

Fiche de programmation Unity

Automate Schneider M340



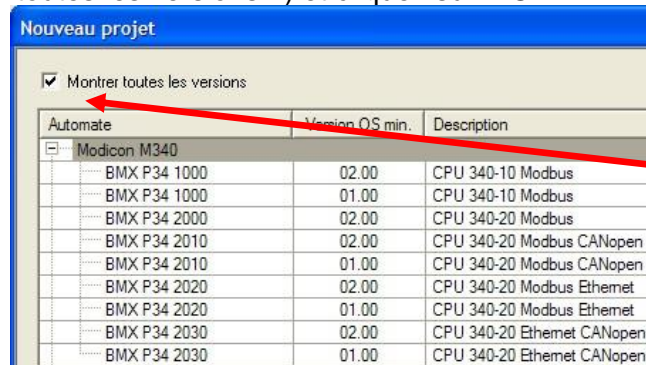
Lancez le programme Unity sous Windows en cliquant 2 fois sur l'icône. Passez au paragraphe 1 ou 2.

1. Ouverture d'un fichier existant :

Cliquez sur « Fichier », puis sur « Ouvrir ». Sélectionnez votre fichier (*.STU) dans son répertoire et cliquez sur « OK ». Si ce fichier a été créé par une autre version de Unity, sélectionnez le fichier (*.STA). Passez au paragraphe 3.

2. Création d'une programmation :

Cliquez sur « Fichier », puis sur « Nouveau ». Choisissez votre CPU et sa version (attention, il faut cocher la case « Montrer toutes les versions ») et cliquez sur « OK ».

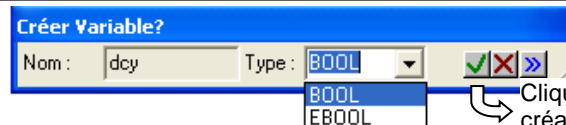
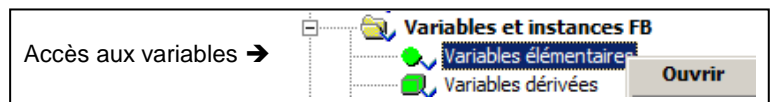
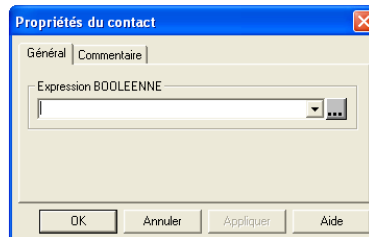
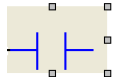


Automate à choisir pour les TP

La suite de la configuration ne se fera qu'au moment du transfert du programme dans l'automate, donc après son développement et sa validation sur le simulateur intégré.

3. Ecriture des mnémoniques :

Il n'est pas nécessaire, à ce stade, de déclarer toutes les variables utilisées dans le programme. Cela se fera au fur et à mesure de la création de chaque variable nouvelle.



Cliquez pour valider la création de la variable

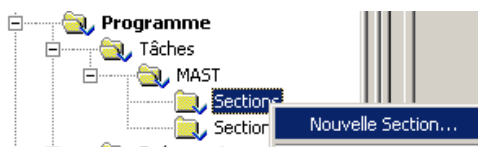
Ajouter la lettre « e » devant les variables d'entrée et un « s » devant les variables de sorties.

4. Ecriture du programme :

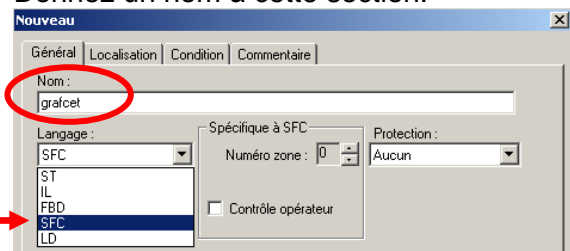
Choix en fonction du langage de programmation utilisé :

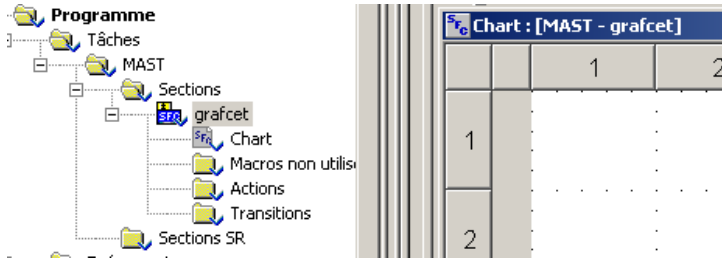
- Programmation en langage SFC (grafcet) :
- Programmation en langage LD d'un grafcet : voir annexe 5 page 22.

Insérez une section SFC (grafcet) :

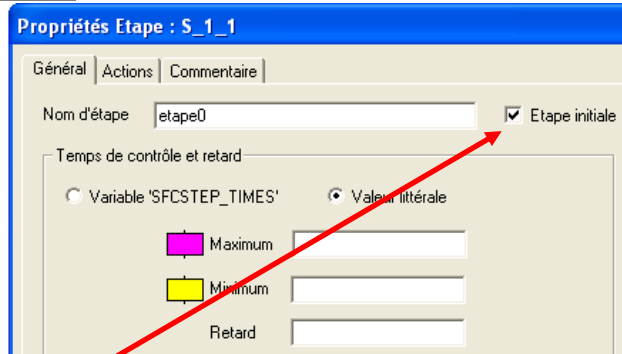
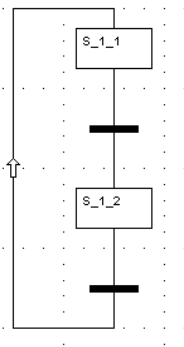
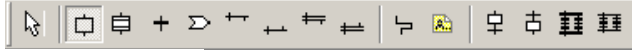


Donnez un nom à cette section.





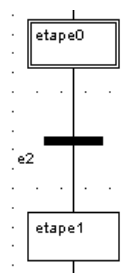
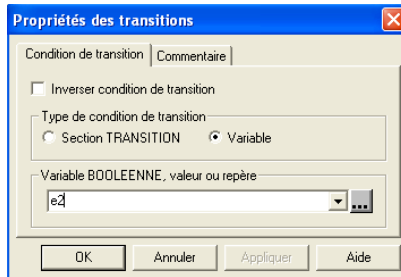
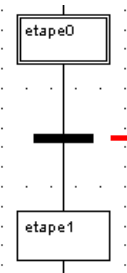
Barre d'outils pour la création d'un grafcet :



L'étape initiale se définit dans la propriété de l'étape.

