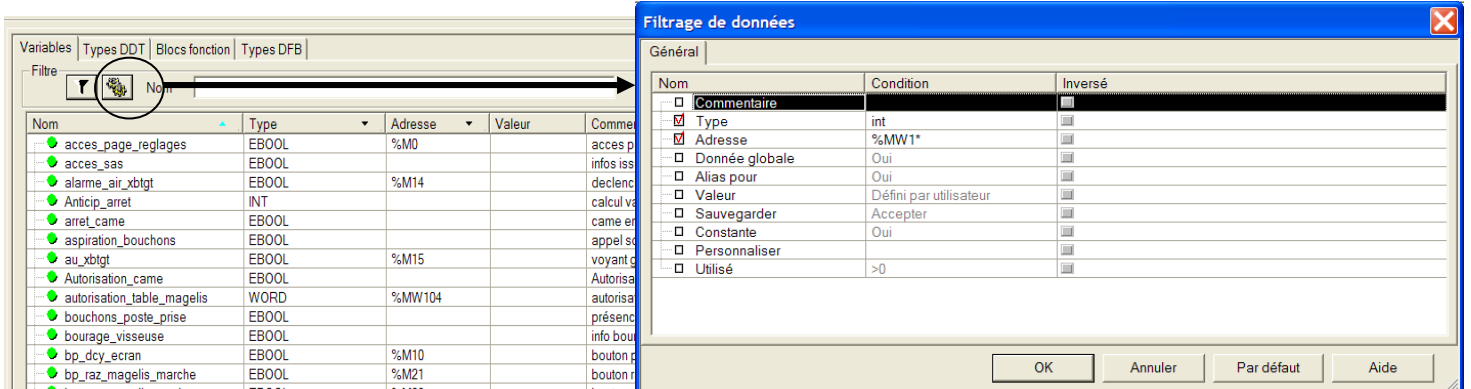
**Exemple de création d'une variable localisée marche convoyeur de type INT**



Il suffit de taper l'adresse topologique, le logiciel se charge automatiquement du format et de lui donner

un nom. On peut ensuite modifier le format dans la limite acceptable par une zone mémoire de 16 bits, à savoir : WORD, UINT (cf. format p 11, 12, 13).

**Ergonomie de l'éditeur de variables**



Le filtrage des données est intéressant quand dans l'éditeur se trouve un grand nombre de variables, on peut alors via les options de filtre ne faire afficher que celle que l'on désire. Dans l'exemple précédent on ne fera afficher que les variable de type INT, dont toutes les adresses débutent par %MW1 (le symbole étoile permet de spécifier toutes).



Via un clic droit lorsque l'on est dans l'éditeur de données, puis **personnaliser colonne** permet de choisir l'affichage désiré. Via les boutons à droite, on peut déplacer les colonnes comme on le souhaite.

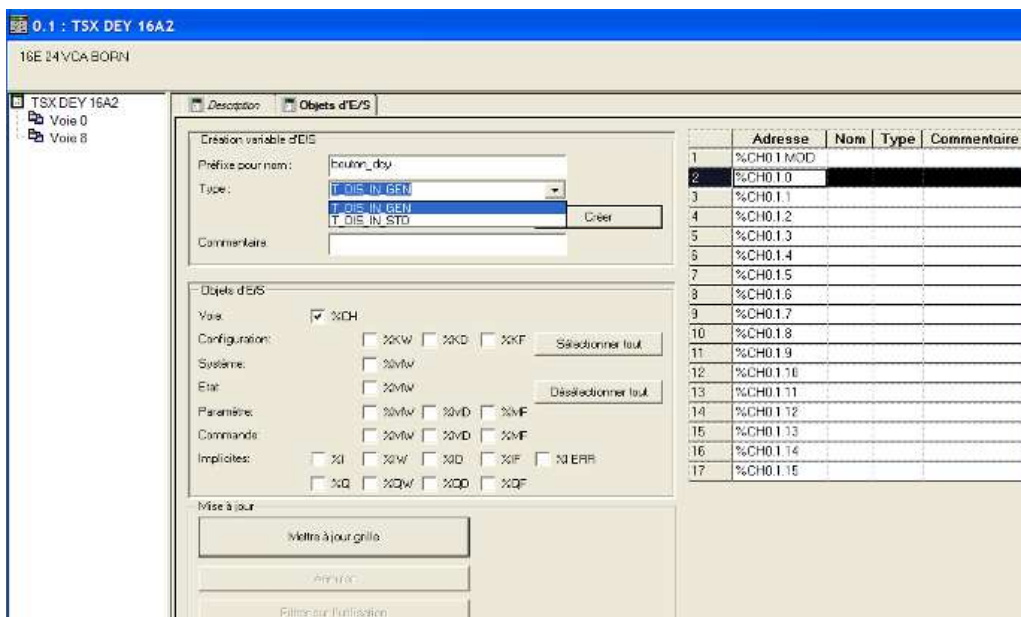
Si toutes les colonnes ne sont pas affichées, il est possible qu'il manque certaines informations sur les variables, pour cela sélectionner la variable puis clic **droit caractéristiques**, la fenêtre ci-contre s'ouvre.



### Création des IODDT affectés aux cartes métier (T.O.R, analogique, ....)

Se rendre dans l'onglet dans l'écran de configuration, puis sélectionner l'onglet E/S de la carte sur laquelle on veut créer les IODDT (cf. p 24). **Sélectionner uniquement %CH (Channel)**, puis mettre à jour la grille. On voit alors apparaître toutes les voies de la carte, sélectionner la voie désirée, puis lui donner un nom, sélectionner le type d'IODDT que l'on veut créer. Par exemple si dessous, la voie 0 est choisie et on l'a nommée bouton\_dcy, on choisit l'IODDT T\_DIS\_IN\_GEN.

**Remarque** : c'est le logiciel qui propose automatiquement le type d'IODDT compatible avec le type de carte sélectionné. En fonction des IODDT, on aura accès à plus ou moins de variables.



A l'issu de la création, cette variable est présente dans l'éditeur de données en autorisant l'affichage des IODDT (cf.p26).

**Résultat après création de la variable bouton\_dcy avec l'IODDT T DIS IN GEN**

Nom	Type	Adresse	Valeur	Commentaire	Personnaliser	R/W Programme	U
bouton_dcy	T_DIS_IN_GEN	%CH0.1.0				<input checked="" type="checkbox"/>	0
CH_ERROR	BOOL	%IO.1.0.ERR		Erreur voie		<input type="checkbox"/>	
VALUE	EBOOL	%IO.1.0.0		Valeur de l'entrée TOR		<input type="checkbox"/>	

**Résultat après création de la variable bouton\_dcy avec l'IODDT T DIS IN STD**

Nom	Type	Adresse	Valeur	Commentaire	Personnaliser	R/W Programme
bouton_dcy	T_DIS_IN_STD	%CH0.1.0				<input checked="" type="checkbox"/>
CH_ERROR	BOOL	%IO.1.0.ERR		Erreur voie		<input type="checkbox"/>
VALUE	EBOOL	%IO.1.0.0		Valeur de l'entrée TOR		<input type="checkbox"/>
EXCH_STS	INT	%MWO.1.0.0		Etat de l'échange		<input type="checkbox"/>
STS_IN_PROGR	BOOL	%MWO.1.0.0.0		Lecture du paramètre d'état en cours		<input type="checkbox"/>
CMD_IN_PROGR	BOOL	%MWO.1.0.0.1		Ecriture du paramètre de commande en cours		<input type="checkbox"/>
EXCH_RPT	INT	%MWO.1.0.1		Rapport de la voie		<input type="checkbox"/>
STS_ERR	BOOL	%MWO.1.0.1.0		Erreur lors de la lecture de l'état de la voie		<input type="checkbox"/>
CMD_ERR	BOOL	%MWO.1.0.1.1		Erreur lors de l'émission d'une commande sur la voie		<input type="checkbox"/>
CH_FLT	INT	%MWO.1.0.2		Défauts sur la voie		<input type="checkbox"/>
TRIP	BOOL	%MWO.1.0.2.0		Défaut externe : disjonction		<input type="checkbox"/>
FUSE	BOOL	%MWO.1.0.2.1		Défaut externe : fusible		<input type="checkbox"/>
BLK	BOOL	%MWO.1.0.2.2		Défaut externe : bornier		<input type="checkbox"/>
EXT_PS_FLT	BOOL	%MWO.1.0.2.3		Défaut externe : alimentation		<input type="checkbox"/>
INTERNAL_FLT	BOOL	%MWO.1.0.2.4		Défaut interne : voie hors service		<input type="checkbox"/>
CONF_FLT	BOOL	%MWO.1.0.2.5		Défaut de configuration matérielle ou logicielle		<input type="checkbox"/>
COM_FLT	BOOL	%MWO.1.0.2.6		Défaut de communication du bus		<input type="checkbox"/>
SHORT_CIRCUIT	BOOL	%MWO.1.0.2.8		Défaut externe : court-circuit		<input type="checkbox"/>
LINE_FLT	BOOL	%MWO.1.0.2.9		Défaut externe : défaut de ligne		<input type="checkbox"/>
CH_CMD	INT	%MWO.1.0.3		Commande voie		<input checked="" type="checkbox"/>
PS_CTRL_DIS	BOOL	%MWO.1.0.3.1		Inhibition de la surveillance de l'alimentation externe		<input checked="" type="checkbox"/>
PS_CTRL_EN	BOOL	%MWO.1.0.3.2		Validation de la surveillance de l'alimentation externe		<input checked="" type="checkbox"/>

On s'aperçoit qu'en fonction de l'IODDT choisit, nous pourrons disposer d'informations supplémentaires, ce choix est laissé au programmeur. L'intérêt d'utiliser des IODDT vient du fait que **c'est le logiciel qui met en relation les variables topologiques par rapport au nom de la variable choisie.**

Par exemple en programmation au lieu de saisir l'adresse topologique **%IO.1.0.0**, il suffira de taper **bouton\_dcy.VALUE**, nous verrons par la suite comment accéder facilement à ses variables depuis l'éditeur de programmation.

**Remarque** : la procédure est à respecter pour chaque voie de cartes utilisées dans la configuration matérielle.

En page suivant un exemple complet d'IODDT crée dans un projet, ou l'on retrouve des IODDT pour des cartes d'entrées T.O.R (**T\_DIS\_IN\_GEN**), des cartes de sorties T.O.R (**T\_DIS\_OUT\_GEN**), des cartes de sorties analogiques (**T\_ANA\_OUT\_GEN**), une carte de communication pour bus Canopen (**T\_COM\_CPP110**). Dans le programme, toutes les adresses topologiques liées à ces cartes auront disparues, on utilisera les variables des différentes IODDT.

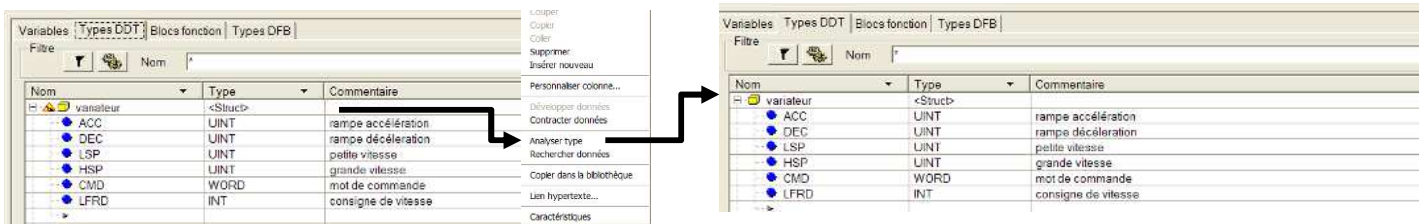
Nom	Type	Adresse	Valeur	Commentaire	Personnaliser	R/W Programme	Utilis
apiration_7v	T_DIS_OUT_GEN	%CH0.3.11		VENTOUSE PRISE BOUCHON 7V	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	4
ibp_dcv_pupitre	T_DIS_IN_GEN	%CH0.2.12		bouton départ cycle pupitre	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	4
canopen	T_COM_CPP110	%CH0.0.1		variable carte maître canopen	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	12
cinq_S0	T_DIS_IN_GEN	%CH0.2.18		VERIN DEPOSE BOUCHON 5C RENTRE	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	3
codeur1	T_DIS_IN_GEN	%CH0.2.0		piste codeur 1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0
codeur2	T_DIS_IN_GEN	%CH0.2.1		piste codeur 2	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0
codeur3	T_DIS_IN_GEN	%CH0.2.2		piste codeur 3	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0
codeur4	T_DIS_IN_GEN	%CH0.2.3		piste codeur 4	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0
codeur5	T_DIS_IN_GEN	%CH0.2.4		piste codeur 5	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0
codeur6	T_DIS_IN_GEN	%CH0.2.5		piste codeur 6	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0
codeur7	T_DIS_IN_GEN	%CH0.2.5		piste codeur 7	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0
codeur8	T_DIS_IN_GEN	%CH0.2.7		piste codeur 8	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0
codeur9	T_DIS_IN_GEN	%CH0.2.8		piste codeur 9	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0
codeur10	T_DIS_IN_GEN	%CH0.2.9		piste codeur 10	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0
comut_auto	T_DIS_IN_GEN	%CH0.2.11		commutateur position automatique	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	3
comut_reglage	T_DIS_IN_GEN	%CH0.2.10		commutateur position réglage	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	3
comut_supervision	T_DIS_IN_GEN	%CH0.2.13		commutateur supervision	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	8
condition_rearm_ok	T_DIS_OUT_GEN	%CH0.3.15		conditions de réarmement remplies	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	1
coupleur_ETY	T_COM_ETYX103	%CH0.1.0			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0
def_var	T_DIS_IN_GEN	%CH0.2.25		défaut variateur	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	2
def_var_convoyeur	T_DIS_IN_GEN	%CH0.2.26		défaut variateur convoyeur	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	2
kmpp	T_DIS_IN_GEN	%CH0.2.24		ARRÊT D'URGENCE	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	14
moteur_bouchons	T_DIS_OUT_GEN	%CH0.3.4		marche variateur convoyeur bouchons	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	2
moteur_came	T_DIS_OUT_GEN	%CH0.3.0		marche variateur moteur came	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	5
moteur_produits	T_DIS_OUT_GEN	%CH0.3.2		marche variateur moteur convoyeur produits	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	3
ouverture_taquets_3C	T_DIS_OUT_GEN	%CH0.3.7		OUVERTURE TAQUET POSITION SOUS VISSAGE 3C	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	9
ouvert_1C	T_DIS_OUT_GEN	%CH0.3.5		ouverture sas par vérin 1C	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	1
ouvert_2C	T_DIS_OUT_GEN	%CH0.3.6		ouverture sas par vérin 2C	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	9
power_removal_lexium05	T_DIS_IN_GEN	%CH0.2.31		pilotage fonction power removal lexium05	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	2
Prise_pose_bouchon	T_DIS_OUT_GEN	%CH0.3.9		DESCENTE PRISE/POSE BOUCHON 5C	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	4
Rentree_blocage_bouchon	T_DIS_OUT_GEN	%CH0.3.10		RENTREE BLOCAGE BOUCHON 6C	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	2
SBS	T_DIS_IN_GEN	%CH0.2.16		BOURRAGE EN SORTIE ETIQUETAGE	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	2
Serrage_pots_4c	T_DIS_OUT_GEN	%CH0.3.8		SERRAGE POT POSTE DE VISSAGE 4C	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	4
Serrage_tete_8c	T_DIS_OUT_GEN	%CH0.3.12		SERRAGE PINCE TETE DE VISSAGE 8C	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	3
SPB	T_DIS_IN_GEN	%CH0.2.17		PRESENCE BOUCHON SUR CONVOYEUR	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	3
SPEE	T_DIS_IN_GEN	%CH0.2.14		PRESENCE POT DEVANT TAQUET 1C	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	1
SPR	T_DIS_IN_GEN	%CH0.2.19		PRESENCE AIR COMPRIME	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	3
SPSB	T_DIS_IN_GEN	%CH0.2.20		PRESENCE POT DEVANT TAQUET 2	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	6
SPSV	T_DIS_IN_GEN	%CH0.2.15		PRESENCE POT AU POSTE DE VISSAGE	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	6
stop_run	T_DIS_IN_GEN	%CH0.2.30		passage automate regroupement en run sur front montant entrée	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0
trigger_camera	T_DIS_OUT_GEN	%CH0.3.17		allumage ou extinction back light	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	2
verme_verte	T_DIS_OUT_GEN	%CH0.3.16		verme de signalisation cycle	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	1
vitesse_came	T_ANA_OUT_GEN	%CH0.4.0		vitesse moteur came	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	5
vitesse_conv_bouchons	T_ANA_OUT_GEN	%CH0.4.3		vitesse convoyeur bouchon	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	5
vitesse_conv_produits	T_ANA_OUT_GEN	%CH0.4.2		vitesse convoyeur produits	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	5

### Création des données dérivées (tableaux ou structures) DDT

On veut créer une structure appelés variateur qui regroupera :

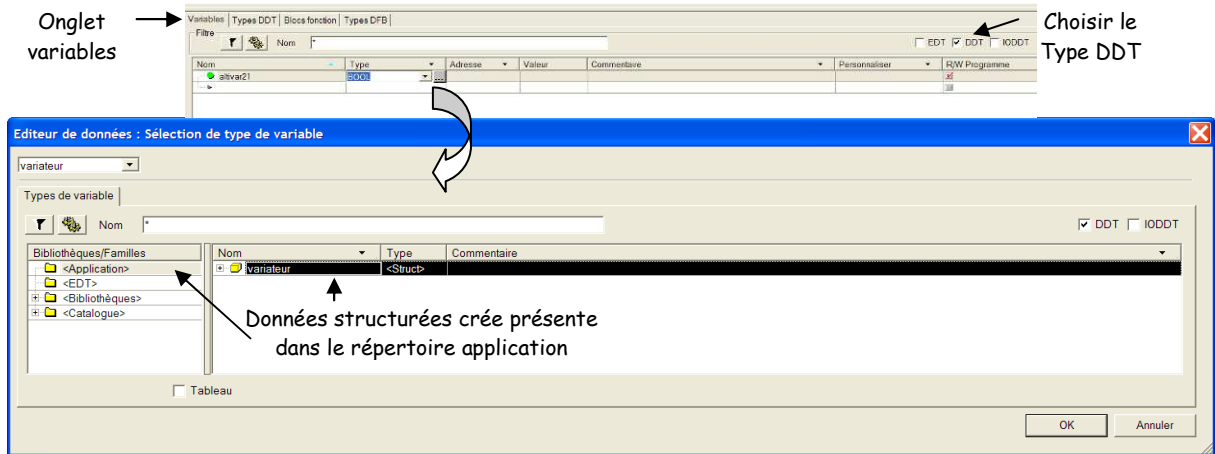
- 4 mots de type UINT (ACC, DEC, LSP, HSP)
- 1 mot de type INT (LFRD)
- 1 mots de type WORD (CMD)

Pour cela depuis le navigateur de projet ouvrir l'éditeur de données et se rendre dans l'onglet types DDT, renseigner les différents champs.



Après avoir saisi les différents champs de la structure que l'on désire créer, en sélectionnant le nom de la structure (ici variateur) puis clic droit et analyser le type, si aucune erreur n'a été décelé, on peut voir que l'icône a changé, la structure est maintenant prête à être instanciée. L'onglet type DDT permet la création de la structure. L'onglet type DFB permet la création de bloc de fonctions dérivées.

On retourne ensuite dans l'onglet variable et l'on donne un nom ici altivar21, altivar31, altivar71, à ces variables on affecte alors l'attribut **variable dérivées variateur crée précédemment**.



On reproduit ces étapes pour les variables altivar31 et altivar71, au résultat on obtient les structures ci-dessous.

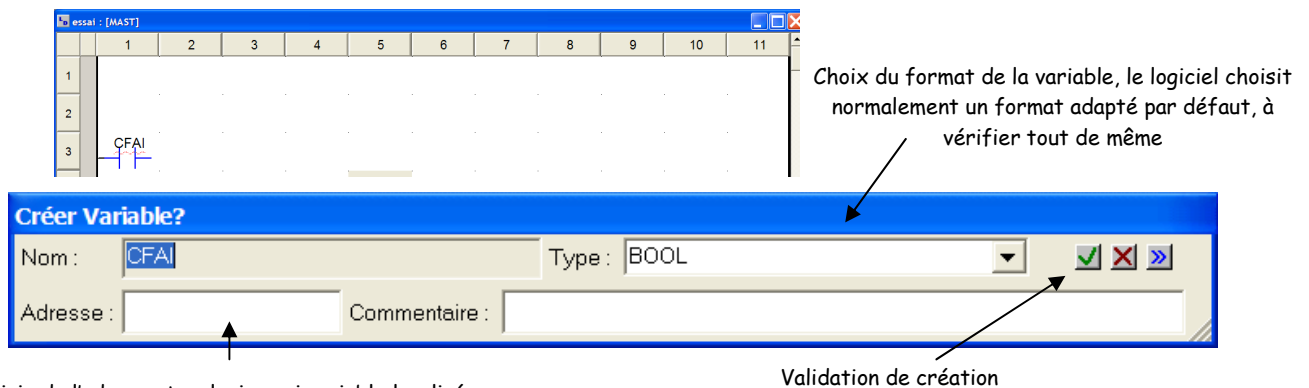
Nom	Type	Adresse	Valeur	Commentaire	Personnaliser	R/W Programme
altivar31	variateur					
ACC	UINT			rampe accélération		
DEC	UINT			rampe décélération		
LSP	UINT			petite vitesse		
HSP	UINT			grande vitesse		
CMD	WORD			mot de commande		
LFRD	INT			consigne de vitesse		
altivar71	variateur					
ACC	UINT			rampe accélération		
DEC	UINT			rampe décélération		
LSP	UINT			petite vitesse		
HSP	UINT			grande vitesse		
CMD	WORD			mot de commande		
LFRD	INT			consigne de vitesse		
altivar21	variateur					
ACC	UINT			rampe accélération		
DEC	UINT			rampe décélération		
LSP	UINT			petite vitesse		
HSP	UINT			grande vitesse		
CMD	WORD			mot de commande		
LFRD	INT			consigne de vitesse		

Ici on a crée 3 variateurs, si on se décide à rajouter un autre variateur, il suffit de créer une nouvelle variable en l'instanciant avec la structure variateur.

**Remarque :** dans l'onglet de type DDT, on aurait également pu créer un tableau (cf. p13), la démarche aurait ensuite été la même que pour les structure : analyse puis on instancie.

### Création de variables depuis les éditeurs de programme

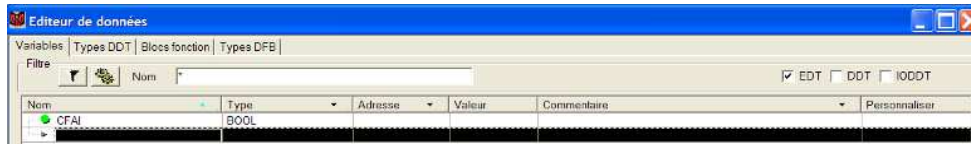
Une variable peut être créée directement depuis des éditeurs de programmation au fur et à mesure du développement du programme. Ci-dessous on place le nom CFAI au dessus d'un contact, si la variable n'existe pas la fenêtre ci-dessous s'ouvre.



Saisie de l'adresse topologique si variable localisée

Validation de création

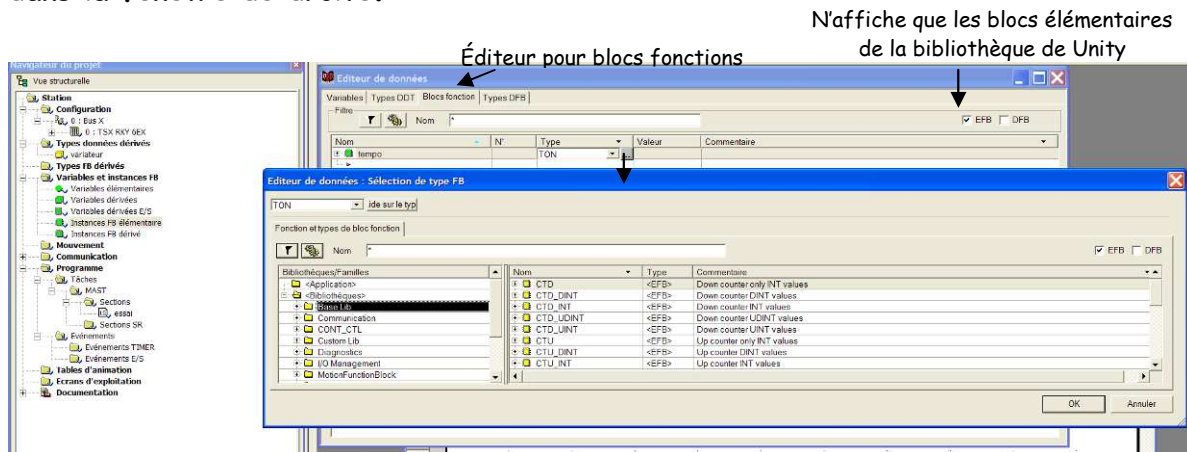
Au final si on retourne dans l'éditeur de configuration des variables, on retrouve notre variable CFAI qui a été crée avec les attributs que nous lui avons donné.



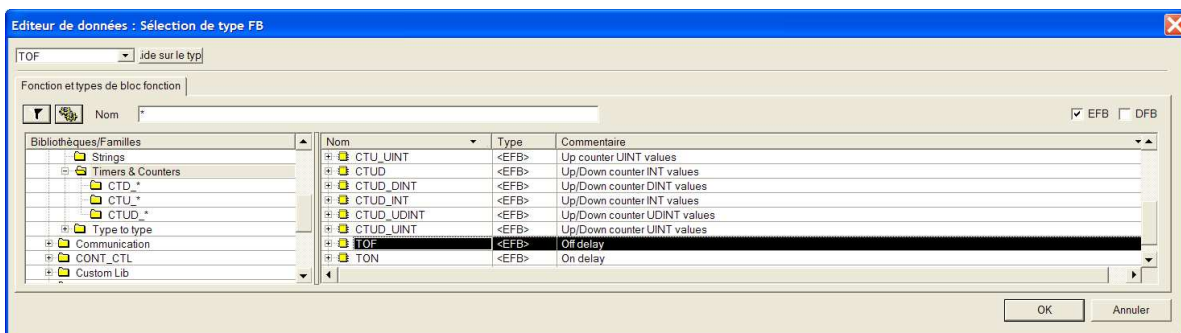
### Création d'instances de blocs fonctions

Les blocs fonctions sont beaucoup plus nombreux que sous PL7, on retrouve bien évidemment les classiques : temporisations, compteurs. Ces blocs sont stockés dans la bibliothèque de Unity, lors de la création d'un programme c'est à l'utilisateur d'aller les chercher.

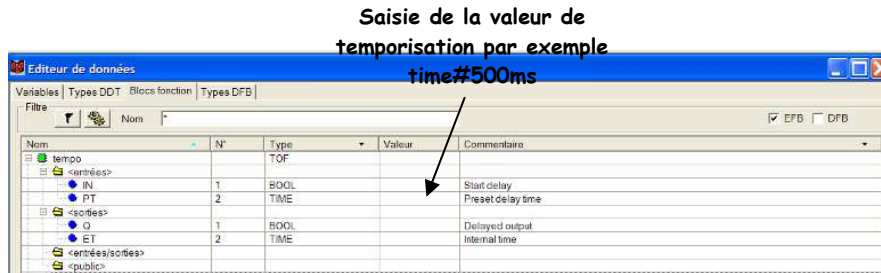
On veut créer une temporisation de type repos (TOF) nommé tempo, après l'ouverture la fenêtre (cf.ci dessous) on peut avoir accès aux bibliothèque de Unity, la plupart des blocs fonctions se trouvent dans le répertoire **Bibliothèque⇒BaseLib**. Concernant les temporisations et les compteurs, ces blocs se trouvent dans le sous répertoire **Timers&Counters** du répertoire **BaseLib**. Tous les blocs disponibles s'affichent alors dans la fenêtre de droite.