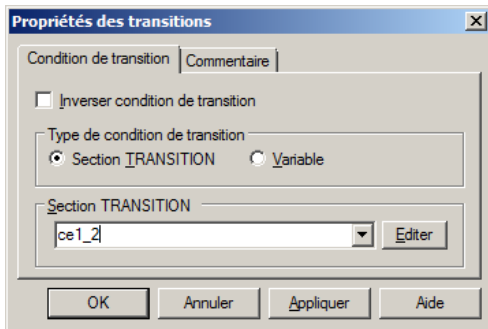
Transitions :

→ Cas d'une seule variable TOR :



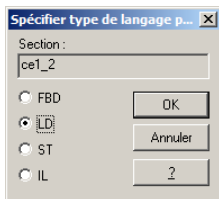
Il suffit d'indiquer le nom de cette variable dans la boîte de dialogue (cliquer sur Inverser pour obtenir la variable /e2).

→ Cas d'une combinaison de variables TOR (équation booléenne) :

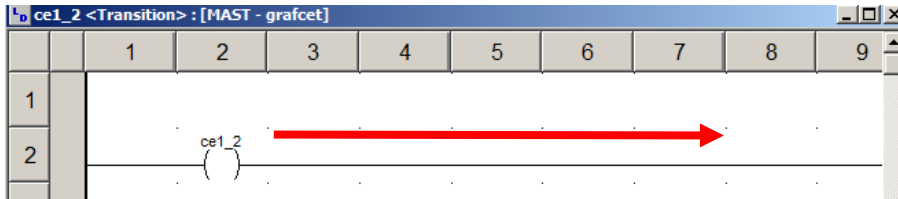


Il faut définir une section TRANSITION, lui donner un nom et cliquer sur Editer.

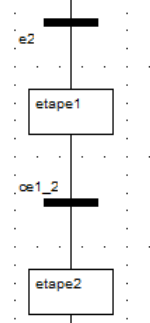
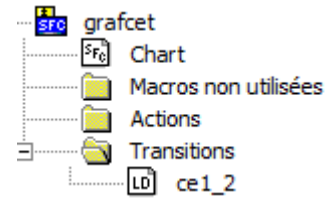
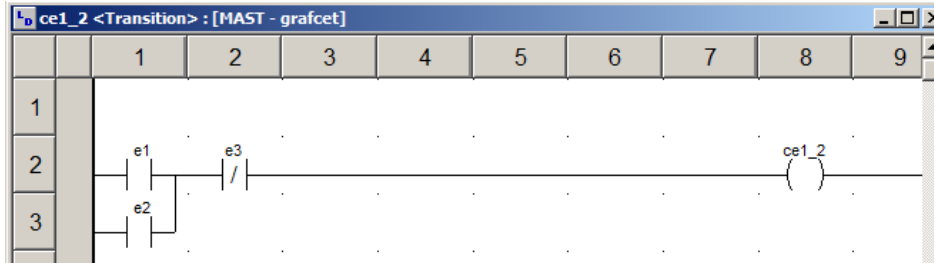
ce1_2 symbolise la condition d'évolution de l'étape 1 vers l'étape2



Il faut choisir le langage LD et éditer la section TRANSITION.

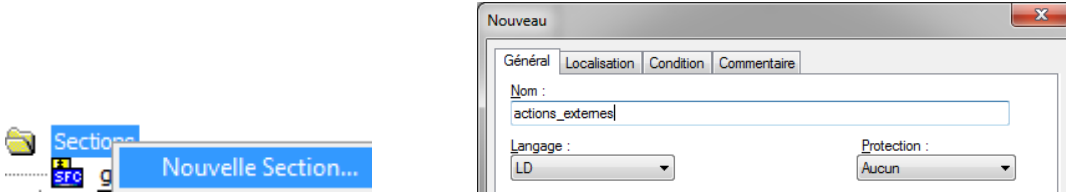


Déplacer la bobine à droite et compléter la transition (ici : /e3.(e1+e2)).

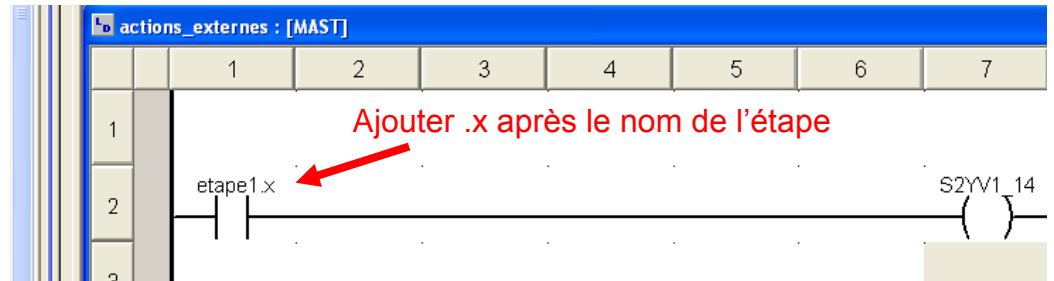
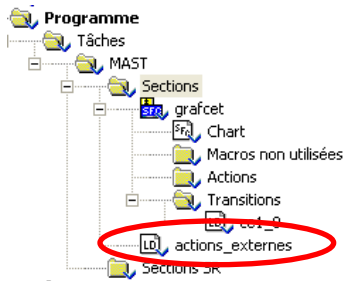


Actions grafcet :