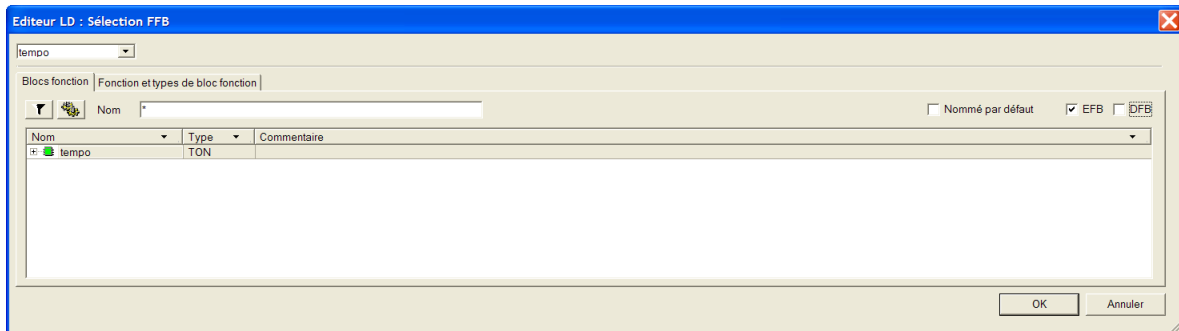
Sélection de la temporisation TOF (repos) ou TON (travail) ou TP (monostable) dans le répertoire indiqué ci-dessus puis valider.



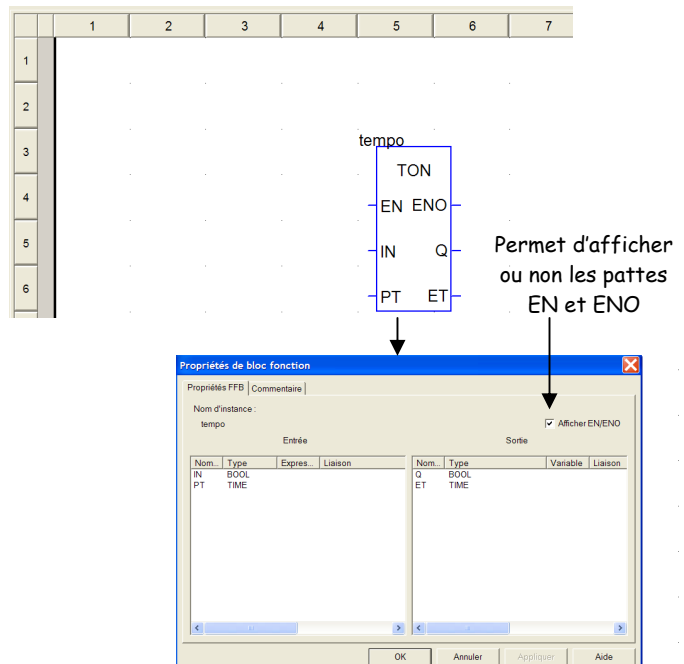
Dans l'éditeur de données sous l'onglet Blocs fonction se trouve le bloc fonction TOF instancié avec le nom tempo, on peut directement saisir la durée de la temporisation au format time (cf. p 12).



La temporisation est réglée, il ne reste plus qu'à l'appeler dans l'éditeur de programme. Depuis cet éditeur sélectionner le menu **édition** puis **sélection de données** la fenêtre ci-dessous s'ouvre, **ne cocher que le type EFB**, apparaît alors tous les blocs fonctions créés dans l'éditeur de données, choisir celui qui nous intéresse, dans le cas ci-dessous on ne retrouve que notre variable tempo.



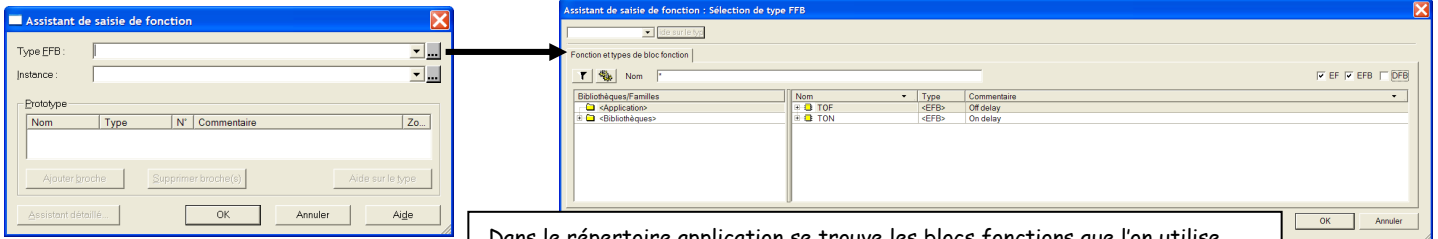
A l'issu, dans un programme ladder placer le bloc ou vous le voulez dans le réseau, on obtient alors les résultats ci-contre. **La valeur de la temporisation n'apparaît pas, mais nous l'avons réglée précédemment. Cette valeur apparaîtra en mode connecté.** Il faut ensuite relier la patte IN à la condition qui lance la temporisation. **Les pattes EN doivent toujours être reliées en permanence à la barre de potentielle gauche, on peut facilement éliminés ces pattes en sélectionnant le bloc dans l'éditeur de programme puis clic droit et propriétés (cf.ci contre).**



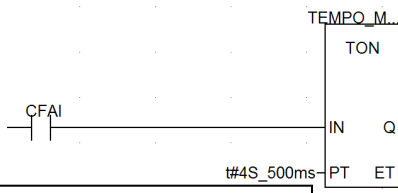


### Création d'instances de blocs fonctions directement depuis l'éditeur de programmation

Dans l'éditeur de programmation sélectionner le menu **EDITION** puis **ASSISTANT DE SAISIE FFB** (ou depuis les icones dans les barres d'outils), la fenêtre ci-dessous s'ouvre.



Dans le répertoire application se trouve les blocs fonctions que l'on utilise dans le programme, dans l'exemple ci-dessus si on veut une tempo TON ou TOF ce n'est pas la peine de retourner les chercher dans la bibliothèque, elle existe déjà. Par contre pour tous autres blocs fonctions, il faut se déplacer dans la bibliothèque, comme vu précédemment.

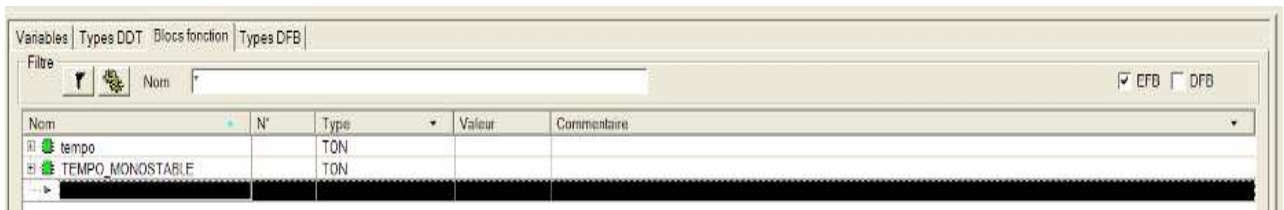


Après placement du bloc fonction, il suffit de compléter les conditions d'activations (patte IN) et la sélection directe sur la patte PT de la de la durée au format time (cf.p12).



Donné le nom de l'instance que l'on veut, ici tempo\_monostable, puis valider.

Si on retourne ensuite dans l'éditeur de données sur l'onglet blocs fonctions, on retrouve notre bloc instancié et ceux préalablement existant.

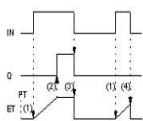


### Fonctionnement des différents blocs TON, TOF, TP et différent compteur

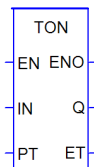
#### TON :

##### Chronogramme

Représentation de la temporisation à l'enclenchement TON :



- (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".



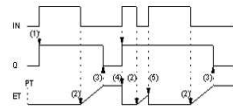
- Le bloc fonction est utilisé pour le retard de mise en marche.
- Les paramètres supplémentaires **EN** et **ENQ** peuvent être configurés

Paramètres d'entrées		
Paramètres	données	signification
IN	BOOL	déclenchement temporisation
PT	TIME	Temps de retard
Paramètres de sorties		
Paramètres	données	signification
Q	BOOL	Sortie
ET	TIME	Horloge interne

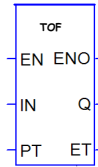
**TOF :**

**Chronogramme**

Représentation de la temporisation au déclenchement TOF :



- (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".



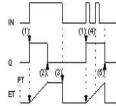
- Le bloc fonction est utilisé pour effectuer un retard au déclenchement
- Les paramètres supplémentaires **EN** et **ENO** peuvent être configurés

Paramètres d'entrées		
Paramètres	données	signification
IN	BOOL	Déclenchement temporisation
PT	TIME	Temps de retard
Paramètres de sorties		
Paramètres	données	signification
Q	BOOL	Sortie
ET	TIME	Horloge interne

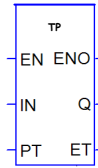
**TP :**

**Chronogramme**

Représentation de l'impulsion TP :



- (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".



- Le bloc fonction est utilisé pour générer une impulsion de durée définie
- Les paramètres supplémentaires **EN** et **ENO** peuvent être configurés

Paramètres d'entrées		
Paramètres	données	signification
IN	BOOL	Déclenchement d'impulsion
PT	TIME	durée de l'impulsion
Paramètres de sorties		
Paramètres	données	signification
Q	BOOL	Sortie
ET	TIME	Horloge interne

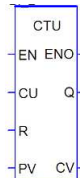
**CTU :**

Ces blocs fonction servent au comptage.

En présence d'un signal "1" à l'entrée R, la valeur "0" 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. Si CV est supérieur ou égal à PV, la sortie Q passe à "1".

Le compteur ne fonctionne que jusqu'aux valeurs maximum du type de données utilisé. Aucun débordement n'a lieu.



- Les paramètres EN et ENO peuvent être configurés.
- Il existe d'autre type de compteur, seul change les formats des données PV et CV qui peuvent être : DINT, UINT, UDINT.
- Les types se nomment CTU\_DINT, CTU\_UINT, CTU\_UDINT

Paramètres d'entrées		
Paramètres	données	signification
CU	BOOL	Comptage
R	BOOL	Remise à 0
PV	INT	Présélection
Paramètres de sorties		
Paramètres	données	signification
Q	BOOL	Sortie
CV	INT	Valeur de comptage

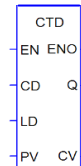
**CTD :**

Ces blocs fonction servent au décomptage.

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 CD, passe de "0" à "1", la valeur de CV est décrémentée de 1.

Si CV est inférieur ou égal à 0, la sortie Q est à 1.

Le bloc ne fonctionne que jusqu'aux valeurs maximum du type de données utilisé. Aucun débordement n'a lieu



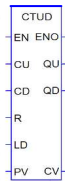
- Les paramètres EN et ENO peuvent être configurés.
- Il existe d'autre type de décompteur, seul change les formats des données PV et CV qui peuvent être : DINT, UINT, UDINT.
- Les types se nomment CTD\_DINT, CTD\_UINT, CTD\_UDINT

Paramètres d'entrées		
Paramètres	données	signification
CD	BOOL	Décomptage
LD	BOOL	Chargement des données
PV	INT	Présélection
Paramètres de sorties		
Paramètres	données	signification
Q	BOOL	Sortie
CV	INT	Valeur de comptage

**Support à l'utilisation du logiciel Unity Pro**

**CTUD :**

Ces blocs fonction servent au décomptage et au comptage.  
 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 supérieur ou égal à PV, la sortie QU passe à "1"  
 Si CV est inférieur ou égal 0, la sortie QD passe à "1".  
 Le bloc ne fonctionne que jusqu'aux valeurs maximum du type de données utilisé. Aucun débordement n'a lieu.



Les paramètres EN et ENO peuvent être configurés.  
 Il existe d'autre type de bloc, seul change les formats des données PV et CV qui peuvent être : DINT, UINT, UDINT. Les types se nomment CTUD\_DINT, CTUD\_UINT, CTUD\_UDINT

Paramètres d'entrées		
Paramètres	données	signification
CU	BOOL	Comptage
CD	BOOL	Décomptage
R	BOOL	Remise à 0
LD	BOOL	Chargement des données
PV	INT	Présélection

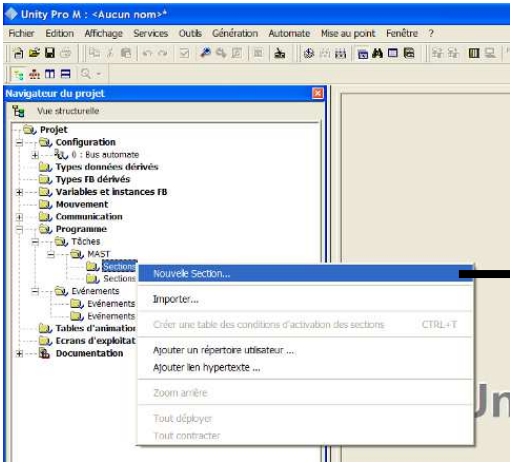
  

Paramètres de sorties		
Paramètres	données	signification
QU	BOOL	Sortie de comptage
QD	BOOL	Sortie de décomptage
CV	INT	Valeur de comptage

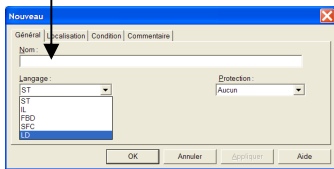
**Remarque ; pour tous les autres blocs se référer à l'aide du logiciel**

**Accès à l'éditeur de programmation**

La création de section et le développement de programme dans les différents langages s'effectuent depuis l'éditeur de programmation. Se rendre dans le répertoire Programme, puis dans la tâche dans laquelle on veut créer un programme (MAST, FAST, Événement), puis clic droit sur section et sélectionner nouvelle section.



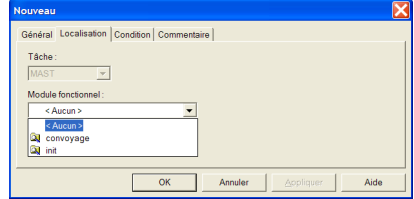
Donner un nom à la section



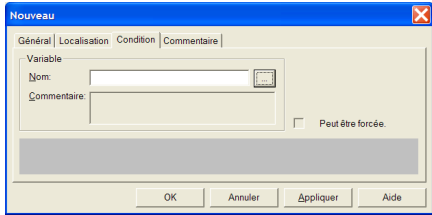
- Sélectionne le langage :
- ST : littéral structure
- IL : list instruction
- FBD : fonctionnal bloc diagram
- SFC : grafcet
- LD : ladder

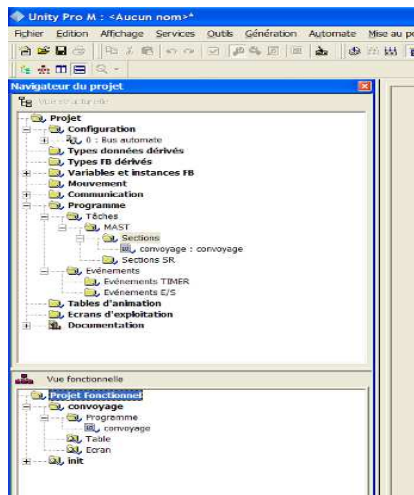
La protection permet de sélectionner la section juste en lecture, ou ni lecture ni écriture, on ne peut alors voire le contenu.

L'onglet localisation, permet d'associer la section à un module de la vue fonctionnelle préalablement créer cf.p 3.



L'onglet condition permet d'assigner une variable comme condition de traitement de la section par l'automate.  
**Variable à 1 : la section n'est pas traitée**  
**Variable à 0 : la section est traitée**  
 Lorsqu'une variable de condition est affectée à une section, en mode connecté si la section n'est pas traitée, apparait un point rouge devant.



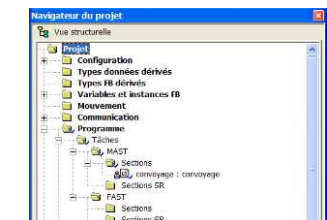
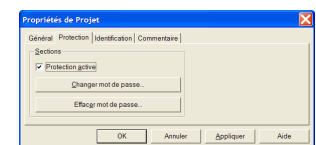
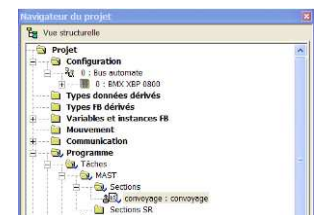


A l'issue de la création, dans ce cas ici une section ladder nommé convoiyage à laquelle on a attribué le module fonctionnel convoiyage.

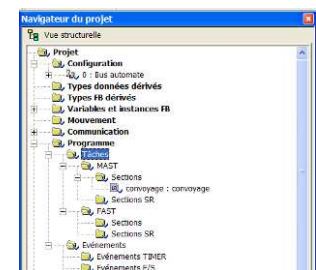
On retrouve notre section dans la vue structurelle et également dans la vue fonctionnelle. On peut voir sur cet exemple qu'un module fonctionnel init a également été créé, module auquel on pourrait associer une ou plusieurs sections.

Si on a placé une section avec une protection, lors de la validation apparaît un cadenas ouvert devant le nom de la section (cf. Ci-contre).

Si l'on veut activer la protection, il faut ensuite cliquer sur Projet, puis clic droit puis propriétés, la fenêtre propriétés du projet apparaît, activer la case à cocher protection active (cf. ci-contre), puis valider. **A l'issue toutes protections associées aux sections sont alors actives.** Lors de l'activation de la case à cocher, un mot de passe sera demandé, si c'est la première fois, c'est alors à vous de définir le mot de passe. **A l'issue un verrou fermé apparaît devant les sections auquel on a affecté un attribut de protection (cf. ci-contre).**



**Remarque :** la plupart des sections sont créées dans la tâche MAST (maitre), pour créer une tâche rapide (FAST), il suffit de sélectionner le répertoire tâche, puis de faire un clic droit nouvelle tâche et de valider, la tâche FAST est alors insérée (cf. ci-contre). **Par défaut un projet contient toujours une tâche maître.**



Les sections SR sont des sous programmes auxquels on peut affecter des sections, ces sections seront alors appelées depuis les tâches MAST ou FAST. Un SR ne peut être appelé que depuis une section qui est dans la tâche à laquelle il est rattaché.

Les traitements événementiels sont des sections de programme qui sont déclenchés par des cartes spécifiques en fonction de la configuration matérielle déclarés dans le logiciel. Par exemple on peut déclarer un traitement événementiel pour des cartes de comptage rapides, lorsque l'on arrive à une certaine valeur, l'automate arrête de lire le

**Support à l'utilisation du logiciel Unity Pro**

programme en cours et traite directement la tâche événementielle. Ces tâches permettent de réagir très rapidement sur des évènements fugitifs.

On pourra trouver des traitements événementiels de type **TIMER**, la section associée à cet évènement sera lancée à chaque intervalle de temps spécifié.

Pour résumer dans un programme on pourra trouver :

- Tache **MAST** : composée de sections pouvant appelées des SR
- Tache **FAST** : composée de sections pouvant appelées des SR
- Tache **AUX** : composé de sections
- **Traitement évènementiel** : soit d'entrées-sorties (ES) ou Timer

Tous les automates acceptent le traitement des taches MAST et FAST, pour le reste cela dépend du type de processeur utilisé (cf. tableau ci-dessous).

Tâches et processus

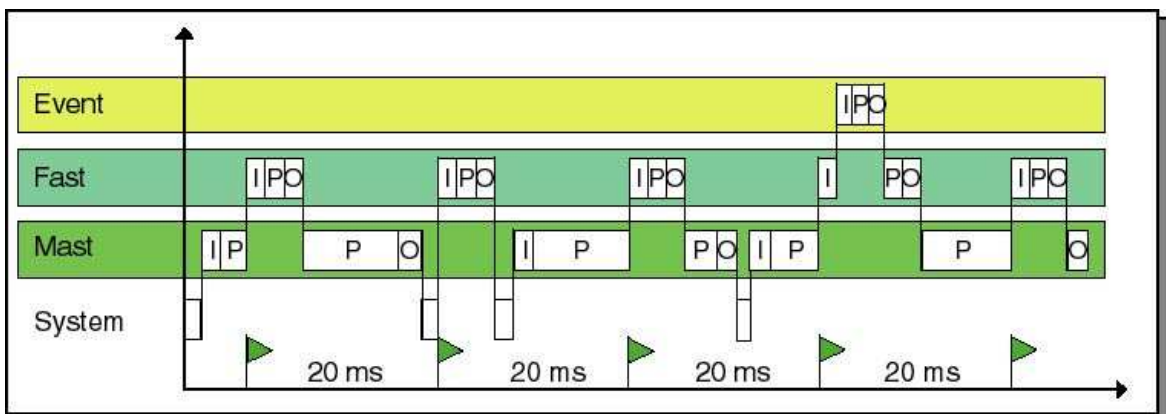
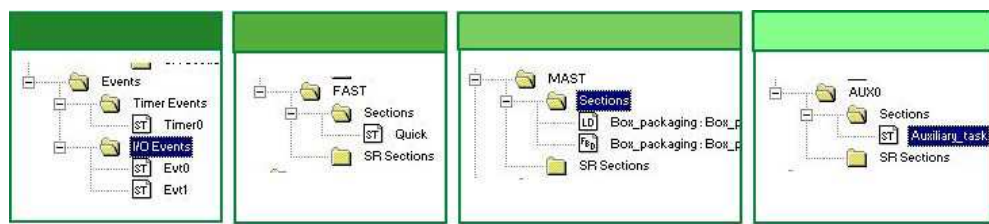
Le tableau suivant décrit les tâches et processus disponibles.

Plates formes	Modicon M340		Premium : TSX			Atrium : TSX		Quantum : 140 CPU	
Processeurs	P34 1000	P34 20**	P57 0244	P57 2**	P57 E**	PCJ 57 204/354	31***	661**	661 60S
			P57 1**	P57 3**	P57 6534		43***	662 60	671 60S
				P57 4**			53***	671 60	
				H57 24M					
				H57 44M					
Tâche maître	X	X	X	X	X	X	X	X	X
cyclique ou périodique									
Tâche rapide	X	X	X	X	X	X	X	X	-
périodique									
Tâches auxiliaires	-	-	-	-	4	-	-	4	-
périodique									
Taille maximum d'une section			64 Ko					16 Mo	-
Traitement évènementiel type	32	64	32	64	128	64	64	128	-
E/S									
Traitement évènementiel type	16	32	-	-	32	-	16	32	-
Timer									
Total des traitements évènementiels type E/S + Timer	32	64	32	64	128	64	64	128	-

X ou valeur tâches ou processus disponibles (la valeur correspond au nombre maximum).  
 - tâches ou traitements indisponibles.

**Description du traitement des tâches par l'automate :**

L'ordre de traitement des sections de programme en fonction des différentes tâches est indiqué ci-dessous :



Légende graphique page précédente :

I : lecture des entrées associée à la tâche

P : lecture des sections de programme associé à la tâche

O : pilotage des sorties associées à la tâche.

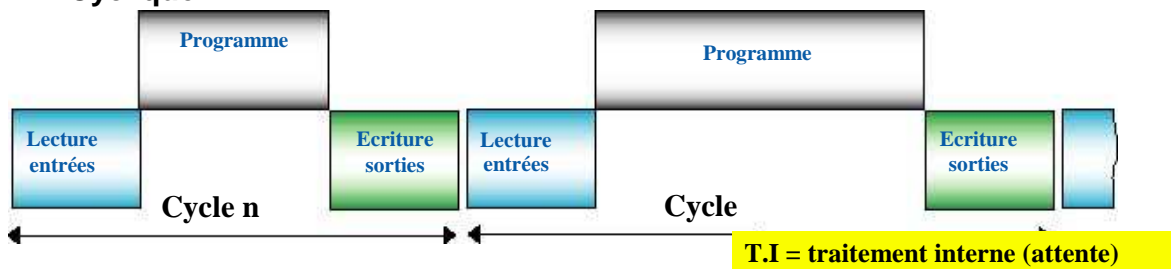
On peut bien voir que le traitement le plus prioritaire est le traitement événementiel, le programme est dérouté pour traiter cette tâches.

**Attention, il faut éviter dans les sections associées aux tâches FAST et événementielle de placer trop de lignes de programmes, car cela sera au détriment du programme principal (MAST). De plus le temps de traitement de tout le programme pourrait s'allonger.**

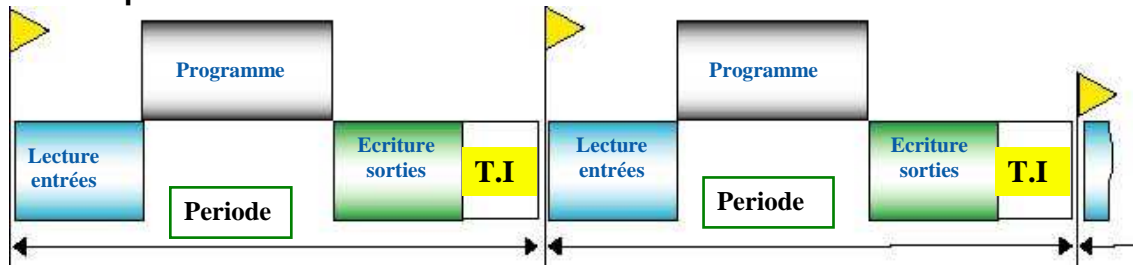
### Fonctionnement cyclique ou périodique

Le choix du type de fonctionnement de la tâche MAST est effectué par configuration.

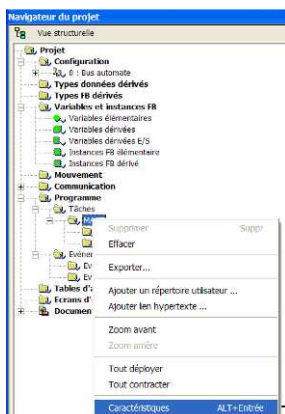
#### ■ Cyclique



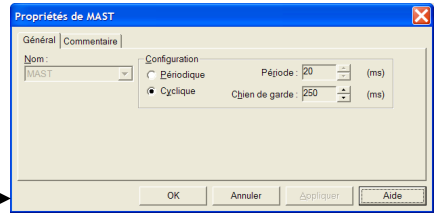
#### ■ Périodique



La période de l'exécution de la tâche maître, en fonctionnement cyclique ou périodique,



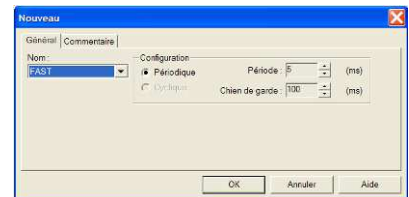
On peut alors sélectionner le mode de fonctionnement. Dans le cas du fonctionnement cyclique, on ne peut régler la période, ce réglage n'est possible que si l'on active le mode périodique. Dans les deux cas on peut régler la valeur du chien de garde



est contrôlée par l'automate (chien de garde) et ne doit pas excéder la valeur définie dans la configuration Tmax (1500 ms). Si le dépassement du chien de garde se produit, l'application est déclarée en erreur, ce qui entraîne l'arrêt immédiat de l'automate. Il ne faut pas configurer une valeur de chien de garde trop

basse sinon le programme n'aura pas le temps de s'exécuter, il est conseillé de laisser la valeur par défaut en général 250 ms.

**Remarque** : au moment de l'insertion d'un tâche maître dans le programme (cf. 36) la fenêtre suivante s'affiche. On s'aperçoit que la tâche rapide (FAST) ne peut être que périodique, c'est à vous de régler sa période, cette tâche est également gérée par un chien de garde à régler.



Sur le graphe de représentation des tâches p 37, on peut voir une tâche maître cyclique, une tâche rapide (FAST) périodique avec une période réglé à 20 ms. Si le temps de la tâche rapide devient supérieur au chien de garde configuré, l'application est déclarée en erreur, ce qui entraîne l'arrêt immédiat de l'automate.

En fonctionnement multitâche, la tâche avec la priorité la plus élevée devra être utilisée en mode périodique pour permettre aux tâches avec la priorité la plus basse de s'exécuter.

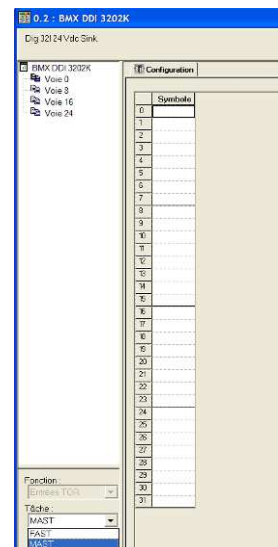
Pour cette raison, seule la tâche avec la priorité la plus faible devrait être utilisée en mode cyclique. Par conséquent, choisir le mode de fonctionnement cyclique pour la tâche maître exclut l'utilisation de tâches auxiliaires qui sont nécessairement périodiques, ces tâches auxiliaires ne sont disponibles que sur certains premium et quantum (cf. tableau p 37), elles sont destinées pour des tâches de traitement lente.

Chaque tâche assure l'écriture et la lecture des entrées/sorties qui lui ont été affectées.

L'association d'une voie, d'un groupe de voies ou d'un module d'entrées/sorties à une tâche est définie dans l'écran de configuration du module correspondant.

La tâche associée par défaut est la tâche MAST.

Sur l'exemple ci-contre de configuration d'une carte 32 entrées dans l'éditeur de configuration, on voit que l'on peut affecter par groupe de 8 entrées (4 voies) soit à la tâche maître ou rapide. Dans tous les éditeurs de configuration d'entrées-sorties, vous aurez la possibilité de choisir l'affectation à la tâche de votre choix. Chaque groupe peut être affecté à une tâche différente.



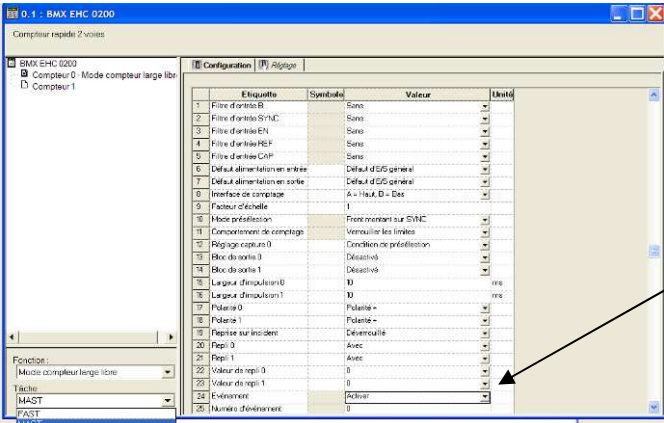
Pour éviter des problèmes ultérieur, si vous affectez une tâche rapide, vérifier que vous l'avez au préalable crée dans le projet cf. 36

## Traitement évènementiel d'entrées sorties (disponibles sur toutes les CPU)

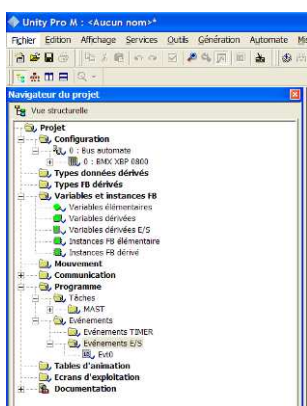
Le traitement de l'évènement est utilisé pour réduire le temps de réponse du programme d'application aux événements pouvant provenir de modules d'entrée-sorties. Attention tous les modules ne présentent pas cette fonctionnalité. Ce traitement sera déclaré dans la l'éditeur de configuration matérielle.

Prenons l'exemple d'une carte de comptage rapide sur lequel est raccordé un codeur incrémental.

Affectation des entrées du codeur à une tâche, de préférence FAST, car les signaux arrivent rapidement



Activation de l'évènement 0, cet évènement sera déclenché quand la valeur de comptage deviendra supérieure à la présélection. Le moment de déclenchement de l'évènement dépend du type de cartes et des fonctions activées.



On peut voir que la section de programme EVT0 (évènement 0) a été créée et programmée en ladder, la création de cette section est à votre charge, le fait d'activer un évènement 0 dans la configuration matériel, ne génère pas la section de programme automatiquement. Lors du déclenchement de l'évènement 0, c'est le programme contenu à l'intérieur de cette section qui sera lue.

Le nombre d'évènement dépend du processeur (cf. tableau p 37). Le traitement évènementiel EVT0 est le traitement le plus prioritaire. Il peut lui-même interrompre les autres

traitements évènementiels.

**Règles de programmation** : les entrées échangées (et le groupe de voies associées) lors de l'exécution du traitement évènementiel sont remis à jour (perte des valeurs historiques, donc des fronts). Il faut donc éviter de tester des fronts sur ces entrées dans les tâches maître (MAST), rapide (FAST) ou auxiliaires (AUX).



**Traitement évènementiel de type TIMER (disponible sur M340, autres CPU cf. tableau p 37)**

Ces évènements sont gérés de manière périodique et entièrement déclenchés par le programme automate et non plus par une carte d'entrées-sorties comme dans le cas ci-dessus.

Il faut tout d'abord créer l'évènement timer dans la vue structurelle du navigateur de projet.

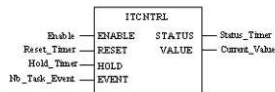
Une section évènement timer sera scruté à chaque fois que la durée sélectionnée dans la configuration (cf.ci-dessous) sera écoulee, dans notre exemple ci-dessous 100ms. Par contre le déclenchement de la temporisation s'effectue par un bloc système de la bibliothèque ITCNTRL. Ce bloc sera à inséré dans une section de la tâche MAST ou FAST.

Les paramètres suivants sont sélectionnés au niveau des propriétés du traitement évènementiel.

Paramètre	Valeur	Valeur par défaut	Rôle
Base de temps	1 ms, 10ms, 100ms, 1 sec	10ms	Base de temps du temporisateur. Note : la base de temps de 1ms est à utiliser avec précaution, risque d'overrun si la fréquence de déclenchement des traitements est trop importante.
Présélection	1..1023	10	Valeur de présélection du temporisateur. La temporisation élaborée vaut : Présélection x Base de temps.
Phase	0..1023	0	Valeur de décalage temporel entre la transition STOP/RUN de l'automate et le premier redémarrage à 0 du temporisateur. La valeur temporelle est égale : Phase x Base de temps.

**Détail de fonctionnement du bloc ICNTRL :**

Représentation en FBD :

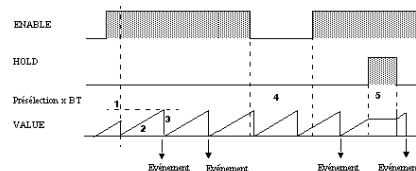


Le tableau suivant décrit les paramètres d'entrée :

Paramètre	Type	Commentaire
Enable	BOOL	Entrée de validation
Reset_Timer	BOOL	Sur état 1 réinitialise le temporisateur.
Hold_Timer	BOOL	Sur état 1 "gèle" l'incrémntation du temporisateur.
Nb_Task_Event	BYTE	Octet d'entrée qui détermine le numéro du traitement évènementiel à déclencher.

Le tableau suivant décrit les paramètres de sortie :

Paramètre	Type	Commentaire
Status_Timer	WORD	Mot d'état.
Current_Value	TIME	Valeur courante du temporisateur.



**Marche normale**

Le tableau suivant décrit le principe de déclenchement des traitements évènementiel de type TIMER (voir chronogramme ci-dessus).

Phase	Description
1	Lors d'un front montant sur l'entrée RESET, le temporisateur est remis à 0.
2	La valeur courante VALUE du temporisateur croît de 0 vers la valeur de présélection d'une unité à chaque impulsion de la base de temps.
3	Un évènement est émis dès que la valeur courante a atteint la valeur de présélection, le temporisateur est remis à 0, puis est denouveau activé. Le traitement évènementiel associé est déclenché, si l'évènement n'est pas masqué. Il peut être différé si un traitement évènementiel de priorité supérieure ou identique est cours d'exécution.
4	Lorsque l'entrée ENABLE est définie sur 0, les évènements ne sont plus émis. Les traitements évènementiel de type TIMER ne sont plus déclenchés.
5	Quand l'entrée HOLD est définie sur 1, le temporisateur est figé, la valeur courante n'évolue plus, tant que cette entrée ne repasse pas à 0.

**Synchronisation de traitement événementiel**

Le paramètre Phase permet de déclencher des traitements événementiels de type TIMER différents à intervalle de temps constant.

Ce paramètre définit un décalage temporel avec une origine de temps absolu, qui est le dernier passage de STOP en RUN de l'automate.

**Condition de fonctionnement :**

- Les traitements événementiels doivent avoir les mêmes valeurs de base de temps et de présélection.
- Les entrées RESET et HOLD ne doivent pas être positionnées à 1.

**Exemple :** 2 traitements événementiels Timer1 et Timer2 à exécuter à 70ms d'intervalle.

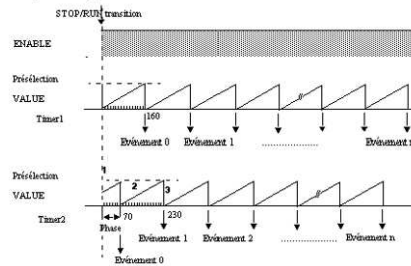
Le premier traitement Timer1 pourra être défini avec une phase égal à 0 et le second Timer2 avec une phase de 70ms (phase de 7 et base de temps de 10ms).

Tout événement déclenché par le temporisateur associé au traitement Timer1

sera suivi 70ms après d'un événement issu temporisateur associé au traitement Timer2

**Chronogramme : Transition STOP/RUN**

Chronogramme de l'exemple décrit ci-dessus avec une même valeur de présélection de 16 (160 ms) pour Timer1 et Timer2.

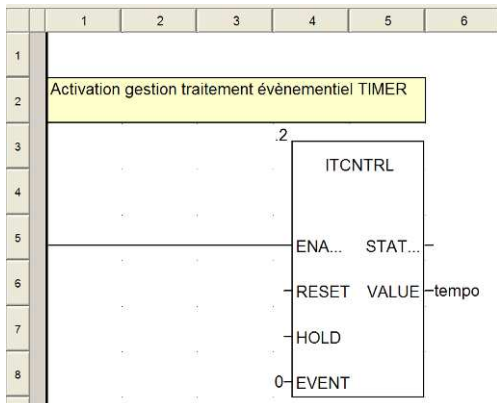


**Fonctionnement après un STOP/RUN pour l'automate**

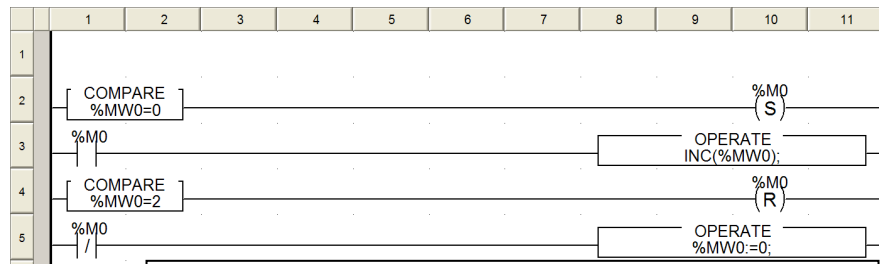
Le tableau suivant décrit le fonctionnement après un passage de STOP en RUN de l'automate (voir chronogramme ci-dessus) :

Phase	Description
1	Sur une transition STOP RUN de l'automate la temporisation se déclenche de façon à ce que la valeur de présélection soit atteinte au bout d'un temps égal à la Phase x base de temps, le premier événement est alors émis.
2	La valeur courante VALUE du temporisateur croît de 0 vers la valeur de présélection d'une unité à chaque impulsion de la base de temps.
3	Un événement est émis dès que la valeur courante a atteint la valeur de présélection, le temporisateur est remis à 0, puis est denouveau activé. Le traitement événementiel associé est déclenché, si l'évènement n'est pas masqué. Il peut être différé si un traitement événementiel de priorité supérieure ou identique est cours d'exécution.

**Exemple de programmation :**



Section ladder crée dans la tache MAST



Évènement Timer0 cf. page précédente, base de temps = 10ms, présélection : 10 ms

Sur cet exemple de programmation l'évènement Timer0 sera déclenché toute les 100ms, à chaque déclenchement de l'évènement Timer0, la section de programme n'est lue que sur un cycle automate.

**Fonctionnement du programme :**

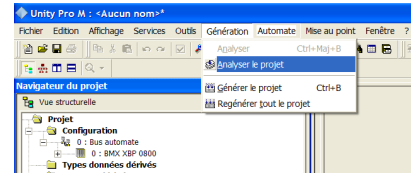
Au bout de 100ms le bit %M0 est mis à 1 et la valeur du mot %MW0 prend la valeur 1 (incrémenter), 100 ms plus tard la valeur du mot %MW0 passe à 2 et le bit %M0 est remis à 0 ainsi que la valeur de %MW0, 100ms on est reparti pour le même fonctionnement qui vient d'être énoncé.

Par conséquent le bit %M0 est mis à 1 pendant 100 ms et remis à 0 pendant 100ms.

**Remarque** : les traitements événementiels EVTi déclenchés par des modules d'entrées/sorties, sont prioritaires sur les traitements événementiels déclenchés par temporisateurs.

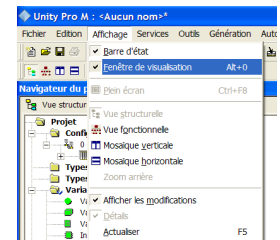
### Validation et transfert d'application

Avant de pouvoir transférer un programme dans l'automate, il est nécessaire que ce dernier ne contienne pas d'erreur, c'est pourquoi il faut effectuer une analyse du programme. Les fonctions d'analyse sont disponibles depuis le menu génération, comme indiqué ci-contre. On trouvera la commande :

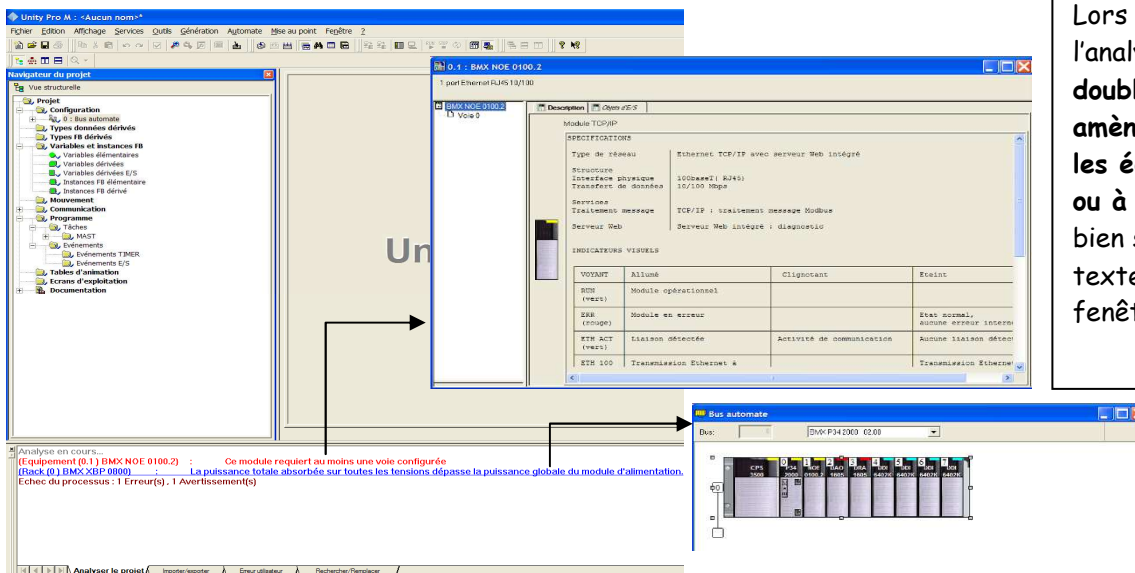


- **Analyser** : permet de s'assurer que dans les sections actuellement ouverte (fenêtre d'éditeur de programme) du projet, il n'y est pas d'erreur, comme symbole indéfini, élément non relié. **Les résultats de l'analyse seront affichés dans la fenêtre de visualisation située en bas de l'écran.**
- **Analyser le projet** : scanne la totalité du projet, les éditeurs de programme comme les éditeurs de configuration. **Les résultats de l'analyse seront affichés dans la fenêtre de visualisation située en bas de l'écran.**

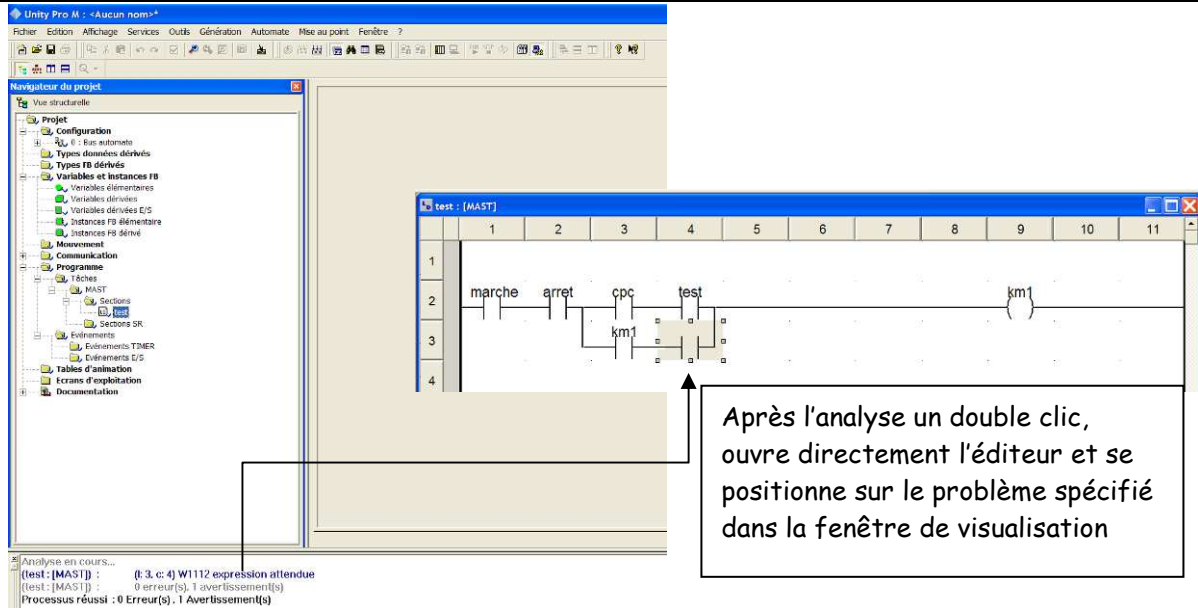
**Remarque** : pour visualiser les résultats dans la fenêtre de visualisation, il faut que celle-ci soit ouverte. Pour cela se rendre dans le menu affichage et sélectionner fenêtre de visualisation



Dès que l'on a créé une configuration matérielle ou que l'on effectue une modification de programme, il est fortement recommandé de lancer une analyse, afin d'éviter d'avoir en fin de projet trop d'erreurs à corriger



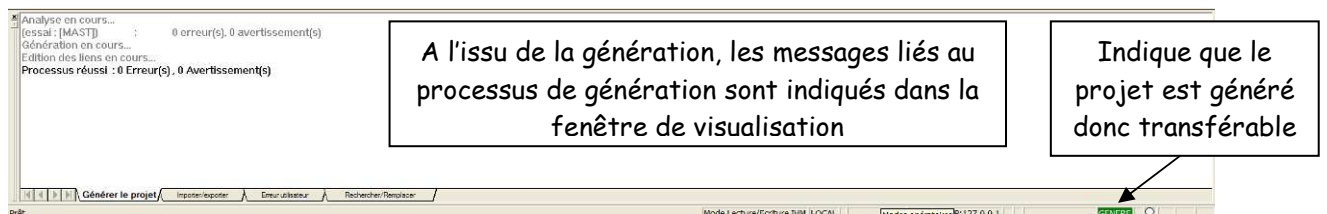
Lors du résultat de l'analyse du projet, un double clic sur la ligne amène directement sur les éditeurs à modifier ou à configurer. Il faut bien sûr s'aider du texte apparaissant la fenêtre de visualisation.



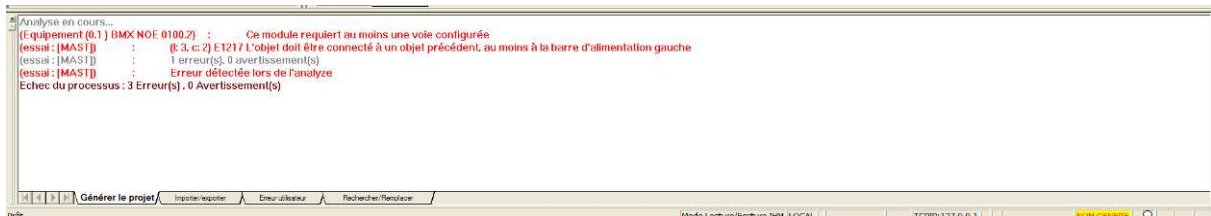
Pour permettre de pouvoir transférer un programme, il est nécessaire de l'avoir généré, la génération crée un fichier qui peut être chargé dans l'automate. Les commandes de génération sont disponibles par le menu génération :



- **Générer le projet** : ne génère que la partie du projet qui a été modifiée. Cette commande n'est disponible que si une génération complète a déjà été effectuée.
- **Regénérer tout le projet** : compile la totalité du projet, il est nécessaire de l'effectuer au moins une fois



### Exemple d'erreur détectée lors de l'analyse



**Remarque** : les erreurs détectées bloquent le processus de génération, par contre si des avertissements sont émis, la génération se déroule quand même. Un double clic sur l'erreur ou l'avertissement amène directement dans l'éditeur en question. Il est conseillé

de ne déclencher le processus de génération qu'après une analyse du projet ne présentant pas d'erreurs.

Après que le processus de génération se soit déroulé sans erreurs, on peut transférer le programme dans l'automate. Il faut en suite vérifier les paramètres de connexion pour savoir à travers quel support on transfère le programme automate. **Pour cela il suffit de se rendre dans le menu Automate-Définir l'adresse**, la fenêtre ci-dessous s'ouvre.

La connexion à l'automate peut s'effectuer directement par câble ou depuis un réseau.

#### Connexion à un automate M340 :

- Connexion direct par câble **USB**
- Connexion par coupleur **Ethernet** (TCPIP)

#### Connexion à un automate Premium

- Connexion direct par câble **TSXPCX3030** ou **TSXCUSB485** sur prise terminal ou AUX, ou **TSXPCX3031** sur prise TER uniquement. (driver Unitelway et Modbus)
- Connexion par réseau Ethernet (driver XIP)
- Connexion USB (disponibles que sur certaines CPU)

**Remarque** : il existe d'autres modes de connexions, se reporter à l'aide du logiciel pour davantage d'informations. Les drivers unytelway, Modbus, TCPIP, et XIP sont les plus utilisés. Pour pouvoir les sélectionner, il faut qu'ils aient été au préalable installés.

Porte d'accès à l'automate. Pour un accès direct depuis un des câbles cités ci-dessus, toujours laissé SYS

Choix du driver à utiliser pour se connecter.

#### Configuration driver Unytelway pas encore disponible sur M340

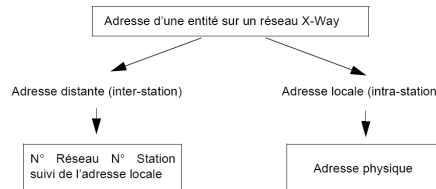
StationID	Port	Password	PhoneNumber	Parity	Data
(Default)	COM1			Odd	8
3	USB			Odd	8
	rogalis			Odd	8
	XBRT			Odd	8

Dans la fenêtre ci-contre, il suffit d'aller chercher la référence du câble que l'on utilise TSXPCX3030 ou TSXCUSB485. Laisser base 1 (adresse de départ du pilote) et nombre 3 (la console nécessite 3 adresses sur le réseau unytelway). Ces références de câbles ne figurent que si les drivers respectifs ont été installés. **Toujours sélectionné le driver correspondant au câble utilisé.**

**Configuration driver XIP**

Le driver XIP permet l'accès à un automate par l'intermédiaire d'un réseau Ethernet en utilisant le protocole d'adressage X-WAY.

Le format général défini pour décrire l'adresse d'une entité destinataire faisant partie d'un réseau X-Way est détaillé ci-dessous :



Le champ "Numéro de réseau" indique le numéro du réseau de la station destinataire de l'échange. **Il doit être compris entre 0 et 127.**

Le champ "Numéro de station" indique le numéro de la station destinataire de l'échange. **Il doit être compris entre 0 et 63.**

Saisir l'adresse X-Way de l'automate au format {x.y}SYS ou x : n° de réseau et y : n° de station

Sélection du driver XIP

On peut ajouter les équipements sur lequel on veut se connecter par simple connaissance de l'adresse IP et X-Way.

Adresse X-Way du PC. Le Pc doit être dans le même numéro de réseau que l'automate et avoir un numéro de station univoque

Sélection de la carte réseau du PC ou de la carte sans fil, si l'on dispose d'un PC équipé et bien sur d'une borne d'accès WIFI pour la connexion au réseau

Vérifier que le driver soit en service par la présence d'un point vert en bas de la fenêtre. Si un point rouge est présent, il faut démarrer le driver par la commande test-démarrer, à l'issu le point devient vert.

**Vérification de l'adresse IP du PC et celle du driver XIP en cas de problèmes de connexion**

```

C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\ADMIN\ipconfig

Configuration IP de Windows

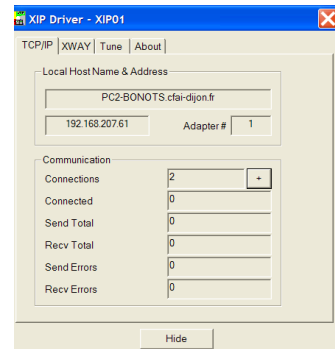
Carte Ethernet Connexion au réseau local:

    Suffixe DNS propre à la connexion : cfai-dijon.fr
    Adresse IP . . . . . : 192.168.207.61
    Masque de sous-réseau . . . . . : 255.255.255.0
    Passerelle par défaut . . . . . : 192.168.207.150

Carte Ethernet Connexion réseau sans fil:

    Statut du média . . . . . : Média déconnecté
    
```

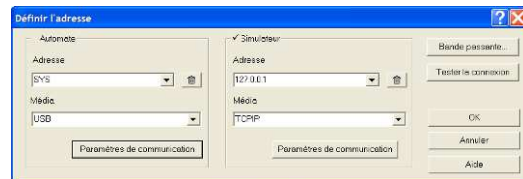
Lancer la commande exécuter depuis le menu démarrer puis saisir cmd, la fenêtre DOS s'affiche, taper **IPCONFIG**



Double cliquer dans la barre vers l'horloge sur l'icône du driver XIP, la fenêtre ci-contre s'ouvre. L'adresse IP doit être la même que celle donnée avec la commande **IPCONFIG**

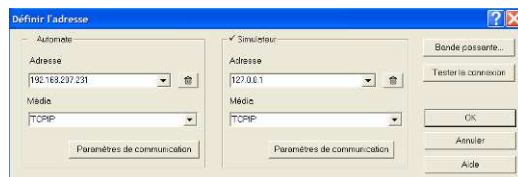
### Configuration driver USB disponible sur tous les M340 et quelques Premium

Il suffit de sélectionner le driver USB (par défaut sur M340) et dans l'adresse de taper SYS (par défaut). Il n'y a pas d'autres réglages à effectuer.