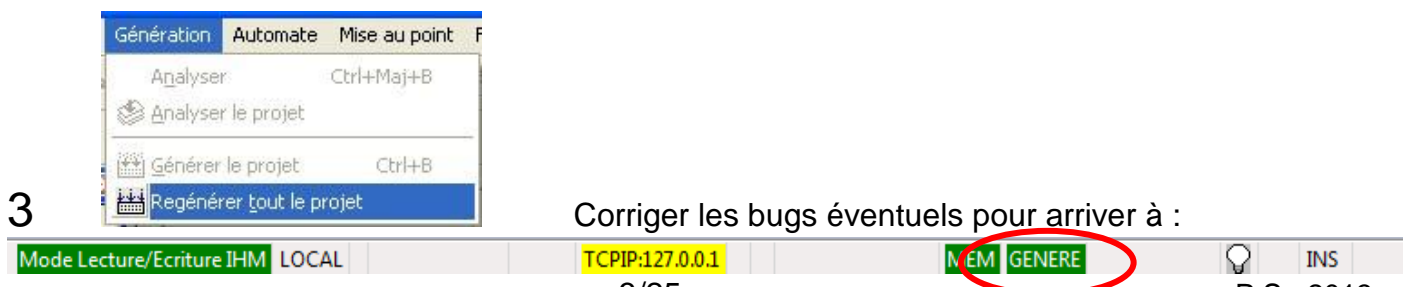
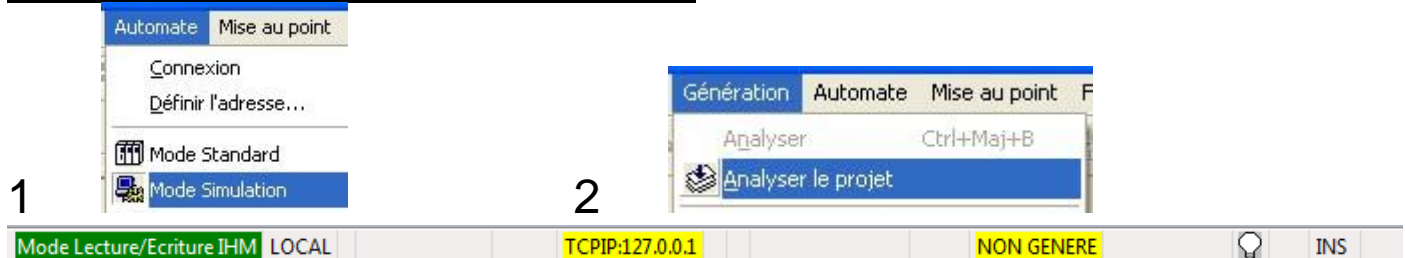
Programmation des sorties dans une section LADDER « Actions_externes » :
Créer une section « actions_externes » en LADDER



Programmer les équations des sorties :
Exemple : S2YV1_14 := étape1

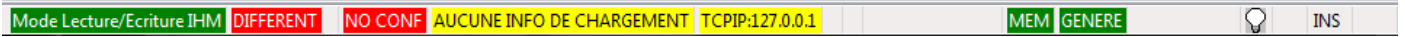
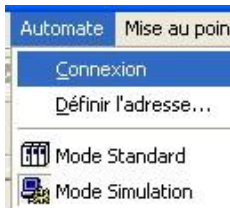


5. Test du programme avec le simulateur intégré :

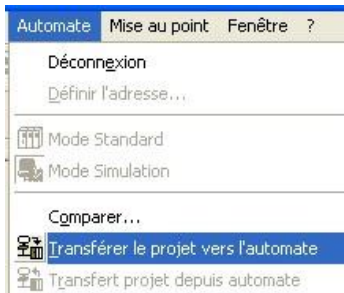


Corriger les bugs éventuels pour arriver à :

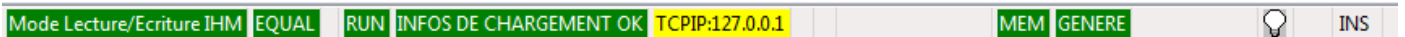
4



5



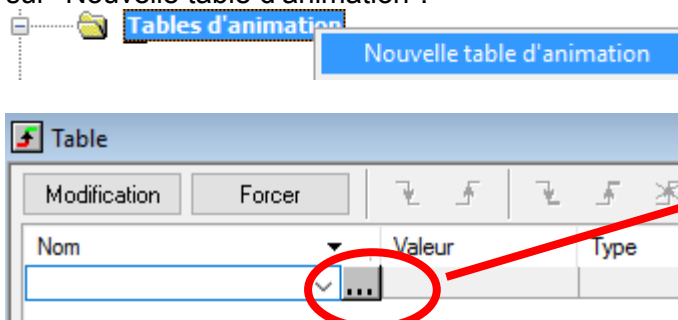
6



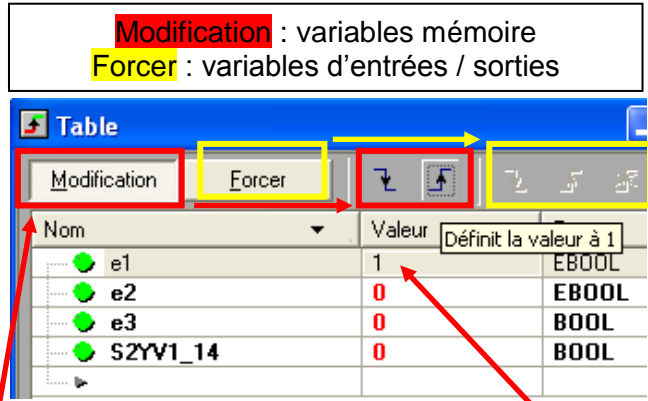
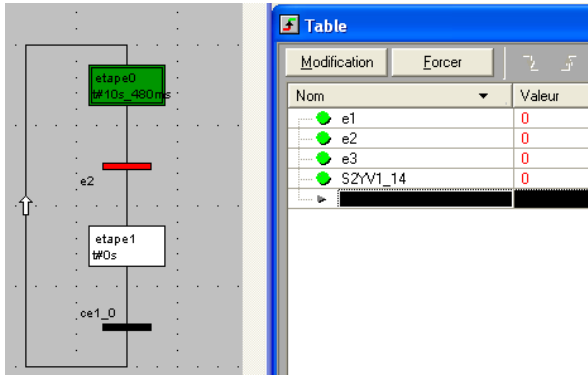
Si besoin, accès à l'automate de simulation :

Test avec une table d'animation :

Vous devez créer une table d'animation en faisant un clic droit sur "table d'animation" et en cliquant sur "Nouvelle table d'animation".