### Configuration driver Ethernet IP pour les CPU M340 native Ethernet et certains coupleur Premium et M340.

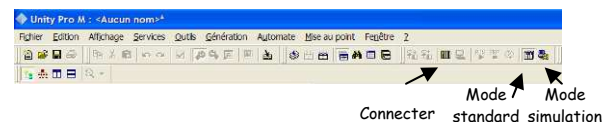
Il suffit de sélectionner le driver TCP/IP et dans l'adresse de taper l'adresse IP de l'automate. Il n'y a pas d'autres réglages à effectuer.



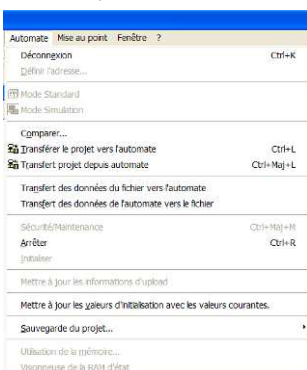
### Connexion et transfert de l'application

Une fois l'application générée, il suffit de se connecter à l'automate en ayant pris soin au préalable de vérifier par quel driver on se connecte (cf. ci-dessus). **Attention Unity possède un mode de simulation qui permet le test de votre programme sans avoir à disposition l'automate, on peut donc se connecter à l'automate physiquement ou au simulateur.**

- Sélectionner le mode standard (automate) ou simulation à l'aide des icones indiqués ci-contre.
- Cliquer sur **connecter**



Une fois la connexion établie, plusieurs choix s'offrent à l'utilisateur :



**Déconnexion** : coupure liaison avec l'automate ou le simulateur

**Comparer** : compare les projets entre le PC et l'automate

**Transférer** : permet le transfert dans l'A.P.I ou depuis l'A.P.I

**Arrêter** : mise automate en stop

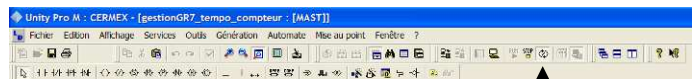
**Exécuter** : mise automate en run

**Initialiser** : initialise toutes les valeurs et les étapes de grafcet

**Sauvegarde du projet** : permet d'enregistrer, de comparer ou de restituer sur un M340 le projet sur la carte SD, cette manipulation est possible sur certains premium équipé d'une carte mémoire : TSX MFP B 096K

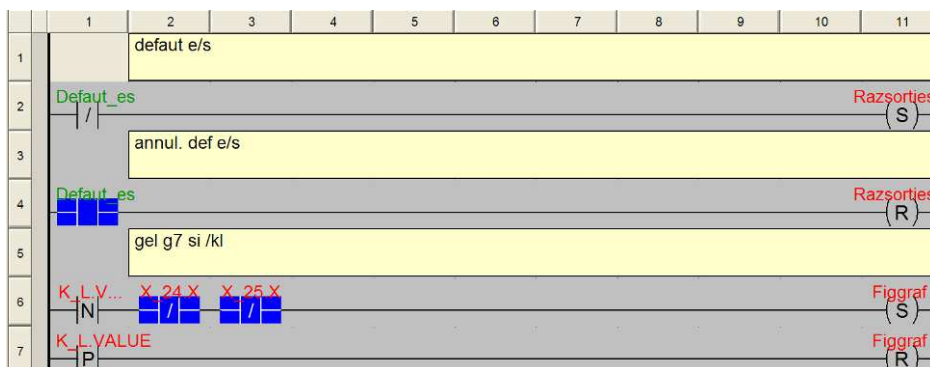
- Après avoir choisi l'option transférer le projet dans l'A.P.I et passer ce dernier en mode RUN, on peut aller voir directement l'animation dynamique de tous les éditeurs de programme et de configuration.

Après le transfert de l'application et la mise en RUN, il est nécessaire d'activer la visualisation des sections de programme par sélection de l'icône



### Animation d'un programme LADDER

#### Exemple de sections de programme en ladder



La couleur de fond de l'éditeur est le gris.

Les couleurs utilisées en mode local pour les différents éléments (instructions, variables, commentaires) sont remplacées :

- pour le type booléen :
  - vert si la variable est TRUE (1)
  - rouge si la variable est FALSE (0).
- jaune pour les types numériques

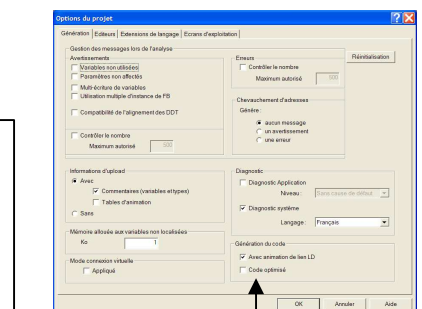
Il existe trois types de lien :

- les liens booléens entre contacts et bobines
- les liens booléens entre les blocs fonction
- les liens numériques entre blocs fonction

Deux sortes d'animations sont possibles selon l'option sélectionnée dans le menu outils-options du projet :

- avec animation de liens pour lesquelles :

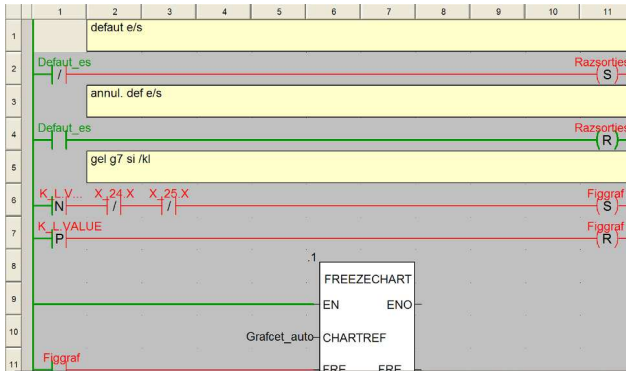
Les liens booléens entre les contacts et les bobines s'affichent en vert ou rouge selon que l'évaluation schéma à contacts en amont renvoie la valeur OUI (1) ou NON (0).



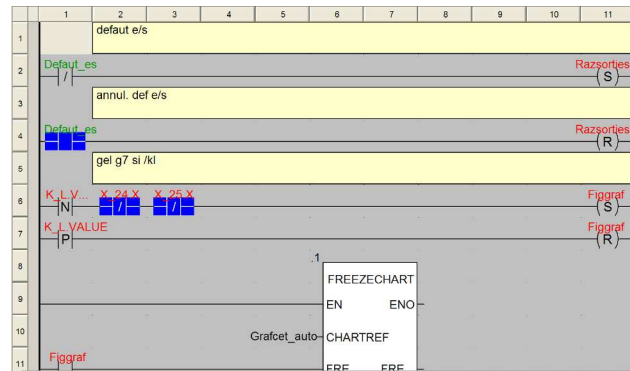


- sans animation de liens où les contacts fermés et les bobines déclenchées s'affichent en vidéo inverse.

**Remarque** : l'animation de lien ralentit l'exécution du projet



Animation programme avec option animation de lien

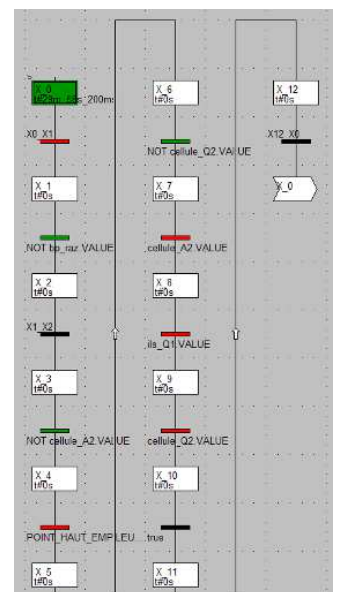


Animation programme avec option sans animation de lien

### Animation d'un programme SFC

La couleur de fond de l'éditeur est grise.

- **Pour les étapes**
  - verte si l'étape est active
  - blanche si l'étape est inactive
  - jaune si le temps d'activité de l'étape est inférieur au temps minimal programmé.
  - rose si le temps d'activité de l'étape est supérieur au temps maximal programmé.
- **Pour les transitions associées à un élément booléen ou une expression booléenne simple :**
  - verte si l'élément ou l'expression est TRUE (vraie)
  - rouge si l'élément ou l'expression est FALSE (fausse).
- **Pour les transitions associées à une section :**
  - noire tant que l'étape précédente est inactive
  - verte si les conditions dans la section sont TRUE (vraies)
  - rouge si les conditions dans la section sont FALSE (fausses)



## Animation d'une List Instruction ou Litteral Structure

La couleur de fond de l'éditeur est grise. Le texte apparaît en noir. Les commentaires sont en vert. Le noir utilisé pour les variables et les instructions en mode local est remplacé. Pendant l'animation, les variables et les instructions sont affichées dans les couleurs suivantes :

- verte si la variable est TRUE (vraie)
- rouge si la variable est FALSE (fausse).
- Jaune pour les autres types

Les types de données autres que booléen ne sont pas animés dans l'éditeur mais dans une fenêtre de visualisation.

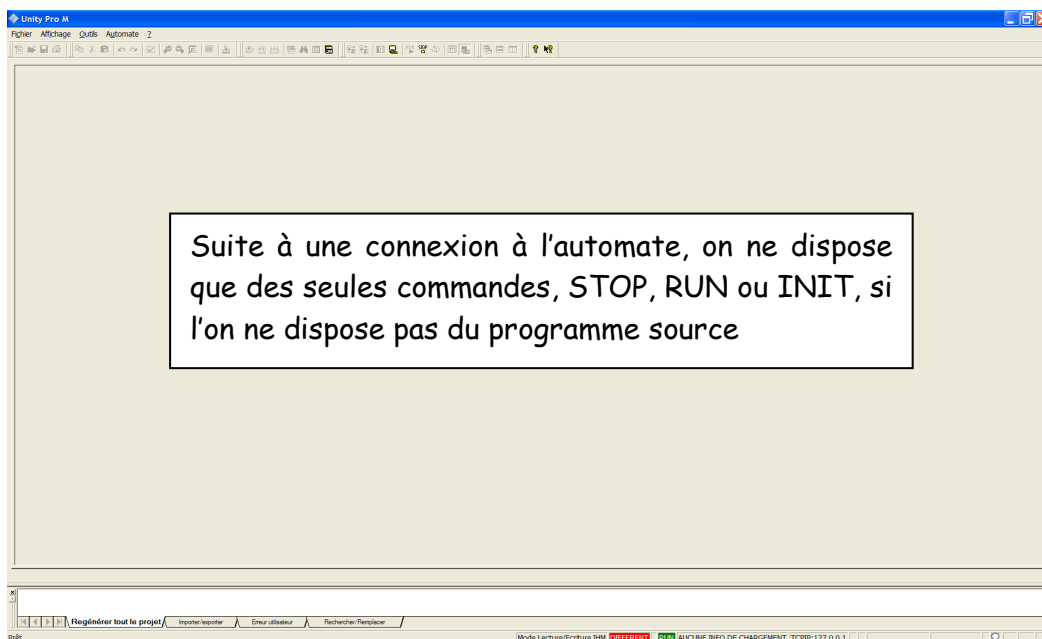
```
ld grafcet_secu (*chargement dans accu 1 etape GR7 secu*)
eg 1 (*comparaison si etape 1 active*)
and $MW5.0
andn $MW5.1
st def_contact0 (*defaut contact 1 voie 1 arret urgence*)

ld grafcet_secu (*chargement dans accu 1 etape GR7 secu*)
eg 1 (*comparaison si etape 1 active*)
andn $MW5.0
and $MW5.1
st def_contact1 (*defaut contact 1 voie 2 arret urgence*)

ld grafcet_secu (*chargement dans accu 1 etape GR7 secu*)
eg 1 (*comparaison si etape 1 active*)
andn $MW5.2 (*auto-contrôle voie 3*)
st def_autocontrôle

ld $MW8.6(*sortie relais 7 XPS*)
andn Kau
st def_cablage
```

**Remarque importante** : si lors du transfert du programme dans les options du projet la case à cocher **informations d'upload a été sélectionné sans (cf.p19)**, suite à une connexion à l'automate, si on ne dispose pas du programme source, on ne peut que passer l'automate en STOP, RUN ou déclencher une commande initialiser, la totalité des éditeurs de configuration et de programmation sont inaccessibles.



## Outils de mise au point du projet

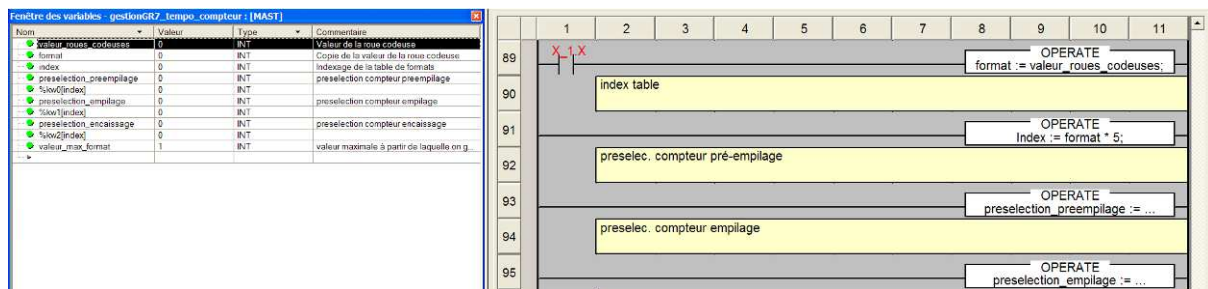
Unity met à votre disposition plusieurs éditeurs permettant de faciliter la mise au point du programme, ces éditeurs nécessite de travailler en connexion avec l'automate ou le simulateur.

- Fenêtre de variable

- Fenêtre d'inspection
- Tables d'animations
- Point d'arrêt pour stopper l'exécution du programme à un point bien précis

**Fenêtre de variables :**

La fenêtre de variable permet de visualiser toutes les variables non booléennes présentes à l'écran dans une section de programme. Pour la faire afficher sélectionner le menu Outils-Fenêtre de variables. Le défilement de la section de programme entraine la mise à jour et l'affichage des nouvelles variables présentes à l'écran.



Fenêtre de variable associée à sa section de programme

**Fenêtre d'inspection :**

Les fenêtres d'inspections sont placées par l'utilisateur sur la variable que l'on veut visualiser. Ces fenêtres sont particulièrement adaptées pour les langages IL et ST.

Sélectionner la variable puis clic droit-nouvelle fenêtre d'inspection. On peut effectuer un réglage de la fenêtre en se positionnant dessus puis clic droit-réglages (cf.ci contre).

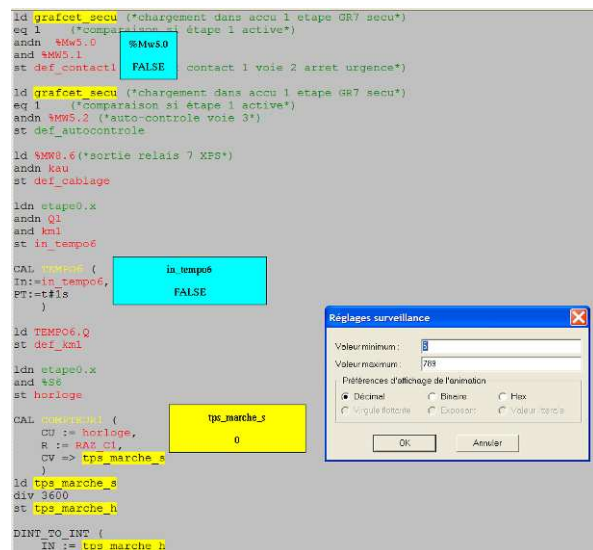
Si la valeur courante est comprise dans la plage définie, le champ de surveillance est affiché.

Si la valeur courante est inférieure à la plage définie, le champ de surveillance est affiché en jaune.

Si la valeur courante est supérieure à la plage définie, le champ de surveillance est affiché en magenta.

On peut masquer toutes les fenêtres via le menu Affichage-Masquer les fenêtres d'inspection.

Pour supprimer les fenêtres, placez le curseur sur le champ de surveillance à supprimer et exécutez la commande Supprimer l'inspection dans le menu contextuel.



**Remarque :** ces fenêtres peuvent être placées en mode local (non connecté). En langage ladder on ne peut placer des fenêtres d'inspection que pour des variables booléennes.

### Table d'animation :

Une table d'animation est utilisée pour surveiller les valeurs des variables et pour modifier et/ou forcer ces valeurs. Seules les variables déclarées dans Variables et instances FB peuvent être ajoutées à la table.

Il existe deux types de tables d'animation :

- Tables d'animations permanentes
- Tables d'animations temporaires

Une table d'animation permanente fait partie intégrante d'un projet. Elle est enregistrée avec le projet et peut être exportée.

Une table d'animation temporaire :

- n'est pas enregistrée avec un projet, mais supprimée à la fermeture du projet. Par conséquent, aucune table d'animation temporaire n'est disponible dans le navigateur de projet à l'ouverture d'un projet
- ne modifie pas un projet
- ne modifie pas l'état de génération d'un projet

La création d'une table d'animation s'effectue depuis le navigateur de projet par clic droit puis nouvelle table d'animation sur le répertoire table d'animation (cf.ci contre).

La création d'une table d'animation peut s'effectuer hors du navigateur, dans un éditeur de langage, en sélectionnant la variable puis par le menu contextuel-initialiser la table d'animation. Dans ces conditions la table est nécessairement temporaire.

En mode local, une table n'affiche pas les valeurs des variables, mais uniquement en mode connecté.

Permet d'associer la table à un module fonctionnel, s'il a été créé au préalable (cf.p 3)

Tables temporaires ou permanentes

Sélection des variables à intégrer

The image contains three screenshots from the Unity Pro software. The first screenshot shows the 'Navigateur du projet' (Project Navigator) with a right-click context menu open over the 'Table d'animation' folder. The second screenshot is a dialog box titled 'Nouvelle table d'animation' (New animation table) with fields for 'Nom' (Name), 'Module fonctionnel' (Functional module), and 'Commentaire' (Comment), along with a checkbox for 'Table temporaire' (Temporary table). The third screenshot shows the 'Table d'animation' editor window with a list of variables and their types for selection.

*Support à l'utilisation du logiciel Unity Pro*

La fonction chaînes étendues n'est utilisée que si l'on veut visualiser des variables de type STRING (chaîne de caractères) de plus de 16 caractères. Par défaut la table permet d'afficher 16 caractères d'une variable STRING. **Lorsque vous utilisez cette fonctionnalité, n'oubliez pas que l'animation de grandes chaînes peut réduire le nombre de sections et de tables d'animation susceptibles d'être animées simultanément.**

**Exemple** : soit une variable **chaines** à laquelle on a affecté l'attribut STRING[40], et stockant les caractères suivant : **aujourd'hui il fait beau demain**

Nom	Valeur	Type
chaines	'aujourd'hui il f'	string[40]

Visualisation variables sans activation de chaînes étendues, on visualise les 16 premiers caractères de la chaîne

Nom	Valeur	Type	Commentaire
chaines	'aujourd'hui il fait beau demain'	string[40]	

Visualisation variables avec activation de chaînes étendues, on visualise toute la chaîne, car on a sélectionné l'animation de 30 variables (cf. p précédente)

Les tables d'animations permettent de visualiser des variables mais surtout de pouvoir modifier ou forcer les valeurs en mode connecté.

Mode modification :

Ce mode permet de basculer les variables TOR et de définir la valeur pour les autres types. **Attention en mode modification le programme est prioritaire sur la valeur que vous définissez à la variable.**

Nom	Valeur	Type
chaines	'aujourd'hui il f'	string[40]
bibi	0	BOOL
essai	0	BOOL
titi	0	BOOL
toto	0	INT

Activer le mode modification, sélectionner la variable booléenne puis positionner la valeur à 0 ou à 1. Si aucun résultat ne se produit, c'est que le programme est prioritaire. Pour les variables non booléennes il suffit directement de saisir la valeur si le mode modification a été activé

Modification de plusieurs variables simultanément

Le but de d'une modification de plusieurs variables est de permettre à l'automate de prendre en compte ces changements sur un même tour de cycle.

Nom	Valeur	Définir la valeur	Type
chaines	'aujourd'hui il f'		string[40]
bibi	0	1	BOOL
essai	0	1	BOOL
titi	0	0	BOOL
toto	89	56	INT

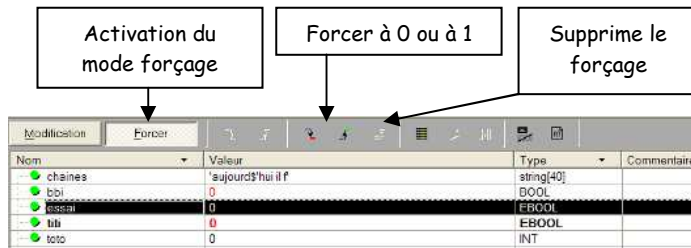
Activer le mode multiple, la colonne définir valeur apparait, sélectionner les valeurs des différentes variables, puis les activer par la commande activation valeur. La mise à jour s'effectue sur un cycle automate.

**Attention :** en mode modification le programme est prioritaire, par conséquent il est possible que la modification ne marche pas car le programme écrase les valeurs que l'on désirait modifier.

Mode forçage :

Ce mode est disponible uniquement pour les variables répondant aux conditions suivantes :

- la variable doit être de type EBool (entrées, sorties, mémentos)
- il doit s'agir d'une variable localisée (adressage topologique)

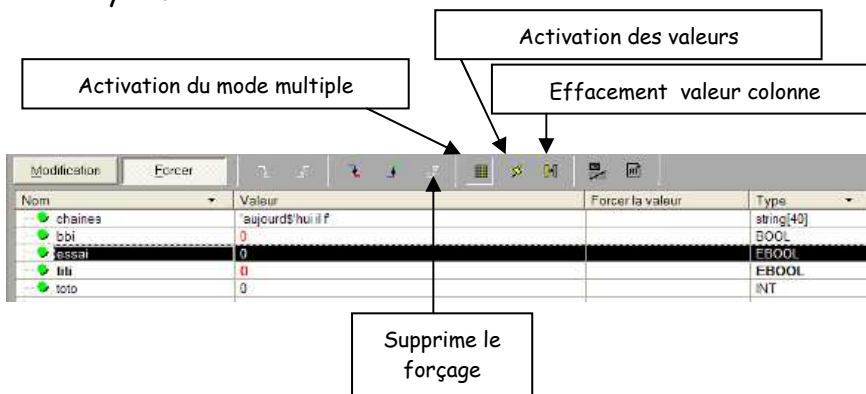


Activer le mode forçage, sélectionner la variable puis forcer la valeur à 0 ou à 1.

Si les boutons forcer à 0 ou à 1 sont grisés, c'est que les variables n'est pas de type EBOOL, ou de type EBOOL mais non localisée

Forçage de plusieurs variables simultanément

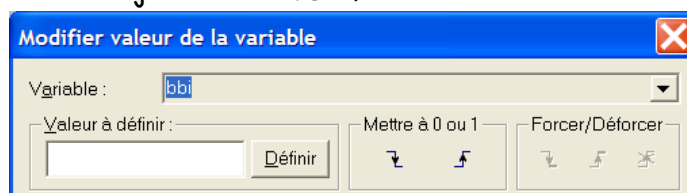
Le but d'un forçage groupé de plusieurs variables booléennes (localisées) est de permettre à l'automate de prendre en compte ces changements sur un même tour de cycle.



Activer le mode multiple, la colonne forcer la valeur apparait, sélectionner les valeurs des différentes variables (0 ou 1), puis les activer par la commande activation valeur. La mise à jour s'effectue sur un cycle automate.

**Remarque :** le forçage de plusieurs variables est possible en sélectionnant les variables avec la souris et appui sur la touche ↑ (minuscule) du clavier, puis en cliquant sur forcer à 0 ou à 1, le forçage n'est pas garanti sur le même cycle automate.

**Autres remarques :** les commandes modifications et forçage d'une variable sont directement accessible depuis l'éditeur de programmation en sélectionnant la variable, puis par le menu contextuel (clic droit de souris) modifier valeur. Toutes les conditions sont identiques à celle déjà énoncées. La fenêtre ci-dessous s'ouvre.



Tables de bits forcés :

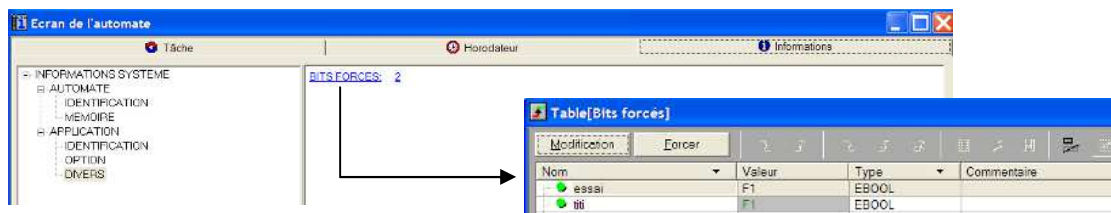
Vous pouvez créer une table d'animation incluant tous les bits forcés d'un projet.

Une table de bits forcés

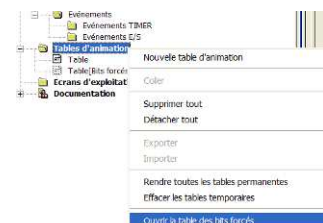
- ne peut pas être modifiée par l'utilisateur (la liste des variables)
- Est actualisée automatiquement si une variable ou une adresse devient forcée/déforcée
- Est toujours créée en tant que table d'animation temporaire

Il existe deux moyens de créer la table d'animation des bits forcés :

Choisissez **Outils** → **Ecran de l'automate** pour ouvrir la boîte de dialogue PLCScreen. Dans l'onglet Informations, sous **APPLICATION** → **DIVERS**, cliquez sur la ligne **BITS FORCES**.



Dans le navigateur de projet, cliquez avec le bouton droit sur Tables d'animation et sélectionnez Ouvrir la table des bits forcés dans le menu contextuel.



**Remarque :** il est conseillé en phase de mise au point de n'utiliser les tables d'animations que pour forcer ou modifier des valeurs de variables, pour effectuer de la visualisation, il est plus simple d'utiliser des fenêtres d'inspection ou des fenêtres de variables (cf. page 50-51).

Outils de mise au point et de réglages

Insertion d'un point d'arrêt :

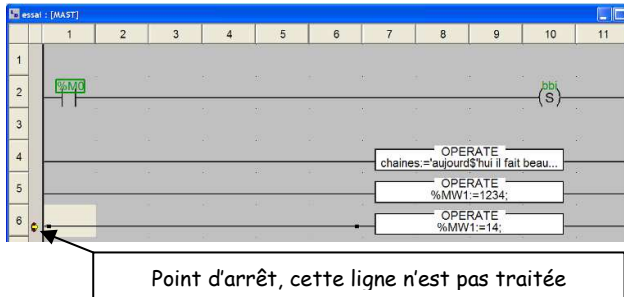
Un point d'arrêt permet de stopper l'exécution de la tâche à l'endroit où il a été posé. Il permet en phase de mise au point :

- d'examiner le comportement du code
- de visualiser la valeur des variables

Il y a un seul point d'arrêt à un instant donné dans le projet.

**Sélectionner le point d'arrêt soit :**

- en sélectionnant à partir du menu la commande Mise au point->Définir le point d'arrêt
- en sélectionnant à partir du menu contextuel la commande Définir le point d'arrêt
- en sélectionnant le bouton avec l'icône en forme de main de la barre d'outils mise au point (cette barre doit être au préalable affichée).



On peut voir dans la table d'animation que la dernière ligne n'est pas traitée, si tel avait été le cas la variable %MW1 aurait pris la valeur 14 qui aurait écrasée la valeur 1234.

A partir du menu, la commande **Mise au point->Afficher le point d'arrêt** vous permet de repérer le point d'arrêt en affichant la partie de l'éditeur langage où il est posé

Pour supprimer un point d'arrêt, sélectionnez à partir du menu la commande **Mise au point->Effacer le point d'arrêt** ou depuis l'icône en forme de main barrée de la barre d'outils mise au point (cette barre doit être au préalable affichée).