Sélectionnez les variables d'entrées et de sorties. Vous pouvez sélectionner plusieurs variables avec les touches SHIFT et CTRL

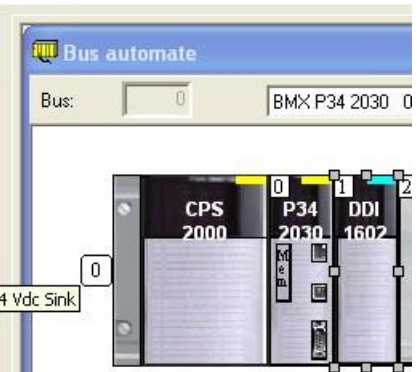
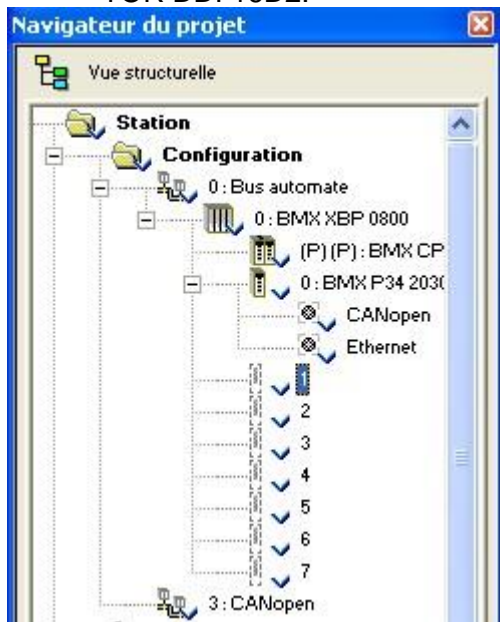


Visualisez vos variables grâce à cette table. Cliquez sur « **Modification** » ou « **Forcer** », changez les valeurs de vos variables, testez le programme et corrigez les erreurs.

6. Configuration de l'automate :



Double cliquez sur « Bus automate »
 Déroulez la partie configuration et double cliquez sur l'emplacement 1 pour insérer la carte d'entrée TOR DDI 16D2.



Faire de même avec la carte de sortie TOR DRA 0805 en position 2.
 Fermez la fenêtre « Bus automate ».

7. Attribution d'adresses réelles :

Retournez dans la partie déclarations des variables et remplissez la colonne « adresse » pour les variables d'entrées sorties.

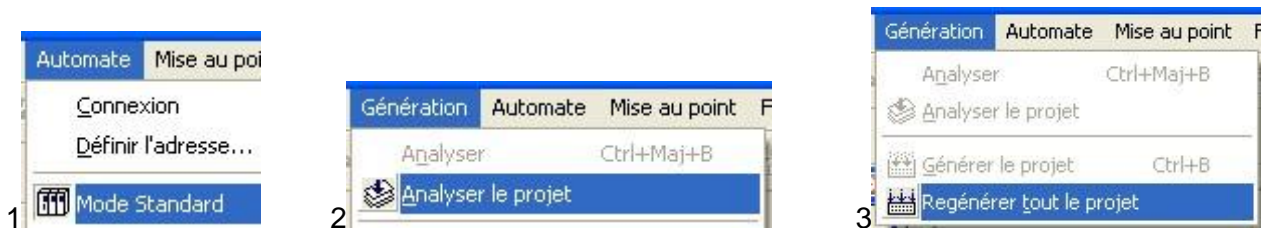
Les entrées de la carte 1 vont de %I0.1.0 à %I0.1.15.

Les sorties de la carte 2 vont de %Q0.2.0 à %Q0.2.15.

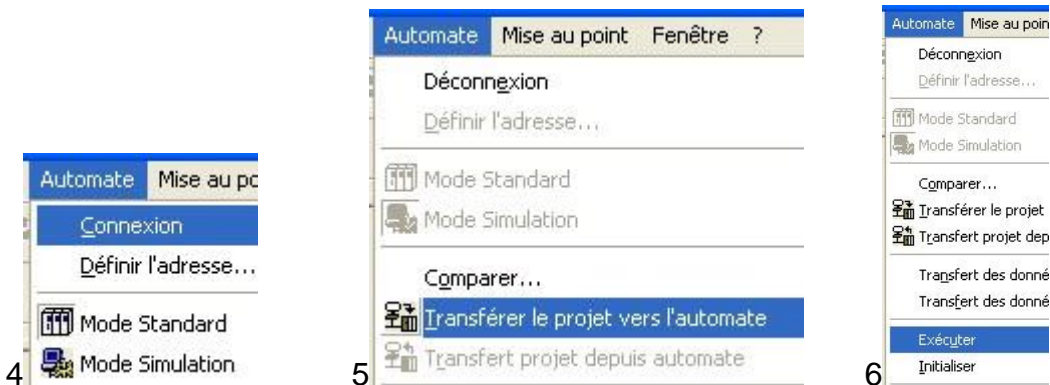
Pour un projet avec un écran, il faut attribuer des adresses aux variables de communication (variables échangées entre l'API et le IHM (terminal de dialogue)).

Variables Types DDT Blocs fonction Types DFB				
Filtre <input type="text" value="Nom = *"/>				
Nom	Type	Adresse	Valeur	Commentaire
consigne_pupitre_descente	INT	%mw2		
consigne_pupitre_montee	INT	%mw1		
f1_0	EBOOL	%m103		BP montée en marche manu
f2_0	EBOOL	%m104		BP descente en marche manu

8. Test réel :



Pour une connexion par Ethernet, vous devez configurer Ethernet (voir annexe 2 page 7)



Testez votre programme.

9. Impression :

Double cliquez sur « Documentation » dans le navigateur du projet. Sélectionnez les rubriques à imprimer par un clic droit et "inclure la rubrique".



Seule la rubrique programme contient votre travail, donc ne sélectionnez qu'elle.

Le reste des rubriques (pour un TP) est superflue à imprimer.

Cliquez droit sur « Projet » puis sur « Générer la documentation ».



Cliquez sur « Projet », puis sur « Imprimer ».

10.Sauvegarde et sortie :

Cliquez sur « Enregistrer sous », sélectionnez le répertoire de votre classe, donnez un nom « xxxxx.STU » (xxxxx correspondant à votre nom) et cliquez sur OK.

Cliquez sur « Archiver », sélectionnez le répertoire de votre classe, donnez un nom « xxxxx.STA » (xxxxx correspondant à votre nom) et cliquez sur OK.

Cliquez sur « Fichier » puis sur « Quitter » pour quitter le programme.

Programmation M

Unity Pro XL : POINÇONNEUSE

Fichier Edition Affichage Services Outils Génération Automate Mise au point Fenêtre Aide

Navigateur du projet

- Projet
 - Configuration
 - 0 : Bus automate
 - 0 : BMX XBP 0400
 - (P) (P) : BMX CPS 2000
 - 0 : BMX P34 20.10
 - 1 : BMX DDI 1602
 - 2 : BMX DRA 0805
 - 3
 - 3 : CANopen
 - Types données dérivés
 - Types FB dérivés
 - Variables et instances FB
 - Variables élémentaires
 - Variables dérivées
 - Variables de DDT d'équipement
 - Variables dérivées E/S
 - Instances FB élémentaire
 - Instances FB dérivé
 - Mouvement
 - Communication
 - Programme
 - Tâches
 - MAST
 - Sections
 - grafcset
 - Chart
 - Macros non utilisées
 - Actions
 - Transitions
 - LD ce0_1
 - actions_externes
 - Sections SR
 - Evénements
 - Evénements TIMER
 - Evénements E/S
 - Tables d'animation
 - RT Table
 - Ecrans d'exploitation
 - Documentation

Chart : [MAST - grafcset]

1 2

1 etape0

2 ce0_1

3 etape1

4 e2S1

5 etape2

6 e1S1

7 etape3

8 e1S0

9 etape4

10 e2S0

11

Editeur de données

Variables Types DDT Blocs fonction Types DFB

Filter

Nom	Type	Adresse	Va
ce0_1	BOOL		
e1S0	EBOOL	%I0.1	
e1S1	EBOOL	%I0.2	
e2S0	EBOOL	%I0.3	
e2S1	EBOOL	%I0.4	
m	EBOOL	%I0.10	
s1yv_12	EBOOL	%Q0.20	
s1yv_14	EBOOL	%Q0.21	
s2yv_12	EBOOL	%Q0.22	
s2yv_14	EBOOL	%Q0.23	

Editeur de d... Chart : [MA... actions_exte... ce0_1 <Tra...

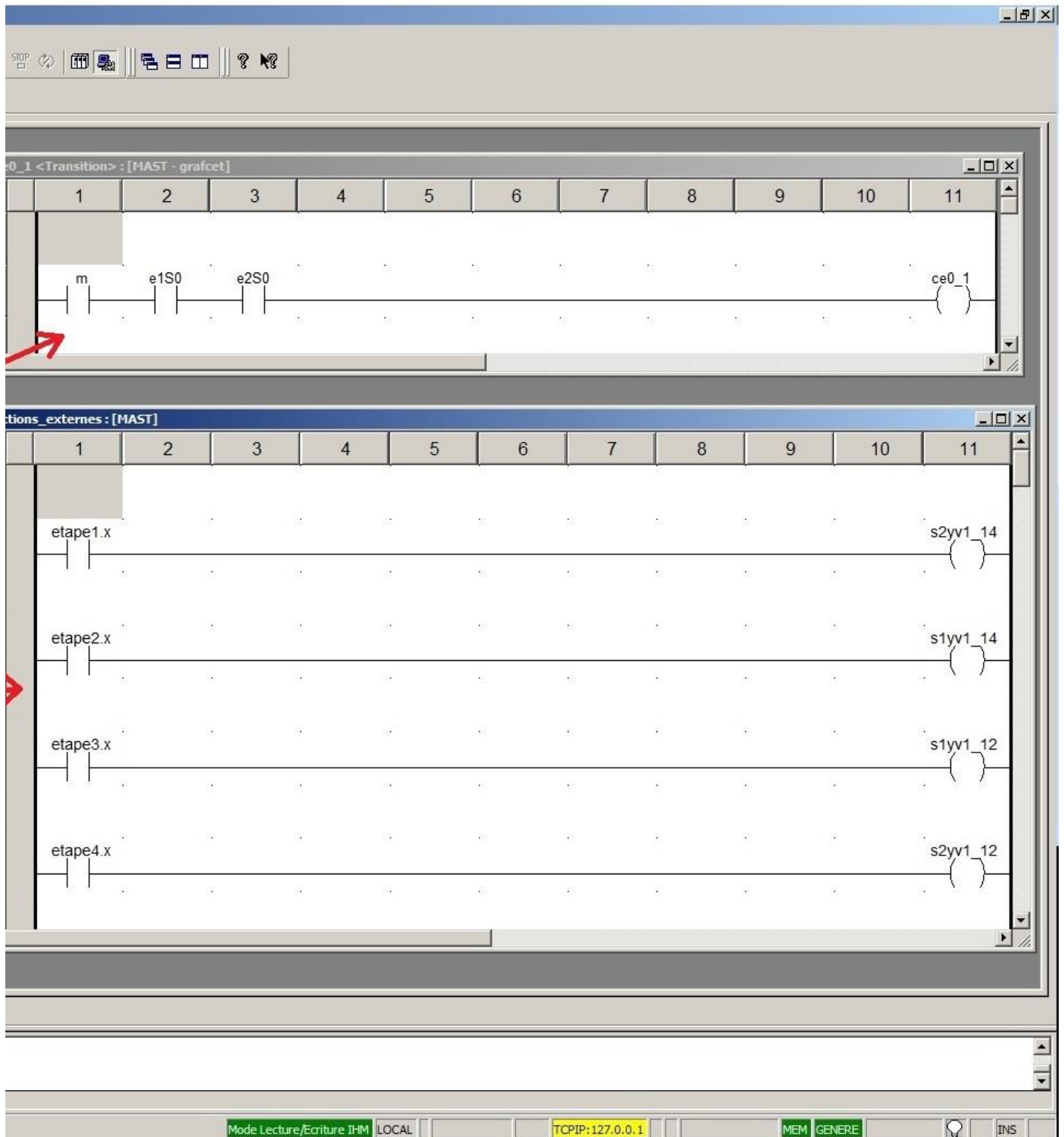
Edition des liens en cours...
Processus réussi : 0 Erreur(s) , 4 Avertissement(s)

Générer le projet Importer/exporter Erreur utilisateur Evénement d'historique FDT Rechercher/Remplacer