La suppression d'un point d'arrêt ne relance pas la tâche, pour cela vous devez sélectionner la commande **Aller à** du menu mise au point, ou depuis l'icône **Aller à** de la barre d'outils mise au point.

**Exécution d'un programme au pas par pas :**

Ce mode est lancé via un point d'arrêt défini au préalable. Il permet d'examiner le comportement du code et la valeur des variables.

Il est mis en œuvre en mode connecté, la section exécutée en mode pas à pas stoppe la tâche correspondante, et les liens ne sont plus animés.

Trois commandes permettent de faire du mode pas à pas, il s'agit :

- de la commande **Pas dans module appelé** : utilisé lors de l'appel de SR
- de la commande **Pas suivant** : utiliser pour se déplacer au réseau suivant
- de la commande **Retour module appelant** : utilisé pour sortir d'un SR

On peut voir la position des réseaux évalués en affichant l'étape courante via le menu **Mise au point->afficher l'étape courante**, une flèche jaune se positionne en face du réseau ladder en évaluation.



### Insertion d'un point de visualisation :

Sans point de visualisation les valeurs des variables animées sont affichées à la fin de traitement de la tâche MAST.

La limite de ce mode de fonctionnement est qu'il ne permet pas de connaître la valeur d'une variable en un point précis du programme si celle-ci est utilisée dans différentes sections.

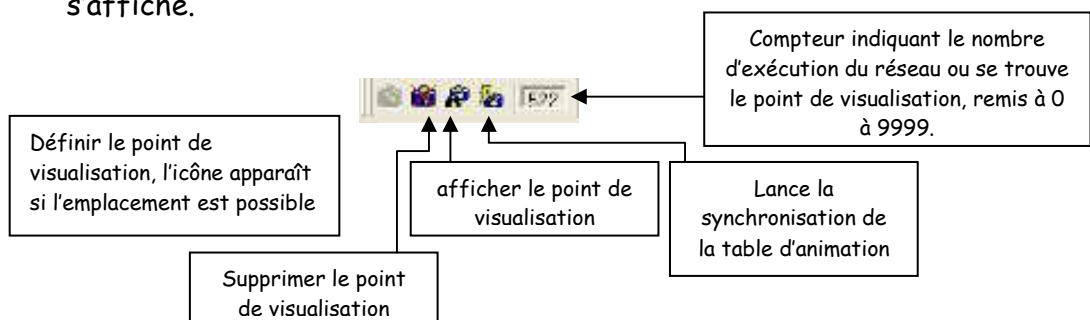
Le point de visualisation permet de synchroniser l'affichage des variables animées avec l'exécution d'un élément programme afin de connaître leur valeur à ce point précis du programme.

Les variables synchrones au point de visualisation appartiennent obligatoirement à la section dans laquelle le point de visualisation est posé, la visualisation des autres variables reste synchrone avec la fin de la tâche MAST.

- la pose du point de visualisation est possible en mode connecté uniquement
- un seul point de visualisation est autorisé à un moment donné
- un compteur est incrémenté chaque fois que le réseau connexe possédant le point de visualisation est exécuté
- la modification d'une section n'est pas autorisée si un point de visualisation est posé

Pour insérer un point de visualisation procédez :

- en sélectionnant à partir du menu la commande :  
**Mise au point->Définir le point de visualisation**, vous devez avoir au préalable sélectionné un des éléments du réseau sur lequel vous voulez placer ce point
- en sélectionnant à partir du menu contextuel la commande  
**Définir le point de visualisation**, vous devez avoir au préalable sélectionné un des éléments du réseau sur lequel vous voulez placer ce point
- en sélectionnant le premier bouton de la barre d'outils ci dessous.  
Cette barre doit être au préalable affichée, par la commande Outils->Personnaliser->choisir la case à cocher Point de surveillance, la barre s'affiche.



**Remarque :** toutes les commandes de la barre d'outils, sont disponibles depuis le menu Mise au point.

Point de visualisation

Section ladder lue en premier par l'automate

Section ladder lue par l'automate après la section ci-contre

Dans la table d'animation, la variable %MW1 est synchronisée par rapport au point de visualisation, et donne bien la valeur 14, alors que la variable %MW2 n'étant pas dans la même section ladder sa valeur est actualisée à la fin du traitement de la tâche MAST.

La synchronisation est obtenue après ouverture de la table d'animation puis via le bouton de la barre d'outils (cf. ci-dessus) ou par la commande Synchroniser la table d'animation du menu mise au point.

Nom	Valeur	Type	Commentaire
%MW1	14	INT	
%MW2	45	INT	

Table d'animation dans laquelle les variables ont été synchronisées

**Remarque :** la commande effacer le point d'arrêt, rend toutes les variables de la table d'animation synchrone avec la fin de la tâche MAST. La commande afficher le point de visualisation ouvre la section ladder et se positionne devant ce dernier.

## Diagnostic avec Unity

Le diagnostic sous Unity Pro est constitué d'outils et de fonctionnalités qui permettent d'agir à toutes les étapes d'un cycle de développement et d'utilisation d'une application d'automatisme.

Les différents moyens sont les suivants :

- **Les écrans de diagnostic :** les écrans de diagnostic sont disponibles à partir de l'éditeur de configuration pour le processeur et pour les modules métier. Pour accéder à l'écran de diagnostic désiré vous double cliquez sur l'élément (processeur ou module) puis vous sélectionnez l'onglet Défaut. Les informations de défaut sont données de manière très claire.
- **Le Viewer de diagnostic :** le Viewer de diagnostic est un outil qui permet de visualiser des alarmes générées par le diagnostic système et les DFB de diagnostic. Il existe un Viewer intégré à Unity Pro.

**Le diagnostic système :** le diagnostic système est un service qui associe les bits et mots système au Viewer de diagnostic. Lorsqu'un projet a été généré avec l'option Diagnostic système, le comportement des bits et mots système génère automatiquement des messages de diagnostic sur les Viewers de diagnostic.

**Le diagnostic système s'effectue de manière automatique.** Lorsque l'automate détecte une erreur système (par exemple, le dépassement du chien de garde, un défaut d'entrées/sorties, une division par zéro,...), une information est transmise au Viewer de diagnostic. **Le viewer de diagnostic affichera un message d'erreur système si dans les options du projet, vous avez coché la case [Diagnostic système](#) (commande options du projet du menu outils).**

- **Le diagnostic projet** : le diagnostic projet est constitué d'EFB et de DFB spécifiques qui sont intégrés dans le code d'un projet d'automatisme (**à intégrer dans le programme automate**) afin de donner à l'exploitant ou à l'agent de maintenance des informations claires sur les conditions de fonctionnement du process surveillé. **Ces éléments de programme vont déclencher des alarmes visualisables au travers des Viewers de diagnostic.**

Le diagnostic SFC est quant à lui analogue au diagnostic système, il est intrinsèque au SFC et permet de surveiller les temps d'activité des étapes. **Le diagnostic projet appelé aussi diagnostic application doit être activé dans les options du projet (commande options du projet du menu outils).**

Comme nous l'avons dit précédemment le diagnostic système s'effectue de manière automatique et chaque erreur est transmise au viewer de diagnostic (si l'option a été activée cf. page précédente). Le tableau ci-dessous récapitule les informations système surveillées de manière automatique par le service de diagnostic système.

Objet système	Brève description de l'alarme
%S10	Erreur d'E/S
%S11	Dépassement du chien de garde
%S15	Défaut chaîne de caractères
%S18	Dépassement ou erreur arithmétique
%S19	Dépassement période de tâche
%S20	Dépassement d'index
%S39	Saturation lors du traitement de l'événement
%S51	Retard de l'horodateur
%S65	Commande d'extraction de carte
%S66	Sauvegarde de l'application dans la carte mémoire
%S67	Etat de la pile de la carte mémoire PCMCIA contenant l'application
%S68	Etat pile processeur
%S76	Tampon de diagnostic configuré
%S77	Tampon de diagnostic plein
%S96	Programme de sauvegarde précédent
%S118	Défaut d'E/S Fipio général
%S119	Défaut d'E/S général sur le rack

**Support à l'utilisation du logiciel Unity Pro**

%SW0	Période de scrutation de la tâche MAST
%SW1	Période de scrutation de la tâche FAST
%SW2	Période de scrutation de la tâche AUX 0
%SW3	Période de scrutation de la tâche AUX 1
%SW4	Période de scrutation de la tâche AUX 2
%SW4	Période de scrutation de la tâche AUX 3
%SW11	Durée du chien de garde
%SW17	Statut de défaut pour opération flottante
%SW76	Fonction de diagnostic : enregistrer
%SW77	Fonction de diagnostic : annuler l'enregistrement
%SW78	Fonction de diagnostic : nombre d'erreurs
%SW96	Enregistrer/restituer %MW dans la mémoire flash
%SW97	Code d'erreur pour carte de stockage
%SW125	Type d'erreur bloquante
%SW146	Fonction d'arbitre sur bus Fipio
%SW153	Liste des défauts du gestionnaire de voie Fipio
%SW154	Liste des défauts du gestionnaire de voie Fipio

Pour le diagnostic projet, ce document ne traitera pas la méthode pour la création de bloc DFB de diagnostic, pour cela se rendre dans l'aide du logiciel sous l'onglet Sommaire puis suivre l'arborescence suivante :

**Logiciel Unity→Mode de marche→Diagnostic→Diagnostic projet**

Il est intéressant d'activer cette fonction via les options du projet car, à ce moment le diagnostic SFC (grafcet) est activé automatiquement et permet la surveillance des temps d'activités d'étapes de grafcet.

**Descriptif du Viewer de diagnostic**

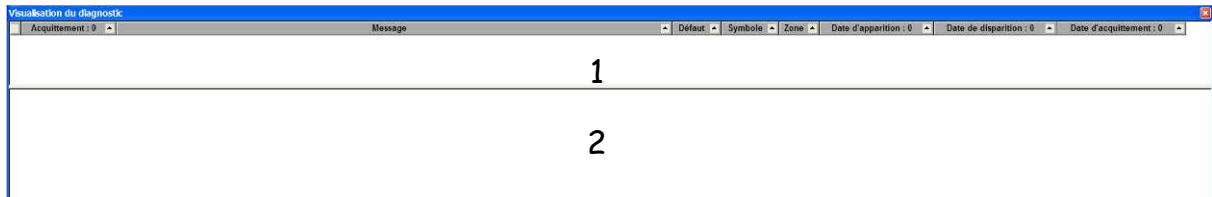
La fenêtre du Viewer de diagnostic comporte deux zones d'affichage :

- une zone dans laquelle s'affichent les messages d'erreur relatifs au diagnostic
- une zone qui affiche des informations supplémentaires concernant le message d'erreur sélectionné dans la liste des alarmes.

L'écran en page suivante présente le Viewer de diagnostic et ses deux zones d'affichage :

- 1 : Liste des messages d'erreur
- 2 : Informations supplémentaires concernant le message d'erreur

sélectionné



Le Viewer de diagnostics utilise les icônes suivantes pour indiquer l'état du message :

Icône	Description
✓	Le défaut a disparu et a été acquitté (si un acquittement était nécessaire).
✗	Le défaut requiert un acquittement.
➡	Le défaut n'a pas disparu.
⚡	Le défaut système a disparu et a été acquitté (si un acquittement était nécessaire).
⚡	Le défaut système n'a pas disparu.

### Exemple d'informations présentes dans le Viewer de diagnostic



La zone d'informations supplémentaires fournit les données suivantes pour le message sélectionné dans la liste :

- **type d'alarme**
- **variables spécifiques en fonction du type d'alarme**
- **liste des variables concernées par l'erreur, avec les commentaires associés**
- **interprétation des mots d'état**

Pour permettre l'affichage l'affichage du Viewer de diagnostic, il suffit de sélectionner le menu : **Outils**→**Viewer de diagnostic**.  
Bien entendu cette fenêtre n'est intéressante qu'en mode connecté

Le Viewer de diagnostic permet de gérer les messages d'erreur. Vous pouvez :

- ❖ trier la liste des messages
- ❖ naviguer dans la liste des messages
- ❖ acquitter un message de la liste
- ❖ supprimer un message de la liste
- ❖ supprimer une alarme de la mémoire de l'automate

### Acquittement des défauts :

Pour acquitter un message qui le nécessite, vous devez le sélectionner et activer l'une des commandes suivantes :

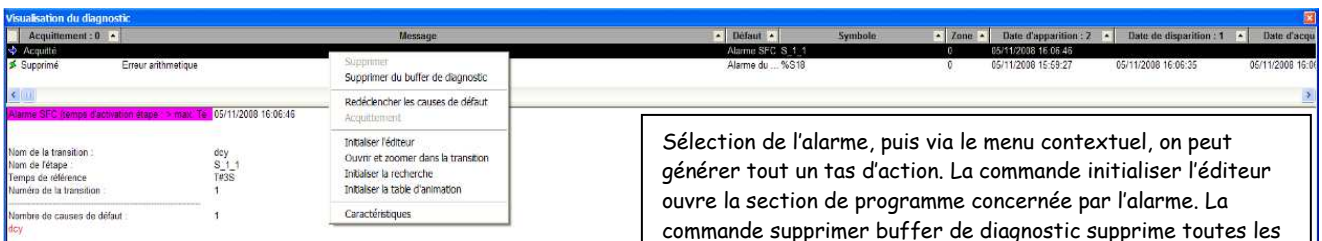
- la rubrique correspondante dans le menu contextuel (accessible par un clic droit de la souris)
- la touche fonction F6
- dans le menu Service puis acquittement ou dans le bouton de la barre d'outils Diagnostic Viewer ci-celle-ci est affichée.

Avec le Viewer de diagnostic, on peut effectuer une recherche des causes en recherchant l'élément ayant déclenché une alarme. Pour cela il est nécessaire que le projet soit généré avec les options concernées (cf. p19 et 20). Il faut sélectionner la case diagnostic application et choisir diagnostic local ou global.

Alors une alarme remontant dans le Viewer sera complétée de tout un tas de renseignements possibles sur la cause l'ayant déclenché.

Vous ne pouvez pas supprimer un message qui nécessite un acquittement ou dont le défaut associé n'a pas disparu. Par contre, vous pouvez supprimer les messages associés à des défauts qui ont disparu et ont été acquittés (s'ils nécessitaient un acquittement). Pour cela activez:

- la rubrique correspondante dans le menu contextuel (cf. page suivante)
- la touche Suppr



Projet ayant été validé avec les options énoncées en page précédente. On voit que la cause ayant déclenché l'alarme est la variable dcy, on peut en la sélectionnant puis clic droit lancer une recherche ou ouvrir une table d'animation. La recherche permet de localiser cette variable dans tous les éditeurs.

Sélection de l'alarme, puis via le menu contextuel, on peut générer tout un tas d'action. La commande initialiser l'éditeur ouvre la section de programme concernée par l'alarme. La commande supprimer buffer de diagnostic supprime toutes les alarmes présentes, on peut ensuite les supprimer dans l'éditeur.

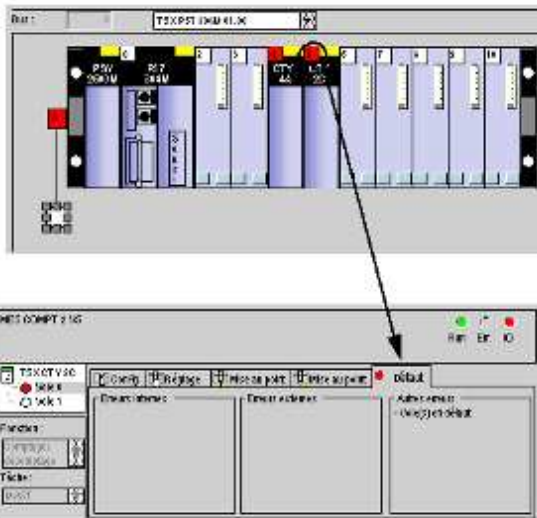
### Diagnostic via les écrans de configuration

Les écrans de diagnostic de niveau module ou de niveau voie ne sont accessibles qu'en mode connecté. Lorsqu'un défaut non masqué apparaît, celui-ci est signalé :

- dans l'écran de configuration du rack, par un carré rouge à la position du module de comptage en défaut.

### Support à l'utilisation du logiciel Unity Pro

- dans tous les écrans de niveau voie (onglets Configuration, Réglage, Mise au point et Défaut)



Sur l'exemple ci-contre, on voit que le rack 0 est en défaut et que sur ce rack, les cartes 4 et 5 le sont. En double cliquant sur ces cartes on ouvre l'éditeur qui permet de connaître un certain nombre d'éléments sur la nature du défaut.

Bien évidemment pour avoir accès à ces éditeurs, il faut se rendre dans la configuration matérielle (cf. p 22).

### Outils de simulation

Le simulateur est installé automatiquement avec Unity Pro. Le simulateur permet de simuler une UC d'automate.

Les points d'arrêt, les procédures pas à pas et la fonction Atteindre vous permettent de tester votre programme utilisateur dans l'automate simulé. Le simulateur d'automate simule un projet complet avec des tâches utilisateur.

Le simulateur d'automate ne prend aucune forme d'E/S en charge. Bien que la simulation contienne des composants de projet pour des E/S, celles-ci ne sont pas traitées par le simulateur d'automate. **Vous ne pouvez accéder aux entrées et sorties qu'à partir du projet ou via les fonctions en ligne du logiciel Unity Pro (lire, écrire, forcer, animer, etc.)**

Le simulateur d'automate ne permet pas de déclencher des événements d'E/S en réglant/forçant les bits %I.

Comme pour un automate réel, avant la connexion vérifier que votre projet soit généré et qu'il ne contienne pas d'erreurs.

Basculer alors en mode simulation via l'icône de la barre d'outils, cf. p47 (il faut que la barre API soit sélectionnée, commande **Personnaliser** du menu **Outils**), ou via la commande **Mode Simulation** du menu **Automate**.

Les commandes sont ensuite identique à celle que l'on effectuerait sur un automate réel.

A l'issu de la connexion au simulateur, un icône apparaît dans la barre d'outils de Windows (à gauche de l'horloge).

Cette icône identifie le simulateur actif et affiche les différents états de l'automate simulé.



Icône simulateur

Exemple	Couleur	Description
	vert	Mode de fonctionnement normal
	jaune	L'automate est à l'état PAUSE.
	rouge	L'automate est à l'état ERREUR.

Un cadre autour de l'icône indique le mode de mise au point actif

Exemple	Couleur	Description
	bleu	Le mode de mise au point est actif. C'est-à-dire qu'au moins un point d'arrêt est défini dans le projet ou qu'il existe au moins une tâche volontaire en mode de mise au point.

Le symbole central indique l'état actuel de l'automate tel que NOCONF, REPOS, ARRETE, RUN.

Exemple	L'état de l'automate simulé est...	Description
	NOCONF (aucune configuration)	Aucun projet utilisateur n'est chargé ou le projet chargé n'est pas valide ou supprimé à l'aide de la commande ERACER.
	REPOS	Le projet chargé sur l'automate n'est pas démarré ou n'a pas été réinitialisé à l'aide du bouton de commande RAZ.
	ARRETE	Aucun projet en cours d'exécution.
	RUN	Un projet comportant au moins une tâche est en cours d'exécution.

La couleur du symbole central indique l'état de la connexion :

Exemple	Couleur	Description
	noir	Aucun client TCP/IP n'est connecté.
	rouge	Au moins un client TCP/IP est connecté.

#### Etats d'erreur

Signification des symboles :

Exemple	L'état de l'automate simulé est...	Description
	PAUSE	Présence d'une erreur dans le projet. L'automate simulé doit être réinitialisé à l'aide du bouton de commande RAZ.
	ERREUR	Présence d'une erreur fatale dans le projet. Cela signifie que la communication n'est plus possible. L'automate simulé doit être réinitialisé à l'aide du bouton de commande RAZ.

#### Etats internes

Les symboles suivants représentent les états internes temporaires qui ne doivent normalement pas être vus. Il n'est pas possible de récupérer de ces états. Le simulateur de l'automate doit donc être fermé et redémarré.

Exemple	L'état de l'automate simulé est...	Description
	POWER OFF	Une erreur interne s'est produite lors de la simulation d'une réinitialisation ou d'un redémarrage de l'automate.
	INIT	Une erreur interne s'est produite lors de l'initialisation du simulateur de l'automate.
	UNKNOWN	Le simulateur de l'automate est passé à un état inconnu.

Un double clic sur l'icône de la barre d'outils de Windows ouvre la fenêtre ci-dessous, cette fenêtre dépend du type d'automate (M340, Premium, Quantum, Atrium).



Le bouton RAZ, permet la simulation d'un démarrage à froid de la CPU, c'est-à-dire que la connexion entre Unity Pro et le simulateur est interrompue, que les variables du projet sont réinitialisées et que le simulateur passe à l'état

RUN (si le démarrage automatique est activé) ou STOP (si le démarrage automatique est désactivé).

Ce bouton de commande correspond au bouton de réinitialisation sur une UC réelle.

Le bouton cycle d'alimentation ou redémarrage permet d'exécuter un redémarrage (mise hors tension/mise sous tension) de l'automate simulé (et du simulateur). Il correspond à un démarrage à chaud de l'automate (la connexion entre Unity Pro et le simulateur est interrompue et toutes les variables du projet en cours restent inchangées).

Un redémarrage correspond au bouton de réinitialisation d'une alimentation Premium ou revient à débrancher, puis rebrancher une alimentation.



Toutes ces commandes sont disponibles via le menu contextuel, on trouvera alors également la commande effacer permettra de supprimer le projet de la mémoire du simulateur. La commande synchronisation permet de voir la charge de la CPU de l'ordinateur pour le traitement (pour plus de renseignements se référer à l'aide du logiciel).

**Remarque importante** : pour que la connexion soit possible, il faut que l'adresse IP du simulateur soit de 127.0.0.1 (cf.p 47). Le simulateur permet également une connexion au runtime de Vijeodesigner (logiciel pour la programmation d'écran), on peut donc simuler une application sans l'automate et sans l'écran. **Il suffit de déclarer dans Vijeodesigner un driver ModbusTCPIP et un équipement Modbus portant l'adresse IP : 127.0.0.1. Le runtime de Vijeodesigner interagit directement alors avec le simulateur de Unity. Pour simuler les entrées-sorties, il faut passer par des tables d'animation et forcer les variables, ou bien par des écrans d'exploitation (cf.p 68).**

### **Ecran d'exploitation**

Les écrans d'exploitation intégrés sont destinés à faciliter l'exploitation d'un procédé automatisé. Ils s'utilisent dans le logiciel Unity Pro.

L'éditeur graphique qui permet de créer ou modifier les écrans. En mode connecté, il permet également de visualiser les écrans animés et de conduire le procédé.

La bibliothèque d'objets qui présente des objets constructeur et permet de les insérer dans les écrans. Elle permet aussi de créer ses propres objets et de les insérer dans une famille de la bibliothèque.

Pour un projet donné, vous pouvez créer des écrans d'exploitation, en utilisant l'éditeur graphique.

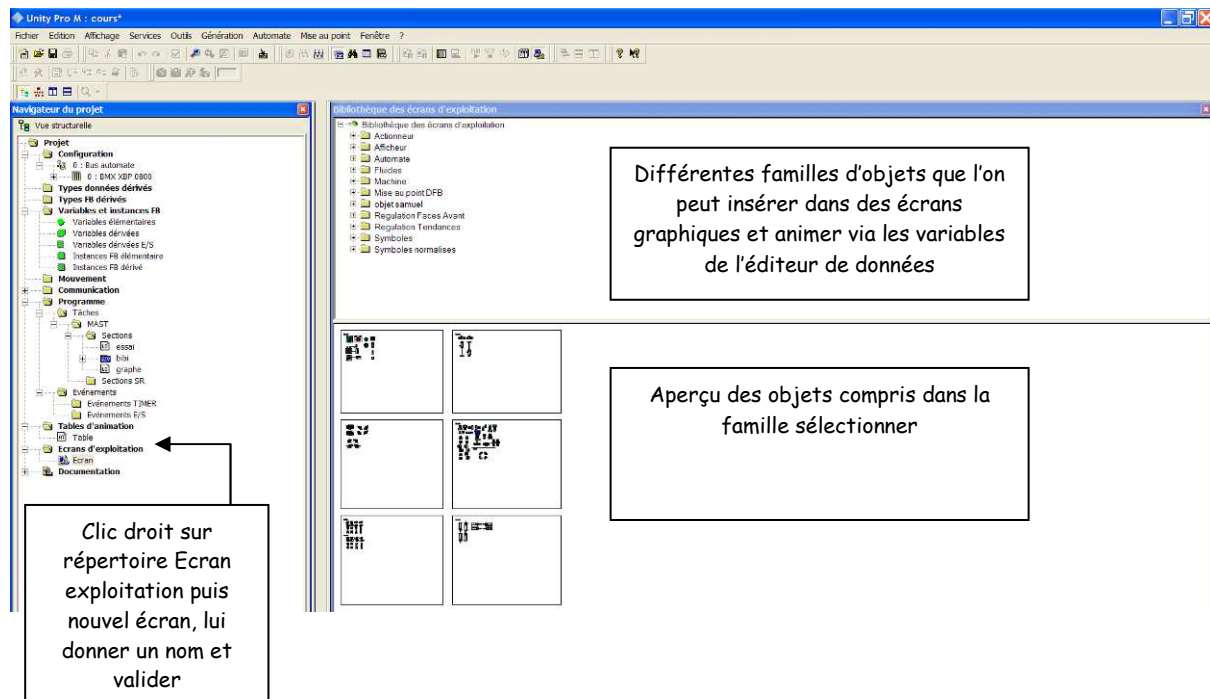
Ces écrans sont réalisés au moyen de textes et d'objets graphiques que vous pouvez dessiner (lignes, rectangles, courbes,...) ou récupérer dans la bibliothèque des objets graphiques. Ils sont constitués de parties statiques (fond de l'écran, titre,...) et de parties dynamiques ou animées qui permettent de refléter l'état du procédé.

Pour animer les objets dynamiques, vous devez leur affecter une variable dont la valeur déterminera l'affichage.

Pour conduire le procédé vous pouvez également insérer dans vos écrans des objets de pilotage (boutons, zones de saisie,...).

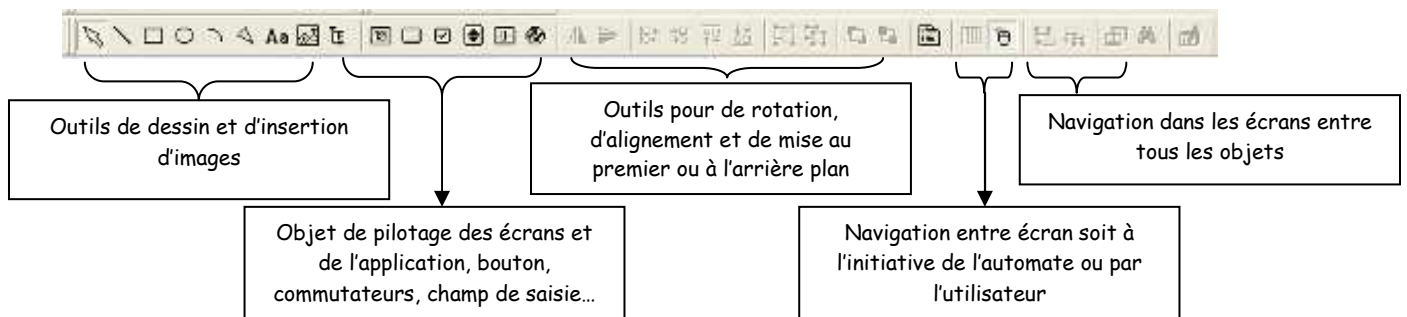
La bibliothèque d'objets présente les objets constructeurs et permet de les insérer dans les écrans d'exploitation. Les objets sont classés dans des familles. La bibliothèque permet aussi de créer ses propres objets en les insérant dans une famille de la bibliothèque.