Prêt

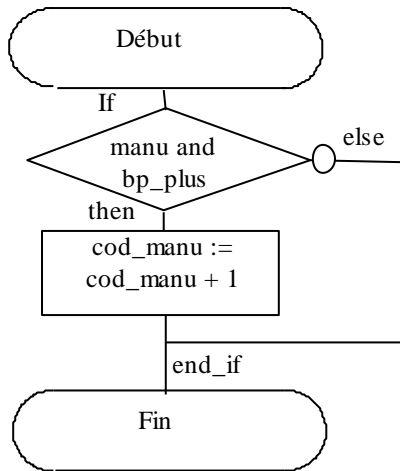
exemple : poinçonneuse

340 sous UNITY

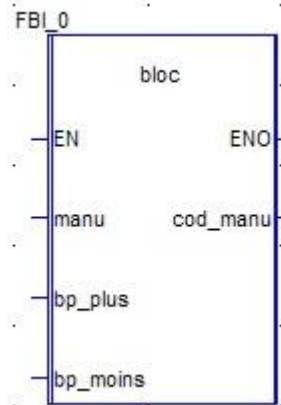


Annexe 1 : programmation d'un module logiciel dans un bloc fonctionnel en langage structuré

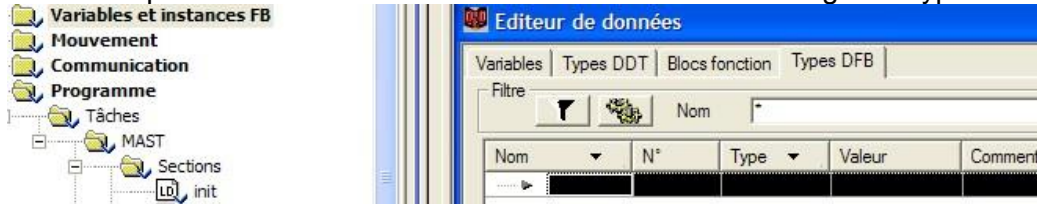
Exemple pour l'algorithme suivant :



Bloc fonctionnel



Double cliquez sur « Variables et Instances FB » et choisir l'onglet « Types DFB ».



Donnez un nom au bloc fonctionnel, aux variables d'entrées, de sorties et au module logiciel.

Nom	N°	Type	Valeur	Commentaire
[Icon] bloc				
[Folder] <entrées>				
[Icon] manu	1	EBOOL		
[Icon] bp_plus	2	EBOOL		
[Icon] bp_moins	3	EBOOL		
[Folder] <sorties>				
[Icon] cod_manu	1	INT		
[Folder] <entrées/sorties>				
[Folder] <public>				
[Folder] <privé>				
[Folder] <sections>				
[Icon] prog		<ST>		

Variables internes au bloc visibles dans une table d'animation

Variables internes au bloc invisibles dans une table d'animation

Le type EBOOL permet d'utiliser les fronts montants (ou descendants) dans le bloc.

Ecriture du module logiciel (section « prog » du bloc « bloc ») :

```

s1 prog <DFB> : [bloc]
  if manu and bp plus then cod manu:=cod manu+1;
  end_if;
    
```

Il faut ensuite instancier le bloc (c'est à dire réserver de la mémoire pour ce bloc dans l'automate)

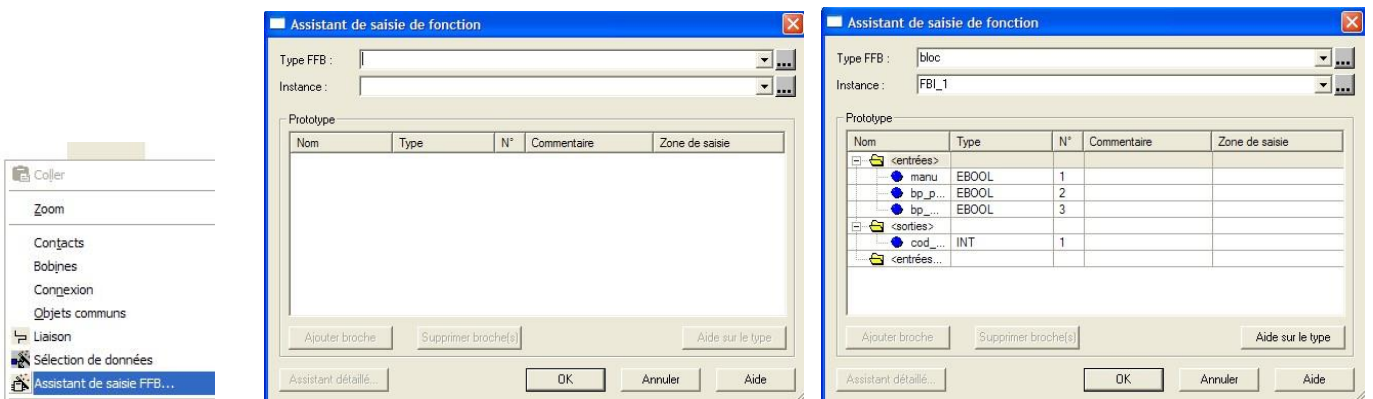


Le panneau travaux disparaît.

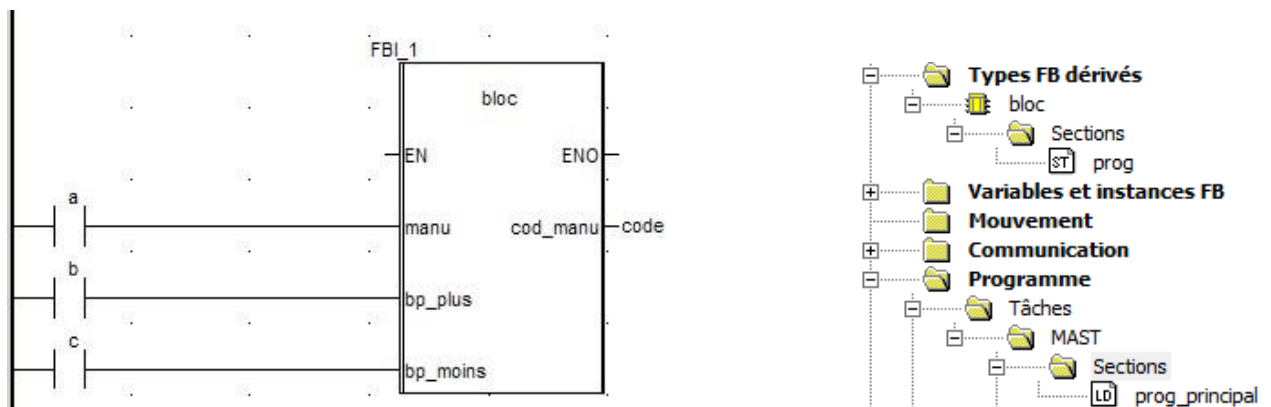
Il ne reste plus qu'à écrire le programme principal qui fera appel à l'instance de ce bloc.