La bibliothèque s'ouvre à partir de la commande **Outils → Bibliothèque des écrans d'exploitation**.



La suppression d'un écran, ou son changement de nom peut s'effectuer directement en le sélectionnant, puis en choisissant la commande en conséquence dans le menu contextuel. Le déplacement d'un écran peut s'effectuer par la fonction glisser-déposer avec la souris. On peut également créer une arborescence par création d'une famille (répertoire) à l'intérieur duquel seront rangés les écrans.

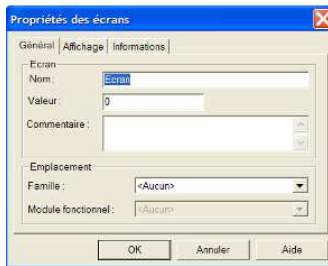
Après la création et l'ouverture d'un écran d'exploitation, la barre d'outils liée à cet éditeur apparaît, si la case **IOEDITOR** a été cochée dans la commande **Personnaliser du menu Outils**. La barre d'outils permet la création, la modification et l'édition des écrans d'exploitation.



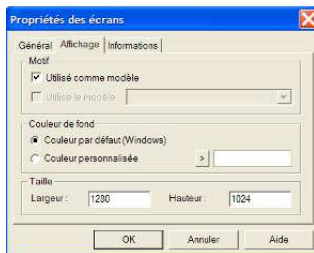
Le passage de la souris sur ces différents éléments fait apparaître la fonction de chacun des éléments.

**Remarque :** beaucoup de fonctions sont disponibles par le menu contextuel, tant sur les écrans que sur les objets de la bibliothèque.

Après un clic droit puis propriétés de l'écran sur l'écran créer dans le navigateur d'application, la fenêtre suivante avec différents onglets apparaît.



Choisir le nom de l'écran. La valeur correspond au numéro d'identification de l'écran, vous pouvez modifier ce numéro qui est utilisé lorsque l'on veut associer un bouton de [navigation à l'écran](#). Ce numéro peut être utilisé lorsque [l'automate](#) contrôle l'affichage des écrans en mode connecté, pour cela il faut avoir sélectionné une variable de l'éditeur de données dans l'onglet écran d'exploitation de la boîte options du projet (accessible via la commande Options du projet du menu Outils). On peut associer un écran à une famille que vous aurez au préalable créée.



Lorsque la case utilisée modèle est cochée, cet écran peut être utilisé comme modèle pour tout autre écran du projet. Un écran modèle ne peut pas contenir d'objets animés. Lorsque la case utilise le modèle est coché, le fond de cet écran utilise le fond d'écran modèle que vous avez défini. Le bouton de droite permet de sélectionner l'écran modèle dans une liste déroulante. On peut modifier les couleurs de fond de l'écran et sa taille

### **Présentation des objets d'un écran graphique :**

Les objets qui peuvent être créés dans un écran graphique sont de 4 types :

- les objets standards : ligne, rectangle, ellipse, courbe, polyligne, texte
- les images : fichiers bitmap avec l'extension BMP ou JPG
- les objets de pilotage (ou de commande) : bouton, case à cocher, champ de saisie, compteur, curseur, objet d'échange explicite, bouton de navigation écran
- les objets composés : ensemble d'objets des 3 types précédents, créé par l'utilisateur ou en provenance de la bibliothèque d'objets

Tous ces objets, utilisés pour créer un écran, peuvent être statiques ou animés.

Il existe des objets statiques et des objets dynamiques (animés)

**Les objets statiques n'ont pas de variable associée. Leur représentation graphique est fixe.**

**Les objets dynamiques ont une variable associée qui permet de modifier leur affichage.**

Les objets de pilotage (ou de commande) sont des objets qui permettent d'exécuter une action :

- naviguer d'un écran à un autre
- modifier la valeur d'une variable
- envoyer une commande vers un module métier de l'automate

Ces objets sont de 7 types : les boutons, les cases à cocher, les zones de saisie, les compteurs, les curseurs, les boutons de navigation d'écrans, les objets d'échanges explicites.

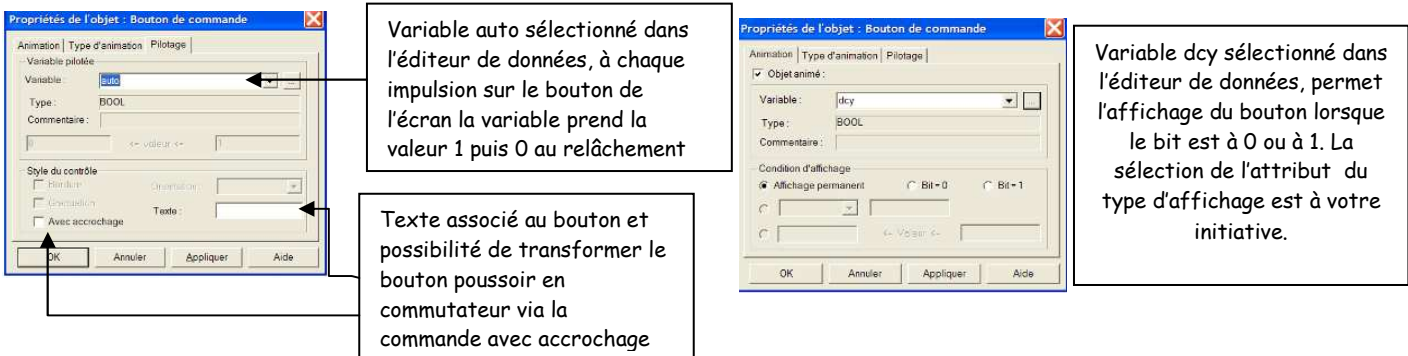
**Pour créer un objet de pilotage, sélectionner dans la palette d'outils le type d'objets à insérer, pointer dans l'écran avec le bouton gauche et agrandir l'objet,**

## Support à l'utilisation du logiciel Unity Pro

quand la taille correspondant est bonne, relâcher la souris, l'objet est alors inséré dans l'écran, on peut par la suite le déplacer et le redimensionner. Sélectionner l'objet avec la souris puis activer la commande caractéristique du menu contextuel de l'objet. Résultat : une boîte de dialogue contextuelle apparaît. Cette boîte dépend du type d'objet et permet d'en fixer les attributs

Les objets de pilotage sont activés par une action de la souris (ou du clavier). En fonction de l'attribut qui a été fixé, ces objets agissent sur leurs variables associées.

### Exemple des attributs d'un bouton de commande :



Variable auto sélectionné dans l'éditeur de données, à chaque impulsion sur le bouton de l'écran la variable prend la valeur 1 puis 0 au relâchement

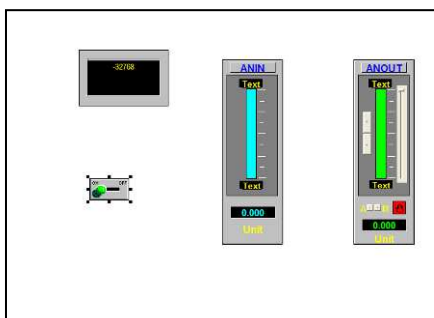
Texte associé au bouton et possibilité de transformer le bouton poussoir en commutateur via la commande avec accrochage

Variable dcy sélectionné dans l'éditeur de données, permet l'affichage du bouton lorsque le bit est à 0 ou à 1. La sélection de l'attribut du type d'affichage est à votre initiative.

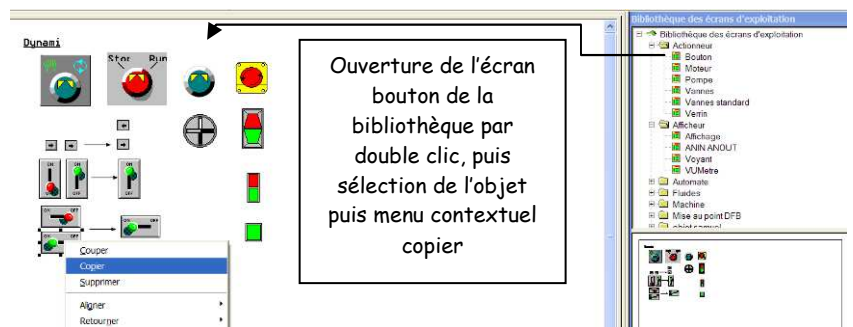
**Remarque :** tous les objets de pilotage sont directement disponibles, depuis le menu contextuel (clic droit puis Nouveau→Contrôles), à condition qu'un écran soit déjà ouvert et que le clic droit intervienne dans cet écran.

Pour connaître les différents attributs de chacun des objets, se rendre dans l'aide du logiciel.

### Manipulation des objets de la bibliothèque



Se rendre dans l'écran d'exploitation que vous avez au préalable créé, puis copier à l'aide du menu contextuel de l'écran, l'objet de la bibliothèque est alors copié. Ci-dessus un écran d'exploitation dans lequel on a copié 4 objets de la bibliothèque

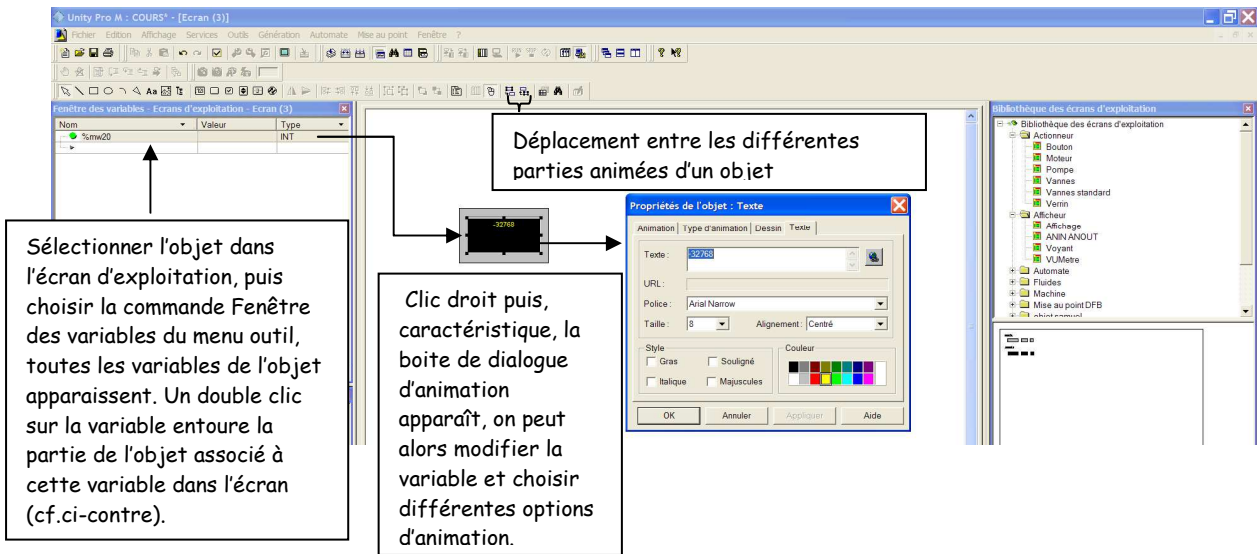


Ouverture de l'écran bouton de la bibliothèque par double clic, puis sélection de l'objet puis menu contextuel copier

La bibliothèque s'ouvre à partir de la commande **Outils** → **Bibliothèque des écrans d'exploitation**.

Les objets graphiques sont animés avec des variables à adresses topologiques. Il vous suffit de remplacer ces variables par celles de votre projet pour assurer l'animation des objets.

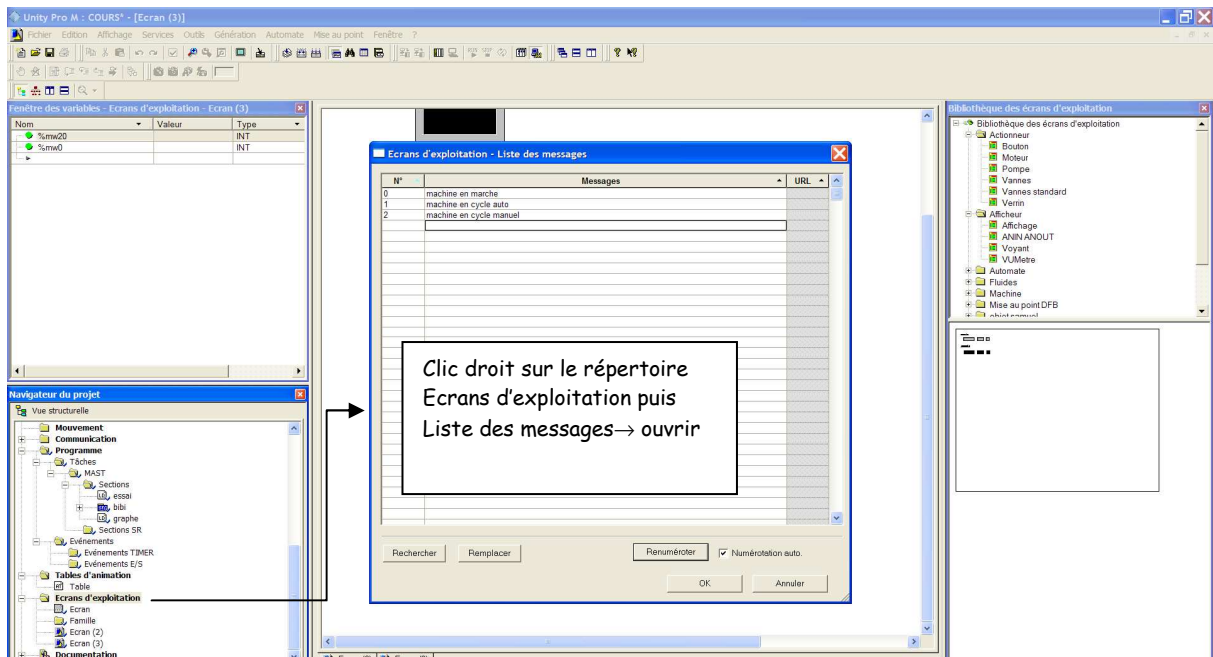
Pour associer une variable à un objet :



**Remarque :** si le composant présente plusieurs variables, il suffit de recommencer l'opération plusieurs fois, vous n'êtes pas obligé d'animer toutes les variables. On peut également se déplacer entre les différents objets directement par les boutons indiqués ci-dessus, les parties concernées de l'objet apparaissent en pointillé, on peut alors via le menu contextuel les animer.

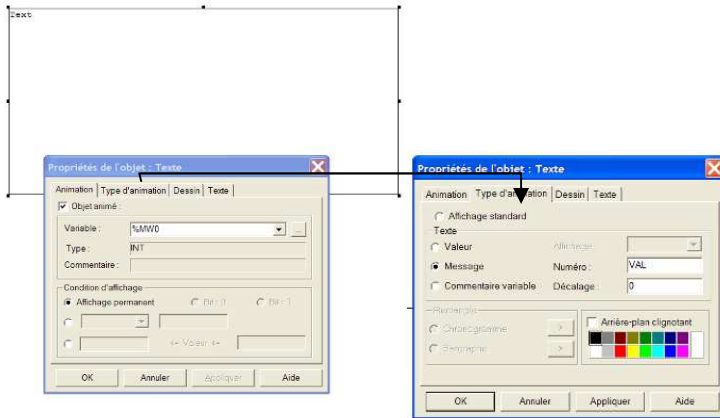
Affichage de messages :

Les messages sont composés d'un texte et d'un libellé. Ils sont mémorisés dans le projet et sont associés aux objets par leur numéro. Ainsi, il n'est pas nécessaire de créer un message pour chaque objet et dupliquer des messages identiques. Lors de la création d'un projet, la liste des messages est vide et c'est à vous de la remplir: en créant vos propres messages.



Ces messages seront ensuite affichés dans une zone de texte que l'on configurera en conséquence dans l'écran d'exploitation.

Une zone de texte a été placée dans l'écran d'exploitation sur laquelle on a affiché le menu contextuel caractéristique, une boîte de dialogue s'ouvre



Si la case valeur est activée la valeur prise par la variable s'affiche selon le mode d'affichage choisi.

Vous devez sélectionner la case message pour afficher un message configuré dans l'[éditeur de messages d'écrans d'exploitation](#). Le champ Numéro vous permet de saisir soit un numéro de message, soit le terme Val. Dans ce dernier cas, le message affiché sera celui dont la valeur est contenue dans la variable associée à l'objet. Le champ Décalage permet de saisir une valeur qui s'ajoutera à celle contenue dans la variable.

Si la case commentaire est sélectionnée et si la condition d'animation du texte est réalisée, le commentaire associé à la variable est affiché.

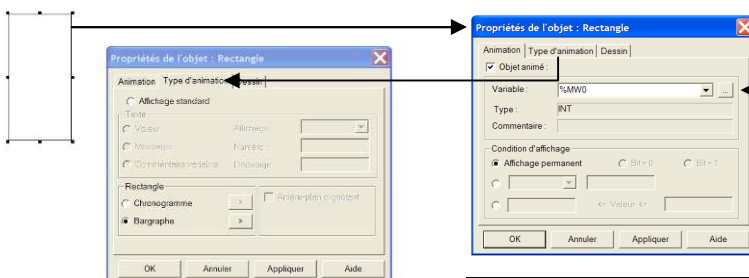
Choix de la variable à associer, que l'on va sélectionner dans l'éditeur de données. On peut en fonction du format de la variable configurer son type d'affichage. Il est en général intéressant de déclarer une variable de type INT, DINT, UINT ou UDINT pour l'animation, ce qui permet de pouvoir offrir les possibilités d'animation explicitées ci-contre.

**Remarque :** dans le cas de la configuration ci-dessus et en fonction de la liste de message déclarée en page précédente, si la variable %MWO prend la valeur 1, le message affiché dans la zone de texte sera : **machine en cycle auto**.

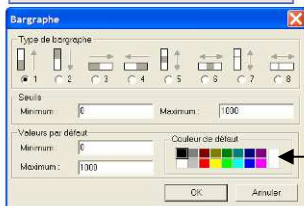
**Objet bargraphe :**

Un bargraphe permet de représenter graphiquement, sous forme de rectangles, l'évolution d'une variable.

Seuls les objets de type rectangle peuvent être des bargraphes



Sélection de la variable depuis l'éditeur de données



Accès à la palette de couleurs étendues

Dans cet exemple le rectangle se remplira de bas en haut suivant la valeur de la variable %MWO, si cette valeur est de 1000, le rectangle sera plein et vide si la valeur est 0. On peut configurer des valeurs à partir desquelles la couleur sélectionnée s'affiche, dans notre cas la couleur devient noir si la variable %MWO est supérieur à 1000 et inférieur à 0

**Chronogramme**

Un chronogramme permet de représenter graphiquement l'évolution d'une variable ; la courbe représentative évoluant de la droite vers la gauche.

Seuls les objets de type rectangle peuvent être des chronogrammes. Pour activer le mode chronogramme, placer un rectangle dans un écran d'exploitation puis via le menu contextuel → caractéristique, effectuer les différents réglages (cf.ci dessous).

The image shows two screenshots of the software's configuration interface. The top window is titled 'Propriétés de l'objet : Rectangle' and has the 'Animation' tab selected. It shows 'Objet animé' checked, 'Variable' set to '%MMW0', and 'Type' as 'INT'. The bottom window is also titled 'Propriétés de l'objet : Rectangle' but has the 'Chronogramme' sub-tab selected. It shows 'Affichage' set to 'Chronogramme', 'Echantillonnage (secondes)' set to 1, 'Définition (pixels)' set to 2, and 'Couleur du bacc' set to a color. A 'Palette de couleurs étendues' is also visible.

Sélection de la variable depuis l'éditeur de données

Accès à la palette de couleurs étendues

L'échantillonnage correspond à la période de rafraîchissement de l'écran (entre 1 et 999 secondes). Par défaut, cette valeur est 1 seconde.  
Les seuils correspondent aux valeurs entre lesquels la représentation de la valeur évolue. La durée de visualisation dépend de la taille du rectangle, de la valeur d'échantillonnage et de la définition.

Il existe d'autres types d'animations, ce document à juste présenter les plus intéressantes.

**Remarque** : lorsque l'on est en mode connecté et que le pilotage de l'écran se fait par l'utilisateur, via des bouton de commande, des champs de saisie..., pour que l'écriture des variables associées aux objets puissent se faire, il faut activer l'icône **valider variable de la barre d'outils**.



**Exemple d'un écran d'exploitation**

The diagram shows a control screen layout titled 'ECRAN DE PILOTAGE ALTIVAR'. It includes several key components:
 

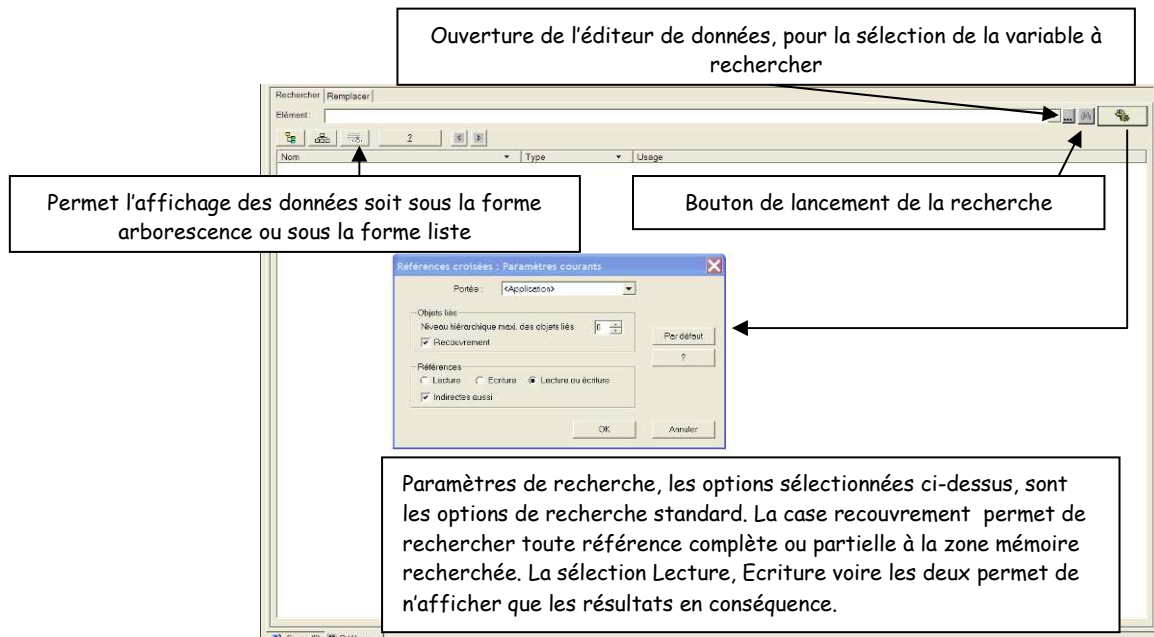
- Pupitre de pilotage**: A central panel with buttons for 'Acquit default', 'Bp dcy', 'Marche avant', 'Marche arrière', and 'Arrêt'.
- Boutons de pilotage**: A group of navigation buttons including 'Variateur en défaut', 'Thesys', and 'altivar\_31\_adressa2'.
- Affichage de messages**: A green box displaying a 'Message de conduite de l'installation'.
- Chap de saisie de vitesse**: A text input field for speed, with a label 'Vitesse en tr/mn - 0 à 1500 tr/mn'.
- Chronogramme de la consigne variateur**: A graph showing 'Evolution consigne reseau modbus' with axes for '1600 tr/mn' and '0 tr/mn' over 'temps'.

## Fonction Rechercher-Remplacer

Il peut être très utile pour effectuer du dépannage de la mise au point de programme de connaître à quel endroit est piloté une variable dans le programme, c'est ce que l'on appelle les références croisées. De plus, il peut être également intéressant de pouvoir remplacer une variable par une autre dans toutes les sections de programme, sans avoir à se déplacer manuellement dans chacune.

Pour accéder à la recherche d'une donnée, exécutez les actions suivantes :

- Dans le menu Outils, activez la commande Rechercher/Remplacer
- Sélectionnez l'onglet Rechercher
- Saisir la donnée à rechercher, en accédant à l'éditeur de donnée
- Configurer les options de recherche
- Lancer la recherche



Nom	Type	Usage
%MW1	INT	
Table	INT	
%MW1	<LD>	R/W
%MW1 (I.5, c.7)		W
%MW1 (I.7, c.7)		W
%MW1 (I.6, c.7)		W
gspche	<LD>	
%MW1 (I.2, c.8)		W

On peut voir l'utilisation de la variable %MW1 dans les différents endroits du programme. L'indication R indique que la variable est en lecture et W que la variable est

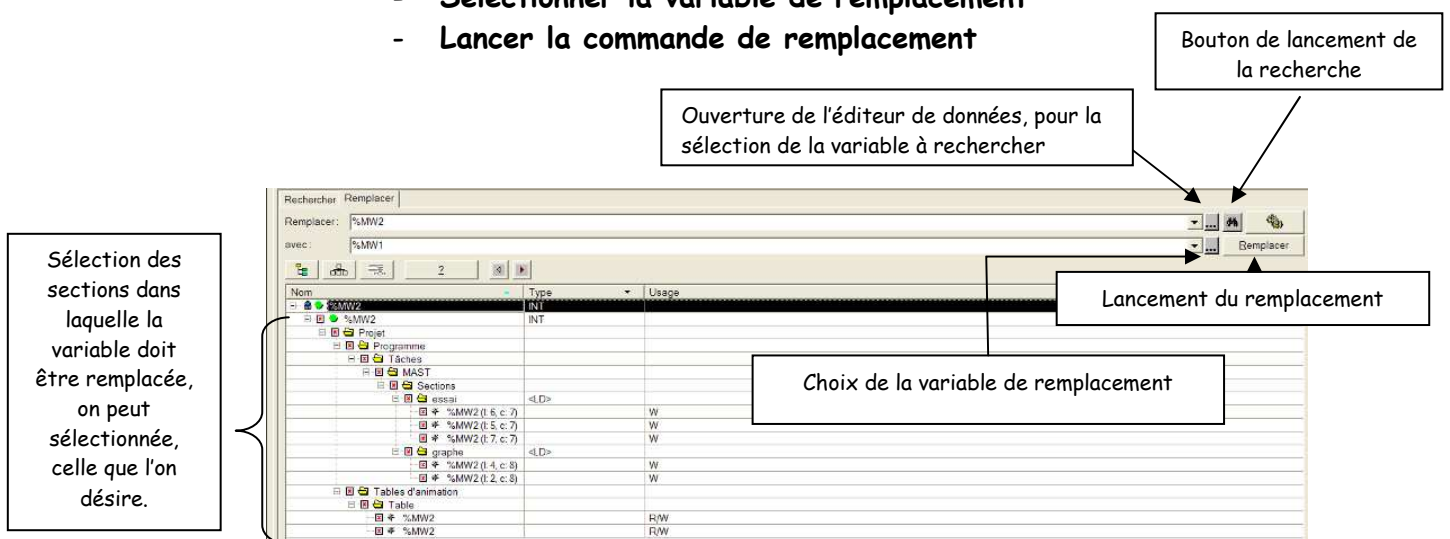


en écriture. Un double clic sur la ligne ouvre l'éditeur en conséquence et se positionne sur la variable.

**Remarque** : on peut directement avoir accès à la fonction Rechercher depuis un éditeur de programme, en sélectionnant la variable avec la souris puis via le menu contextuel Initialiser la recherche.