Créez une section LADDER dans le programme principal :

Pour faire apparaître un bloc fonctionnel dans un programme LADDER, cliquez sur "Assistant de saisie FFB ».

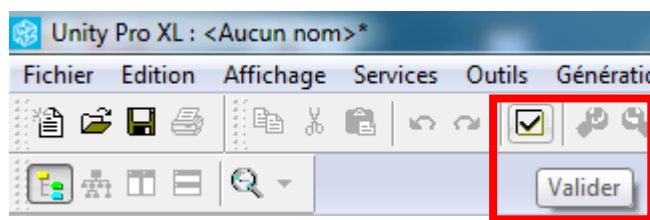
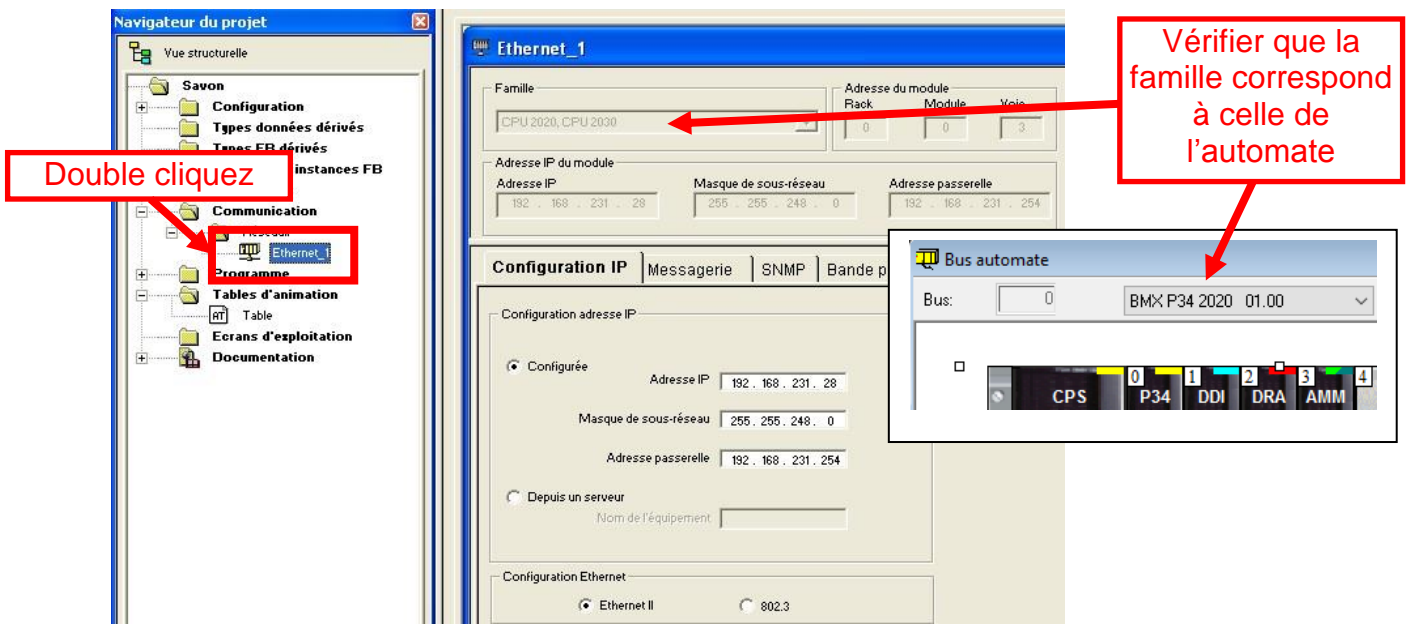
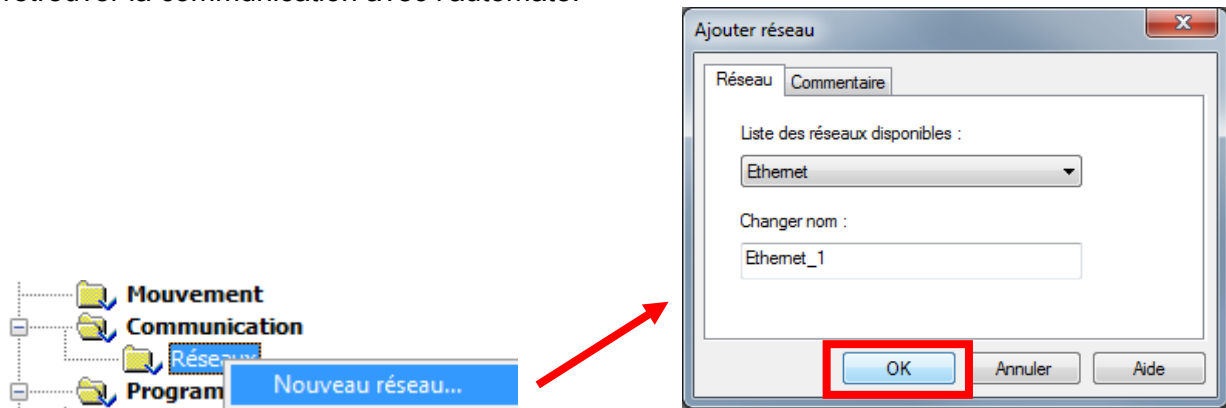