Pour accéder au remplacement d'une donnée, exécutez les actions suivantes :

- Dans le menu Outils, activez la commande Rechercher/ Remplacer
- Sélectionnez l'onglet Remplacer
- Saisir la donnée à rechercher, en accédant à l'éditeur de donnée, lancer la recherche
- Sélectionner la variable de remplacement
- Lancer la commande de remplacement

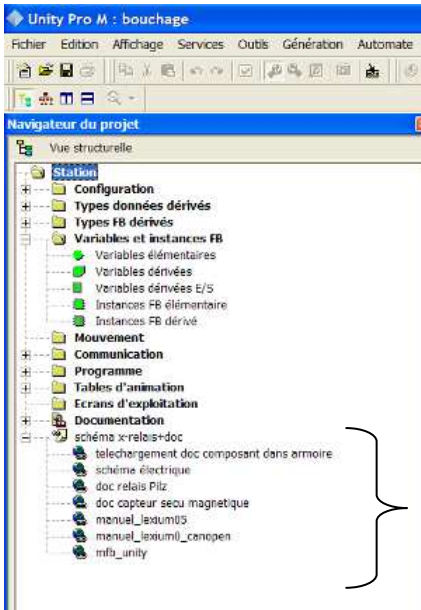


**Remarque** : on peut directement avoir accès à la fonction Remplacer depuis un éditeur de programme, en sélectionnant la variable avec la souris puis via le menu contextuel Initialiser la recherche, puis se rendre dans l'onglet Remplacer.

## Fonction supplémentaires de Unity

Unity vous permet de pouvoir insérer des répertoires utilisateur dans la vue structurelle de votre application, ainsi que des liens hypertextes (lien permettant d'aller chercher une donnée sur le disque dur).

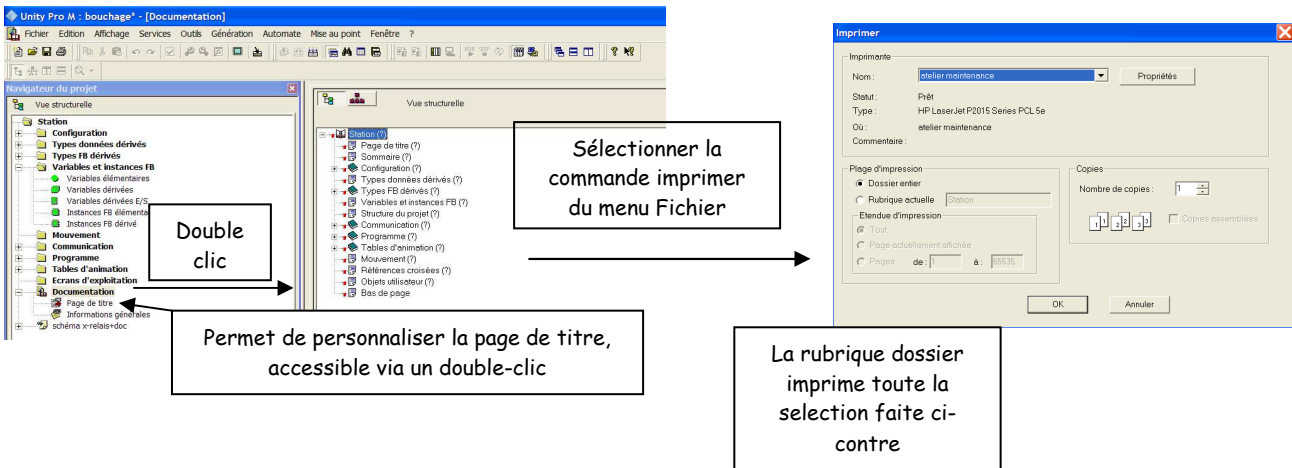
Pour cela faire afficher la vue structurelle du navigateur (cf.p 3,4), sélectionner le répertoire sous lequel on veut insérer un répertoire personnalisé, puis via le menu contextuel choisir la **commande Ajouter un répertoire utilisateur**, une boîte de dialogue s'ouvre demandant le nom à donner, valider, le répertoire est inséré dans l'arborescence de la vue structurelle. On peut ensuite insérer un lien hypertexte pour aller lancer une documentation technique, voire un autre logiciel, l'insertion de ce lien se fait toujours dans la vue structurelle par le menu contextuel **Ajouter lien hypertexte**.



### Exemple de répertoire et lien hypertexte

Répertoire et lien hypertexte inséré dans l'arborescence. Les liens hypertextes permettent de lancer les documentations technique des composants du projet, un lien a été également créé pour lancer le schéma électrique de la machine. **Attention si l'on ouvre le programme sur un autre PC, les liens seront toujours présents, mais ne marcheront que si tous les fichiers appelés sont stockés au même endroit que sur le PC d'origine**

### Fonction imprimer



**Remarque :** vous pouvez imprimer juste un éditeur, pour cela quand un éditeur (programmation, données...) est ouvert effectuer la commande imprimer du menu Fichier et choisir alors Rubrique actuelle, seul l'éditeur en cours est imprimé.

### Fonction enregistrer-archiver et exporter

Le format d'enregistrement par défaut d'une application est le format STU, par contre il existe d'autres format STA et XEF

**Format STA :** format compressé du projet

**Format XEF :** format d'échange entre version (format XML)

D'une manière générale le fichier STU ne peut être ouvert par Unity que dans la version avec laquelle il a été compilé. Par exemple un fichier STU compilé avec une version 3.1 de Unity, ne pourra être ouverte avec Unity V4.0. Les fichiers STA et XEF permettent de

pouvoir s'affranchir de ces limitations. **C'est pour cette raison qu'il est nécessaire après validation de l'application de disposer de ces trois formats de fichier.**

Prenons un exemple, une entreprise dispose de Unity V3.1 et ne dispose que des sauvegardes de programme au format STU. Si la mise à jour en versions supérieure est installée (V4.0), Unity V4.0 refusera d'ouvrir le fichier STU compilé en V3.1. **Par contre on pourra se connecter aux automates et récupérer les programmes existant, et le cas échéant les commentaires si le projet a été validé avec les options adéquat (cf.p 19).**

**Format STA** : sélectionner la commande **Archiver** du menu **Fichier**, une boîte de dialogue s'ouvre demandant le nom et l'emplacement où l'on veut stocker le fichier.

**Format XEF** : dans la vue structurelle ou fonctionnelle, sélectionner le répertoire STATION (en tête de l'arborescence), **sélectionner la commande Exporter** du menu **Fichier**, une boîte de dialogue s'ouvre demandant le nom et l'emplacement où l'on veut stocker le fichier.

**Remarque** : pour ouvrir un fichier STA, STU, XEF, sélectionner la commande ouvrir du menu **Fichier**. Le format XEF exporte la totalité du projet, on peut exporter juste un éditeur de programmation, la configuration matériel, ou les données de l'éditeur de données, pour cela se positionner dans la vue structurelle ou fonctionnelle sur l'élément que l'on veut exporter puis **sélectionner la commande Exporter** du menu **Fichier**, une boîte de dialogue s'ouvre demandant le nom et l'emplacement où l'on veut stocker le fichier.

**XLD** : format XML d'export de sections de programme ladder

**XIL** : format XML d'export de sections de programme List Instructions

**XST** : format XML d'export de sections de programme Litteral Structuré

**XBD** : format XML d'export de sections de programme FBD

**XSF** : format XML d'export de sections de programme grafcet

**XHW**: format d'export de la configuration matérielle

**XSX**: export de variables de l'éditeur de données

**XCR** : format d'export d'écrans d'exploitation

Ces éditeurs seront importés via la commande **Importer** du menu **Fichier** et si l'on est positionné sur le bon éditeur dans la vue structurelle.

**Petite astuce** : si vous ne disposez que du fichier STU, modifier l'extension de votre fichier en .STA et Unity vous l'ouvrira.

## **Logiciel additionnel à Unity**

Le logiciel OSLoader fourni avec Unity Pro permet d'effectuer les tâches suivantes :

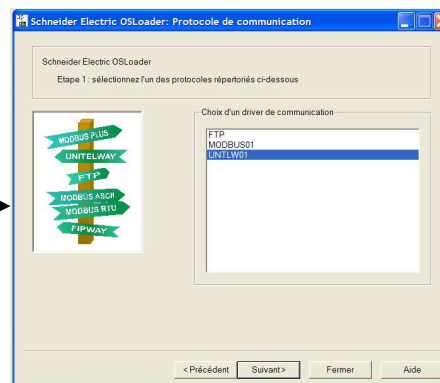
- mise à niveau des systèmes d'exploitation de certains processeurs Premium/Quantum

## Support à l'utilisation du logiciel Unity Pro

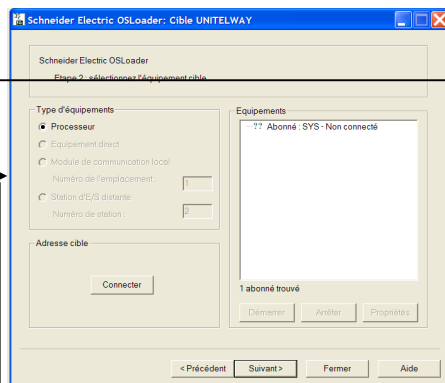
- mise à niveau des systèmes d'exploitation de certains processeurs Premium/Quantum équipés de ports Ethernet
- Mise à niveau des systèmes d'exploitation de certains modules Ethernet
- installation d'anciennes versions sur les processeurs Premium/Quantum afin qu'ils soient compatibles avec les logiciels de programmation PL7 V4 et Concept V2.6.

Le logiciel OS LOADER se trouve dans le répertoire :

Schneider Electric → Unity Pro → OS LOADER du menu démarrer de windows



Le mode de transfert le plus simple est via le protocole unitelway, on peut alors transférer un système d'exploitation directement avec le câble de transfert TSXPXC3030 ou 3031.



En cliquant sur le bouton démarrer une recherche des équipements connectés est lancée.



Sélectionner le sens de transfert du système d'exploitation. Les systèmes sont fournis sur CD avec le logiciel ou téléchargeable sur le site Schneider XSL, ce sont des fichiers au format bin.

Se laisser ensuite guider par les autres fenêtres

**Attention pendant la phase de mise à jour du système d'exploitation, surtout ne couper jamais l'alimentation de l'automate sous réserve de dégâts irréversibles. Vérifier bien la compatibilité de votre processeur avec le système d'exploitation que vous voulez transférer (se rendre dans l'aide du logiciel et consulter les tableaux de compatibilité). Il est également possible de faire migrer certaines CPU PREMIUM programmée avec PL7-Pro, par contre il est nécessaire au préalable de transférer un firmware intermédiaire (se référer à l'aide du logiciel).**

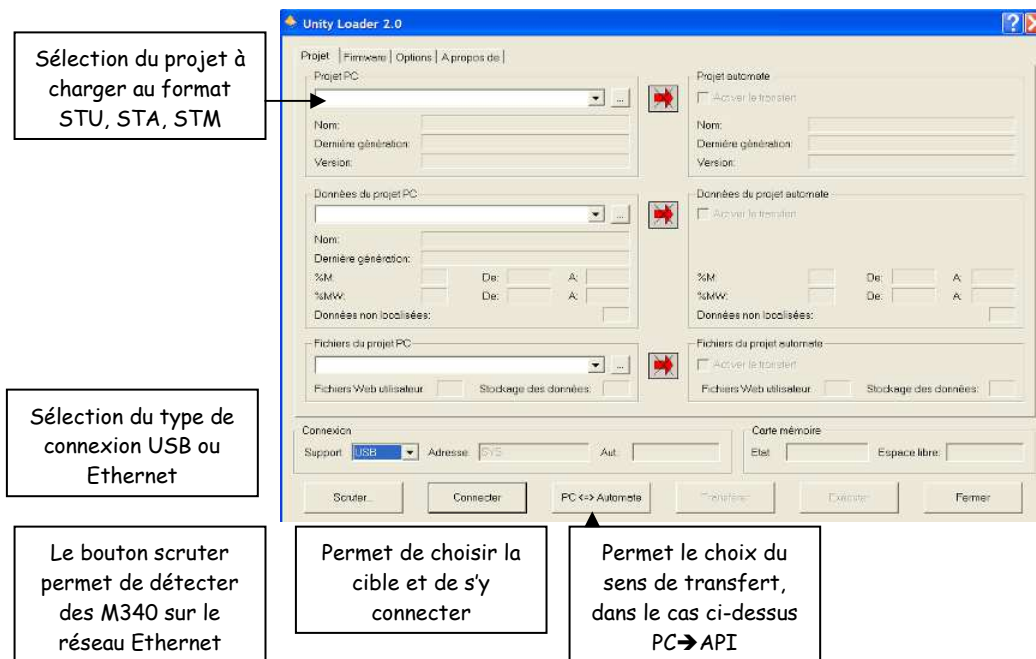
### Logiciel Unity Loader

Unity Loader est un outil logiciel autonome destiné à la plate-forme matérielle M340. Il n'est pas nécessaire d'avoir une licence Unity Pro pour utiliser ce chargeur. Le logiciel Unity Loader propose les fonctionnalités de transfert suivantes :

## Support à l'utilisation du logiciel Unity Pro

- transfert d'une application Unity Pro d'un PC vers un automate
- transfert d'une application Unity Pro d'un automate vers un PC
- transfert d'un micrologiciel (FW, Firmware) d'un PC vers un automate

Ce logiciel est très intéressant, il permet à une personne ne possédant pas le logiciel Unity de pouvoir transférer ou récupérer des données ou le programme complet. Il se lance depuis le menu démarrer de Windows → Schneider Electric → Unityloader



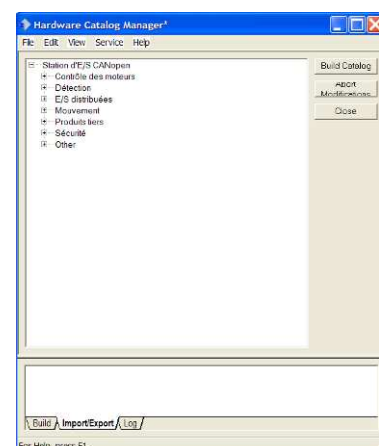
L'onglet Firmware permet de pouvoir mettre à jour le système d'exploitation du M340, il faut sélectionner un fichier *Idx* fournis sur CD avec le logiciel ou téléchargeable sur le site Schneider XSL.

### Logiciel Catalog Manager

Catalog Manager est un outil logiciel intégré à Unity V4.0 qui simplifie la gestion des équipements CANOpen dans la base de données de catalogue Unity Pro. Catalog Manager est un logiciel distinct permettant d'effectuer les opérations suivantes :

- Intégrer des produits de fournisseurs tiers
- Ajouter, supprimer et configurer l'accès aux équipements CANOpen sur le bus terrain
- Réduire la taille de la mémoire de l'UC réservée à un équipement donné
- Personnaliser l'interface utilisateur

Les équipements sont ajoutés dans le catalogue standard de Unity Pro et les équipements peuvent être utilisés dans les projets comme tous les équipements fournis avec Unity Pro.

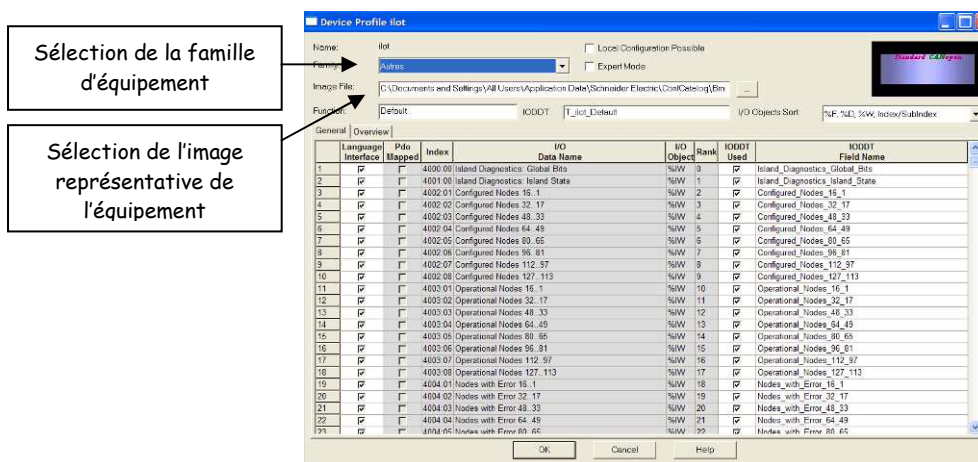


## Support à l'utilisation du logiciel Unity Pro

Ce logiciel permet de s'affranchir de l'utilisation de Sycon (logiciel permettant de gérer le bus Canopen, Profibus et Interbus, dans la configuration du maître et des esclaves).

Procédure d'ajout d'un équipement CANOPEN dans la base de CATALOG MANAGER :

- Sélectionner la commande **ADD DEVICE** du menu **EDIT**
- Aller chercher le fichier EDS de l'équipement sur votre disque dur
- Une boîte de dialogue s'ouvre (cf.ci-dessous ou vous pourrez attribuer une famille et un fichier d'image à votre équipement
- Valider, puis sous **CATALOGUE MANAGER** lancer la commande **BUILD CATALOG**, le fichier est alors intégré à la base d'équipement **CANOPEN**
- Lorsque vous ouvrez Unity, lors de la configuration du bus **CANOPEN**, vous verrez apparaître votre équipement et la famille à laquelle il se rattache.



**EDS : Electronic Data Sheet**, c'est un fichier texte contenant toute la description de l'équipement sur le bus.

Méthodologie de déclaration d'un réseau de communication

Les différents réseaux de communication que l'on pourra retrouver sont :

- Ethernet
- Modbus plus
- Fipway

En fonction des automates certaines CPU seront natives avec un ou plusieurs réseaux, sinon on pourra leur adjoindre des cartes faisant office de coupleurs, ou pour des CPU PREMIUM des cartes PCMCIA.

La méthode nécessite les quatre étapes suivantes :

- création d'un réseau logique
- configuration du réseau logique
- déclaration du module réseau ou de la carte PCMCIA (pour Premium) dans la configuration matérielle
- association de la carte ou du module au réseau logique

Création d'un réseau logique :

A partir du navigateur de projet, développez le répertoire Communication, cliquez avec le bouton droit sur le sous-répertoire Réseaux et choisissez l'option Nouveau réseau.

Effectuer les réglages ci-dessous puis valider par OK

Choix du réseau, dépend du type de CPU, si l'on avait configuré une CPU Premium TSXP57154M, on aurait eu les 3 réseaux énoncées ci-dessus.

Nom que l'on donne au réseau dans le projet

Choix de la famille à laquelle se rattache le réseau :

- CPU si celle-ci est native Ethernet
- Carte ou coupleur

La sélection se fait par la liste déroulante

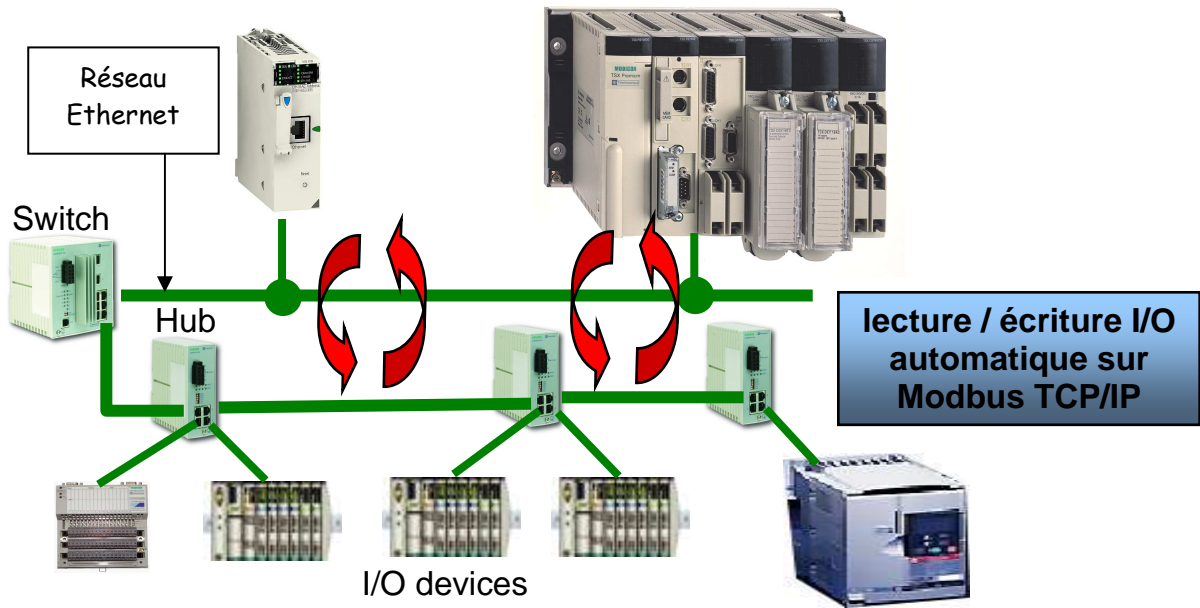
Service pris en charge sur le réseau à activer ou non

Paramètre du réseau Ethernet de la machine.  
La case à partir d'un serveur est utilisée quand on attribue pas une adresse IP fixe, mais celle-ci sera affectée par un serveur DHCP à chaque démarrage.

La fenêtre est directement lié au réseau que l'on configure, on s'intéresse ici juste au réseau Ethernet, pour les 2 autres se référer à l'aide du logiciel.

**Détails des services pris en charge sur Ethernet**

IO Scanning : ce service permet la lecture et l'écriture d'entrées-sorties déportées sur le réseau Ethernet. La période de scrutation des entrées-sorties est indépendant du cycle de la CPU pour chaque équipement. Le I/O scanner est utilisé périodiquement pour lire ou écrire des entrées/sorties distantes sur le réseau Ethernet sans programmation spécifique. Le I/O scanner est configuré sous Unity Pro.



Le nombre d'équipement dépend en général du type de coupleur ou de CPU en général on peut scruter 64 ou 128 équipement. La lecture et l'écriture des entrées et sorties se fait dans des zones mémoires %MW de la CPU, c'est le programmeur qui effectue le mapping (correspondance des mots entre les entrées-sorties des équipements et de la zone mémoire CPU) de chacun des équipements.

Après activation du service l'onglet devient accessible

Services du module: ID Scanning, Global Data, Service d'adresses, NTP

Zones %MW du maître: Lecture: De 0 à 12, Ecriture: De 10 à 12

	Adresse IP	ID unité	Timeout de validité (ms)	Période de répétition (ms)	Objet maître (lecture)	Index esclave (lecture)	Longueur (lecture)	Dernière valeur (entrée)	Objet maître (écriture)	Index esclave (écriture)	Longueur (écriture)	Description
1	192.168.207.222	255	1500	60	%MW0	5	5	Maintien de la voie	%MW0	2	1	
2	192.168.207.223	255	1500	60	%MW5	3	3	Maintien de la voie	%MW1	2	2	
3	192.168.207.224	255	1500	60	%MW8	2	2	Maintien de la voie	%MW3	6	6	
4												
5												
6												
7												
8												
9												
10												
11												
12												

Paramètres automatiquement calculés

Exemple de 3 équipements raccordés sur le réseau Ethernet, déclarés dans Unity.



Chaque équipement est représenté par une adresse IP fixe et univoque sur le réseau.

La zone %MW du maître de définir les plages de mots internes de la mémoire de l'application (%MW) spécifiques aux zones de lecture et d'écriture.

Pour ce faire, vous devez compléter :

- pour la zone de lecture, Lecture (adresse de démarrage dans la table des mots internes pour la lecture des entrées)
- pour la zone d'écriture, Ecriture (adresse de démarrage dans la table des mots internes pour l'écriture des sorties)

Le nombre de mots d'échanges dépend du type de modules utilisés.

Le champ Timeout de santé sert à définir l'intervalle maximal entre deux réponses d'un équipement distant, de 1 à 50000 ms.

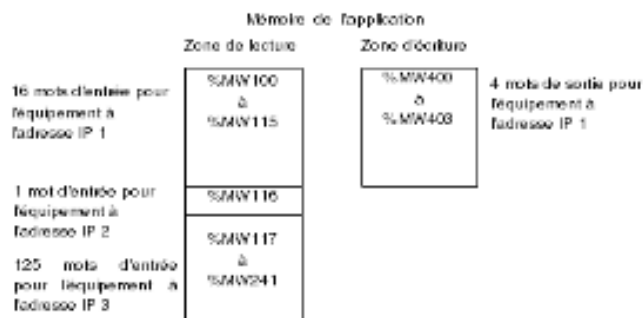
Le champ Période de répétition permet d'associer une adresse IP à sa [période de scrutation](#), de 0 à 50000 ms. plus la période de scrutation est courte, plus la mise à jour des entrées/sorties est rapide. Toutefois, cette vitesse augmente la charge du réseau.

### Exemple :

Un module Ethernet scrute trois équipements :

- un module à l'adresse IP1, 16 entrées analogiques, ce module possède 16 mots d'entrée et 4 mots de sorties
- un module à l'adresse IP2
- un automate Premium à l'adresse IP3 avec 125 mots d'entrées

La zone de lecture commence à %MW100 et la zone d'écriture à %MW400.



**Remarque :** les champs réservés aux équipements distants ne doivent pas se chevaucher. Il en est de même pour les zones de lecture et d'écriture.

### Service Global Data

Le service Global Data, a pour objet de permettre un échange de données automatique pour la coordination des applications d'automate sans ligne de programmes particulière, ce service n'est pris en charge que par certains modules ou processeurs.

Les modules de communication sont rassemblés dans un groupe de distribution afin d'échanger les variables utilisées pour la coordination de l'automate. Chaque module de communication publie une variable d'application locale pour les autres modules de communication du groupe.

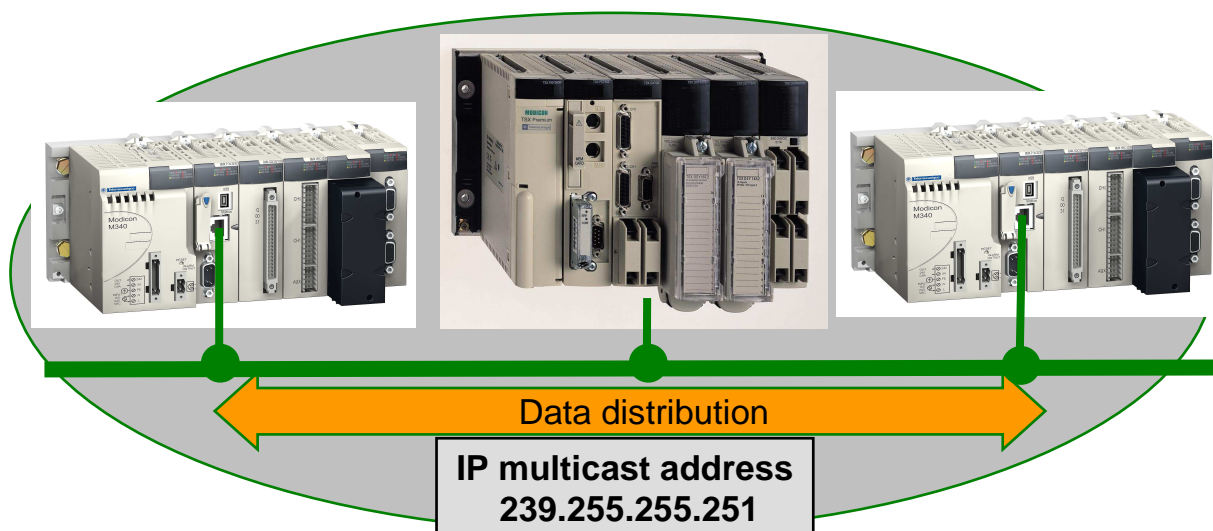
Chaque module peut également souscrire aux variables d'application publiées par l'ensemble des autres modules partageant le groupe de distribution, quel que soit son emplacement.

Une fois le module configuré, les échanges entre les modules de communication partageant le même groupe de distribution s'effectuent automatiquement lorsque l'automate est en mode RUN.

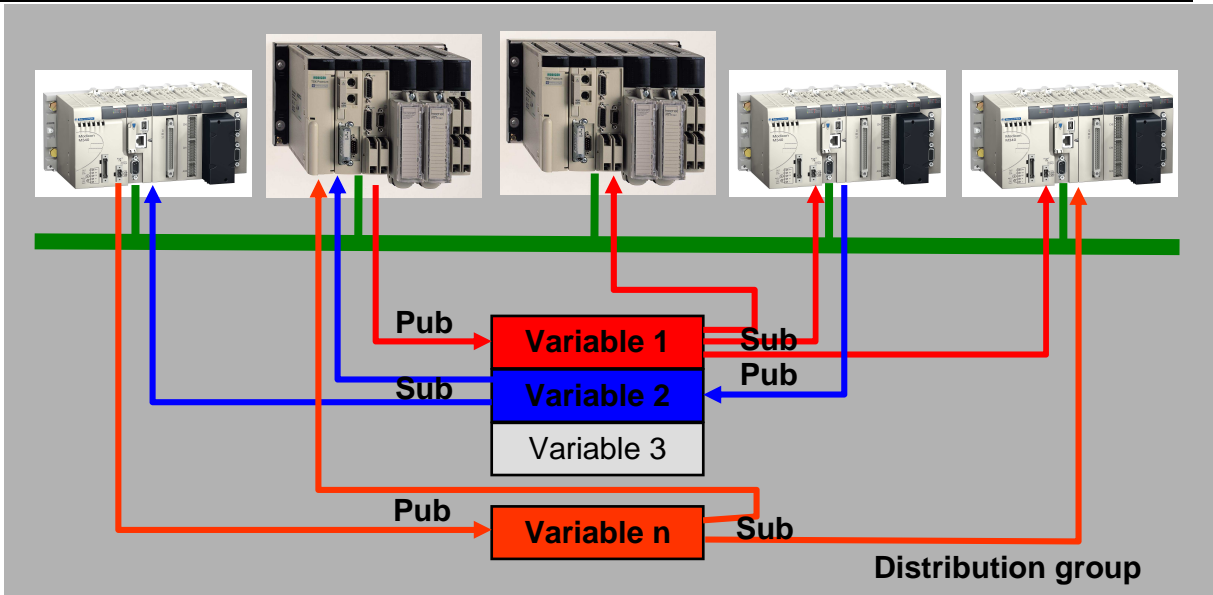
Un groupe de distribution est un groupe de modules de communication identifiés par la même adresse IP de multidiffusion. Les échanges en "multidiffusion" sont utilisés pour distribuer les données globales. Plusieurs groupes de distribution indépendants peuvent coexister sur un même sous-réseau avec leur propre adresse de multidiffusion.

Il n'existe aucune limite théorique au nombre de stations partageant un groupe de distribution. La limite principale réside dans le nombre de variables échangées au sein du groupe de distribution (64 variables).

- Chaque station peut publier une variable (jusqu'à 1024 bytes) et souscrire de 1 à 64 variables.
- La sélection des variables à publier ou souscrire s'effectue dans l'éditeur de données.



**Support à l'utilisation du logiciel Unity Pro**



**Ecran de configuration sous Unity Pro du service Global Data**

**Sélection du service Global Data**

**Après activation du service l'onglet devient accessible**

**Adresse IP de multi-diffusion**

**Nom du groupe de diffusion**

**Déclaration sous Unity des variables à souscrire et publier pour la machine 1**

Nom	Type	Adresse	Valeur	Commentaire	Donnée globale	ID	Groupe
com_regroup1	ARRAY[0..1] OF INT	%MW22		mot à destination du poste de regroupement	PUB	2	SUP
com_regroup2	ARRAY[0..1] OF INT	%MW24		mot venant du poste de regroupement via...	SUB	1	SUP

**Variable publiée**

**Variable souscrite**

Tableau de 2 mots de type integer pour la publication

Tableau de 2 mots de type integer pour la souscription

Nom du groupe de diffusion en adéquation avec la config décrite ci-dessus (SUP)

**Déclaration sous Unity des variables à souscrire et publier pour la machine 2 (autres fichier de programme Unity)**

Nom	Type	Adresse	Valeur	Commentaire	Donnée globale	ID	Groupe
com_bouch1	ARRAY[0..1] OF INT	%MW41		mot à destination du poste de bouchage	PUB	1	SUP
com_bouchage	ARRAY[0..1] OF INT	%MW43		mot venant du poste de bouchage via glob...	SUB	2	SUP

**Variable publiée**

**Variable souscrite**

Tableau de 2 mots de type integer pour la publication

Tableau de 2 mots de type integer pour la souscription

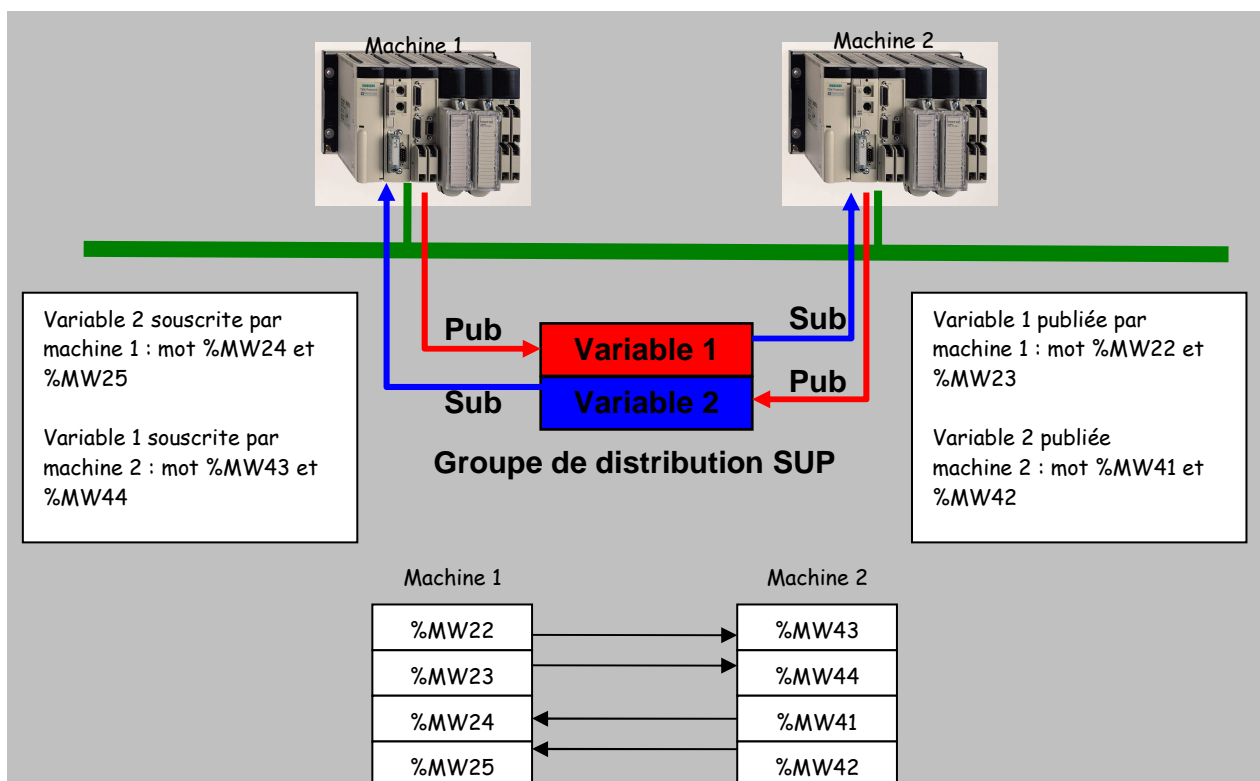
Nom du groupe de diffusion en adéquation avec la config décrite ci-dessus (SUP)

**Remarque** : il est nécessaire de faire afficher les colonnes SUB (souscription) et PUB (publication) dans l'éditeur de données, pour cela ouvrir l'éditeur de données, dans le bandeau supérieur d'une colonne choisir personnaliser colonne via le menu contextuel.

La configuration du service Global Data est la même pour la machine 2 que pour la machine 1, il est fondamental que l'adresse IP de multidiffusion soit la même sur les deux programmes Unity. L'ID de données doit être univoque entre 1 et 64, c'est grâce à cet ID que l'on pourra définir quelle variable sera souscrite pour tel équipement.

Donc si on reprend les 2 configurations global data de la machine 1 et 2 en page précédente, on peut donc en déduire que :

- La machine 1 lit 2 mots de la machine 2 et vis versa



### Service Serveur d'adresses

Les services BOOT/TP et DHCP permettent d'obtenir directement la configuration IP de l'équipement depuis un serveur.

**DHCP** : le protocole DHCP (Dynamic Host Configuration Protocol) gère les paramètres réseau pour les équipements réseau. Les équipements individuels peuvent obtenir des adresses IP réseau d'un serveur DHCP par le biais d'une requête

Certains modules peuvent être configurés comme serveur DHCP. Leur adresse peut également être configurée par l'utilisateur ou allouée dynamiquement par un serveur d'adresses (configuration en tant que client BOOTP).

## Support à l'utilisation du logiciel Unity Pro

Un module configuré comme client BOOTP transmet des requêtes sur le réseau toutes les secondes au cours du démarrage, jusqu'à l'obtention d'une réponse de la part d'un serveur d'adresses.

L'équipement distant agissant comme serveur BOOTP/DHCP répond à cette requête et affecte au module client :

- une adresse IP
- l'adresse IP du Gateway
- le masque de sous-réseau correspondant

La distribution des adresses IP fait partie intégrante du service de configuration. BOOTP et DHCP allouent tous deux les adresses IP au démarrage, mais utilisent des méthodes d'allocation différentes. En général, BOOTP procède par allocation fixe d'une unique adresse IP à chaque client, et réserve cette adresse de façon permanente dans la base de données du serveur BOOTP. Tandis que DHCP procède par allocation dynamique des adresses IP disponibles, réservant chaque adresse de client DHCP de façon temporaire dans la base de données du serveur DHCP.

La différence entre DHCP et BOOTP est que DHCP peut fournir à une station une certaine plage d'adresses et que chacune de ces adresses est négociée et n'est valable que pour une certaine période de temps.

Nécessité pour le serveur de connaître l'adresse MAC (physique) du client pour être autorisé à lui répondre et disposer de ses paramètres IP. **En bootp, il est nécessaire d'utiliser une table contenant l'adresse mac des différents protagonistes en correspondance avec les adresses IP à leurs affecter.**

### Ecran de configuration sous Unity Pro du service Serveur d'adresses

	Adresse MAC	Nom	Adresse IP	Masque réseau	Gateway
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					

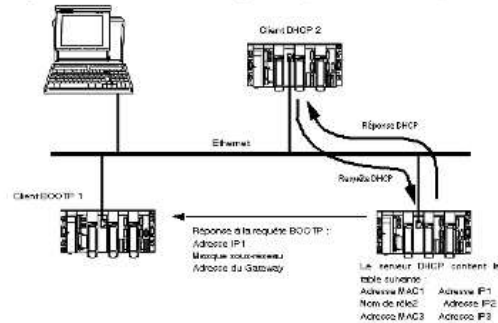
Sélection des équipements à adresser en fonction de leurs adresse MAC (bootp, ou de leur nom (DHCP). Les infos adresses IP, Masque de réseau et passerelles sont transmises à l'équipement concernés

Access	MAC address	Name	IP address	Netmask
1		Device1	139.168.12.32	255.255.0.0
2	00.00.54.00.1D.B7		139.168.12.31	255.255.0.0
3	00.00.54.00.1F.ED		139.168.12.33	255.255.0.0

Exemple d'une configuration

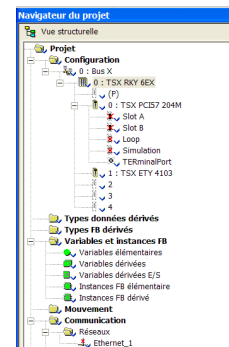
Exemple de serveur DHCP

Le diagramme suivant illustre le routage des requêtes lors d'une réponse à une requête de démarrage d'un serveur :

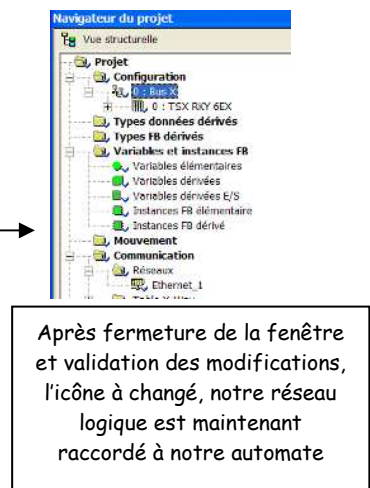
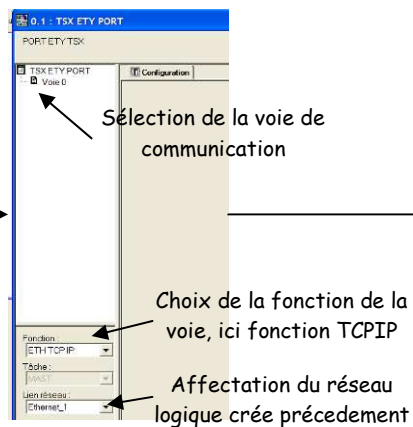
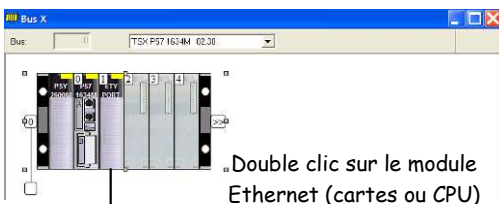


**Remarque :** le choix de client bootp, ou serveur DHCP est soit fixe pour certains modules, voire configurable par l'intermédiaire du serveur web embarqué (réglage en se connectant au serveur web par un navigateur internet, par exemple internet explorer).

Après avoir réglé les différents services à utiliser et les différents paramètres présentés page 79, on voit apparaître notre réseaux crée, dans le répertoire réseaux (cf.ci contre).



Il faut maintenant attacher ce réseau logique à une voie de communication d'un module Ethernet (ou CPU ci-celle-ci est native Ethernet) dans la configuration matérielle.



Méthodologie de création d'un bloc DFB et gestion des bibliothèques

Le logiciel Unity Pro vous permet de créer des blocs fonction utilisateur DFB, en utilisant les langages d'automatismes. Un DFB (Derivated Function Block) est un bloc de programme que vous avez écrit, afin de répondre aux spécificités de votre application.

Les blocs fonction vous permettent de structurer et d'optimiser votre application. Vous pouvez les utiliser dès qu'une séquence de programme est répétée plusieurs fois dans votre application ou pour figer une programmation standard. L'export puis l'import de ces blocs fonction permet leur utilisation par un groupe de programmeurs travaillant sur une même application ou dans des applications différentes.

Par rapport à un sous programme SR, l'utilisation d'un DFB permet :

- ❖ d'utiliser des variables internes propres au DFB, donc indépendantes de l'application
- ❖ de tester son fonctionnement indépendamment de l'application

Un bloc DFB comprend :

- une ou plusieurs sections écrites en langage à contacts (LD), en liste d'instructions (IL), en littéral structuré (ST) ou en langage à blocs fonctionnels (FBD)
- des paramètres d'entrée/de sortie
- des variables internes publiques ou privées

### Paramètre d'un bloc DFB :

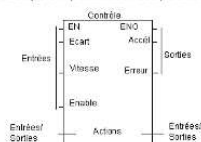
**Entrées** : ces paramètres permettent de passer des valeurs du programme application vers le programme interne du DFB. Ils sont accessibles en lecture par le DFB, mais ne sont pas accessibles par le programme application.

**Sorties** : ces paramètres permettent de passer des valeurs du DFB vers le programme application. Ils sont accessibles en lecture par le programme.

**Entrées-sorties** : ces paramètres permettent de passer des données du programme application vers le DFB qui peut la modifier et la repasser vers le programme application. Ces paramètres ne sont pas accessibles par le programme application.

#### Paramètres EN et ENO

EN est un paramètre d'entrée et ENO un paramètre de sortie. Ils sont de type BOOL et peuvent être ou ne pas être utilisés (facultatif) lors de la définition d'un type de DFB. Dans le cas où l'utilisateur souhaite les utiliser l'éditeur les positionne automatiquement. EN est le premier paramètre d'entrée et ENO le premier paramètre de sortie. Exemple d'implémentation des paramètres EN/ENO



**Remarque** : le nombre d'entrées + d'entrées/sorties doit être inférieur ou égal à 32. Le nombre de sorties + d'entrées/sorties doit être inférieur ou égal à 32.

**Variable public** : Ces variables internes du DFB peuvent être utilisées par le DFB, par le programme application dans des sections de programme à l'extérieur du bloc DFB.

**Variable privées** : Ces variables internes de DFB peuvent être utilisées uniquement par ce bloc de fonction et ne sont par conséquent pas accessibles par le programme application.

On se décide à créer un bloc DFB permettant de réaliser un compteur, on nommera notre bloc compteur, on trouvera :

- 1 entrée RAZ de type EBOOL (format capable de détecter les fronts)
- 1 entrée de comptage de type EBOOL (format capable de détecter les fronts)
- 1 entrée de présélection de type UINT pour la valeur maximale jusqu'à laquelle compter
- 1 variable que l'on pourra utiliser à l'extérieur du programme (variable public) contenant la valeur de comptage
- 1 variable de sortie de type EBOOL indiquant que la valeur du compteur est supérieure à la présélection

Nom	N°	Type	Valeur	Commentaire
compteur	<DFB>			
entrées				
raz	1	EBOOL		
comptage	2	EBOOL		
presélection	3	UINT		
sorties				
done	1	EBOOL		
variables/variables publiques				
public				
valeur_comptage		UINT		
sections				
comptage	<LD>			

Comptage par opération d'addition

signalisation valeur de comptage atteinte

RAZ compteur

Programme gérant le fonctionnement du bloc



**Support à l'utilisation du logiciel Unity Pro**

Après avoir terminé la modification du bloc de code, il faut analyser le bloc DFB :

L'icône a changé, si la validation est OK

Le bloc DFB apparaît avec sa section de programme

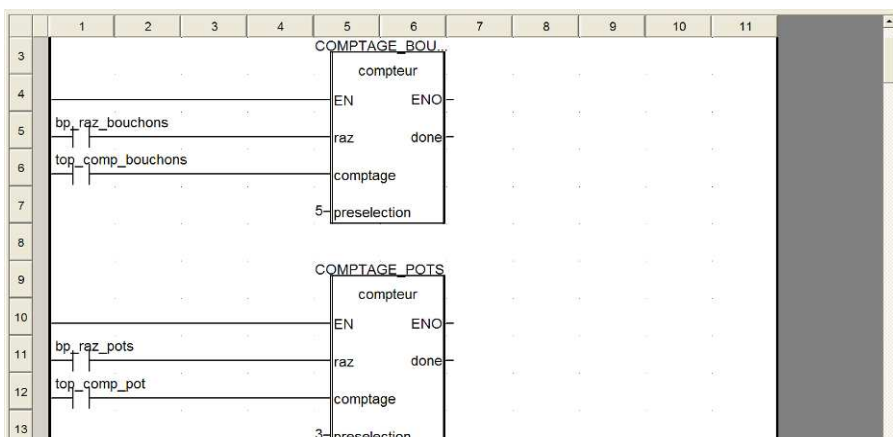
Sélectionner le nom du bloc, puis via le menu contextuel, choisir analyser type, s'il n'y a pas d'erreurs l'icône a changé. Si des erreurs sont détectées, elles seront affichées dans la fenêtre de visualisation (cf. p 44) en bas de l'écran et l'icône ne change pas.

A l'issu ouvrez votre éditeur de programme et sélectionné assistant de saisie FFB

Donner ensuite un nom à l'instance puis valider

Dans l'onglet application, on retrouve notre bloc DFB, que l'on sélectionne, puis on valide

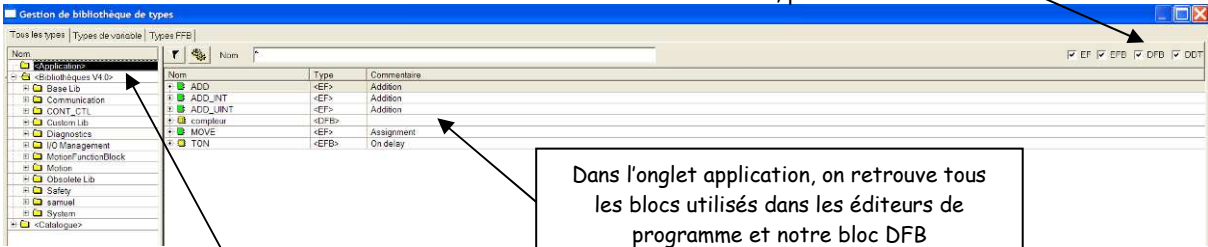
Répéter cette opération autant de fois que nécessaire, sur l'exemple ci-dessous on trouve deux blocs DFB compteur chacun gérant sa propre opération de comptage.



## Placement du bloc DFB crée dans la bibliothèque de Unity

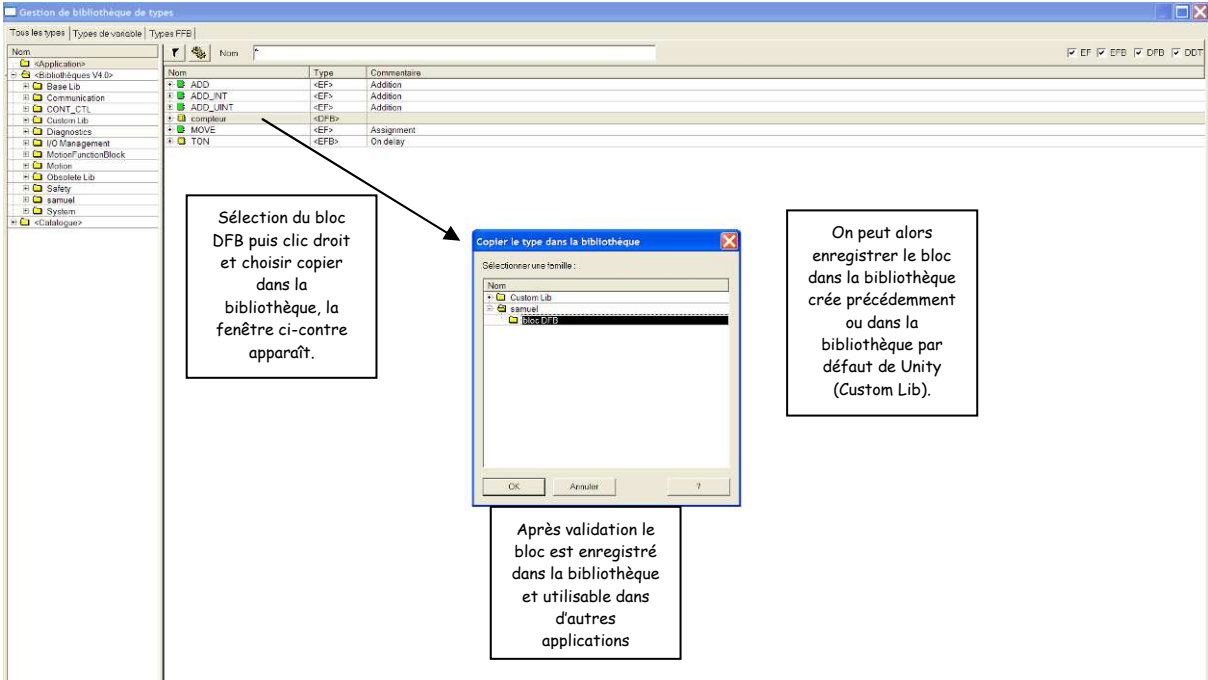
Activer la commande **Gestion de bibliothèques de types** du menu **Outils** ou via l'icône de la barre service (icône en forme de tiroir). La fenêtre ci-dessous apparaît.

Activer la case DFB, pour voir notre bloc



En sélectionnant le répertoire Bibliothèque, puis clic droit créer une nouvelle bibliothèque, une fenêtre apparaît, donner alors le nom que vous voulez (ici samuel) puis valider. A l'issu votre bibliothèque est insérée. On peut créer une sous famille (un sous répertoire à notre bibliothèque)

Dans l'onglet application, on retrouve tous les blocs utilisés dans les éditeurs de programme et notre bloc DFB



Sélection du bloc DFB puis clic droit et choisir copier dans la bibliothèque, la fenêtre ci-contre apparaît.

On peut alors enregistrer le bloc dans la bibliothèque crée précédemment ou dans la bibliothèque par défaut de Unity (Custom Lib).

Après validation le bloc est enregistré dans la bibliothèque et utilisable dans d'autres applications

## Conversion de projet

Unity vous permet de convertir des projets PL7-Pro en type de projet Unity. Depuis la version 4.0, le convertisseur peut convertir des applications TSX micro, avec cependant certaines restrictions.

## Support à l'utilisation du logiciel Unity Pro

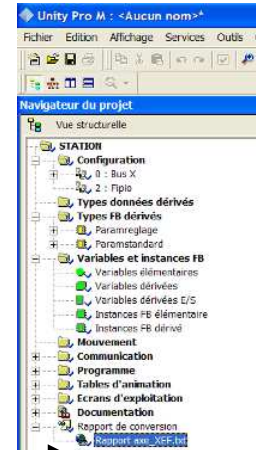
Pour la conversion vous devez disposer des fichiers XEF de PL7-Pro de l'application que l'on veut convertir.

Sélectionner la commande ouvrir du menu fichier, choisir l'extension de fichier FEF, puis aller chercher votre fichier sur le disque dur, puis valider, la conversion débute.

A l'issue de la conversion, un rapport de conversion est émis, il faut le consulter pour voir quelles sont les erreurs de conversion éventuelles. A l'issue de la conversion Unity, vous proposera automatiquement une CPU compatible avec celles de PL7.

Les erreurs et les avertissements s'affichent également dans la fenêtre de visualisation (cf.p 44).

Double clic pour ouvrir le rapport de conversion



Cas particulier pour les TSX micros, à l'issue de la conversion le logiciel indique un message : **Erreur d'analyse l'UC TSX37 ne peut être convertie en UC Unity compatible.**

Ce message vient du fait qu'il n'y a pas d'automates similaires dans Unity au TSXMICRO. Ce message est sans gravité pour la conversion, Unity place par défaut une CPU M340, il suffit ensuite de se rendre dans la configuration matérielle et le cas échéant de modifier la CPU.

## Sommaire

Les différents éditeurs	p 2
Architecture mémoire et règles de recouvrement	p 7
Adressage topologique variables	p 8
Les différents formats de variables	p 11
Création d'une application	p 18
Définition des options de projet	p 19
Configuration matérielle	p 22
Configuration du processeur	p 23
Configuration entrées-sorties T.O.R	p 24
Configuration voies analogiques	p 25
Edition et création de variables dans l'éditeur de données	p 25
Création de variables de types IODDT	p 27
Création de variables de types tableaux et structures	p 29
Création de blocs fonctions	p 31
Blocs fonctions temporisations et compteurs	p 33
Editeurs de programmation	p 35
Tâches rapides et maître	p 36
Tâches événementielle d'entrées-sorties	p 40
Tâches événementielle de type TIMER	p 41
Analyse et génération de projets	p 43
Transfert de l'application dans l'automate	p 45
Animation de programme en mode connecté	p 48
Outils de mise au point en mode connecté	p 50
Outils de réglages (points d'arrêts, point de visu)	p 55
Diagnostic système	p 58
Diagnostic projet	p 58
Simulateur de programme intégré à Unity	p 63
Ecran d'exploitation	p 65
Rechercher, remplacer variables	p 72
Enregistrer, archiver, exporter	p 74
Logiciel OS Loader	p 76
Logiciel Unity Loader	p 76
Logiciel Catalog Manager	p 77
Réseaux de communication	p 79
Service I/O scanner	p 80
Service Global Data	p 81
Service de serveur d'adresses	p 84
Création de blocs DFB et utilisation des bibliothèques	p 86
Conversion de projet	p 90

Auteur : Samuel Bonot

Support à l'utilisation du logiciel Unity Pro

# LOGICIEL UNITY PRO