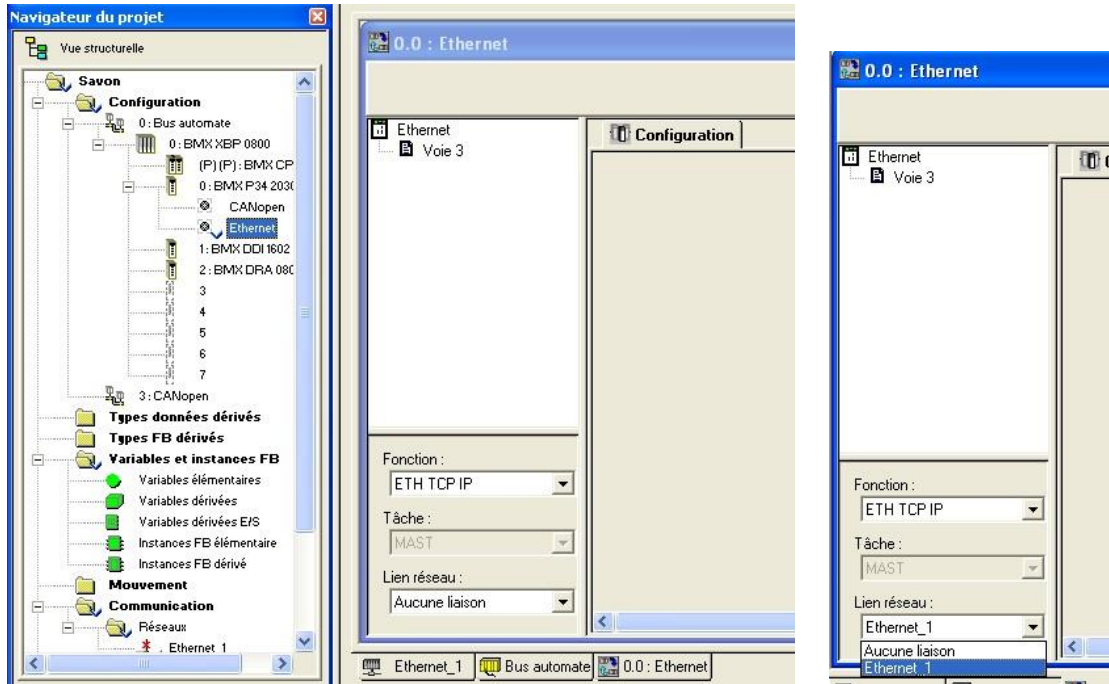
Programme principal :



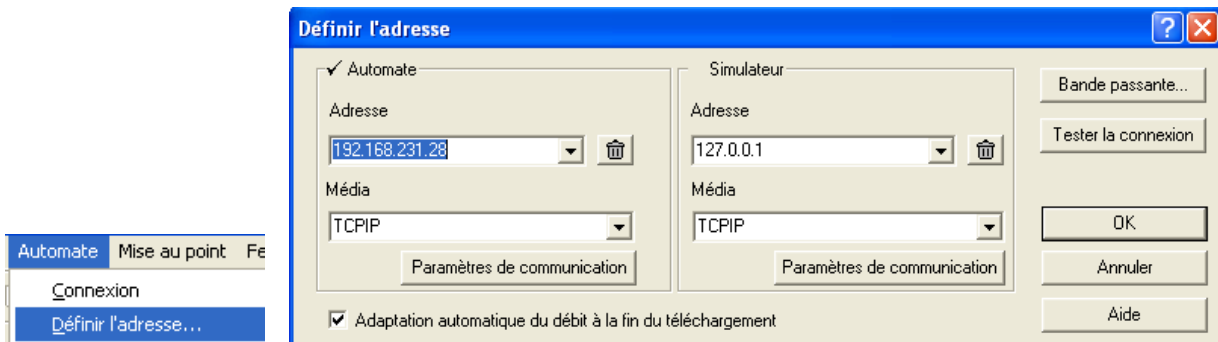
Annexe 2 : Configuration de l'interface Ethernet :

Ne pas configurer l'interface Ethernet désactive cette communication. Il faut repasser en USB pour retrouver la communication avec l'automate.



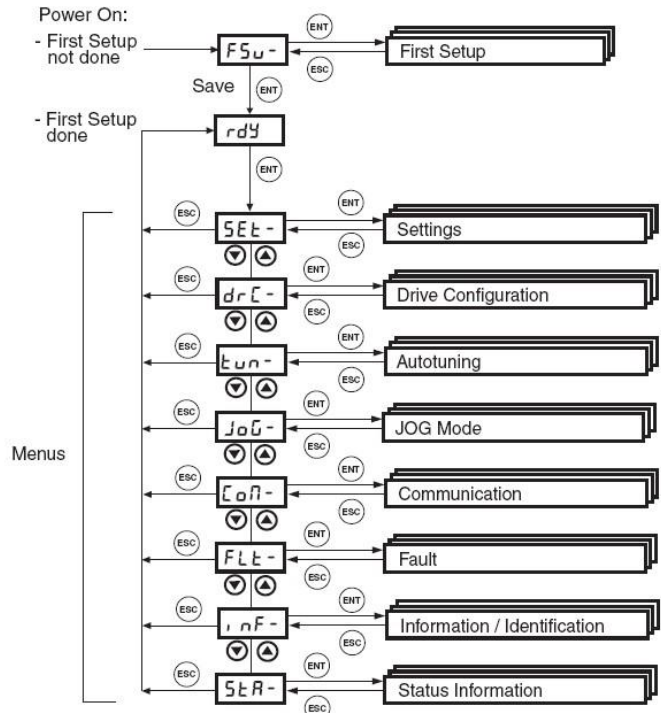
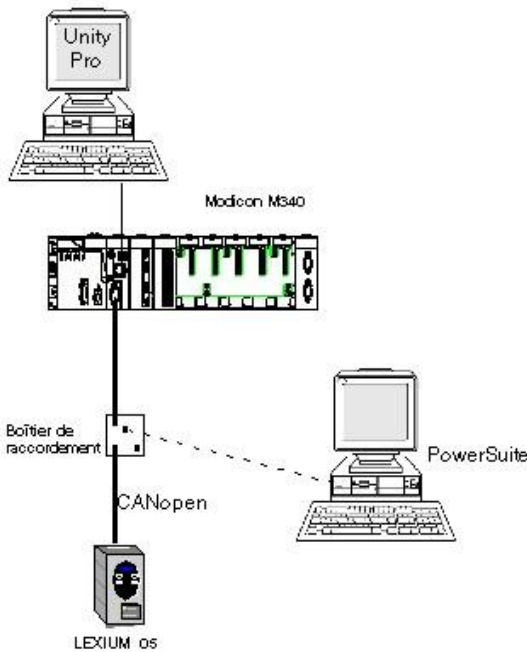


Connexion à l'automate par l'interface Ethernet :



Annexe 3 : Configuration d'un axe lexium sur bus CAN

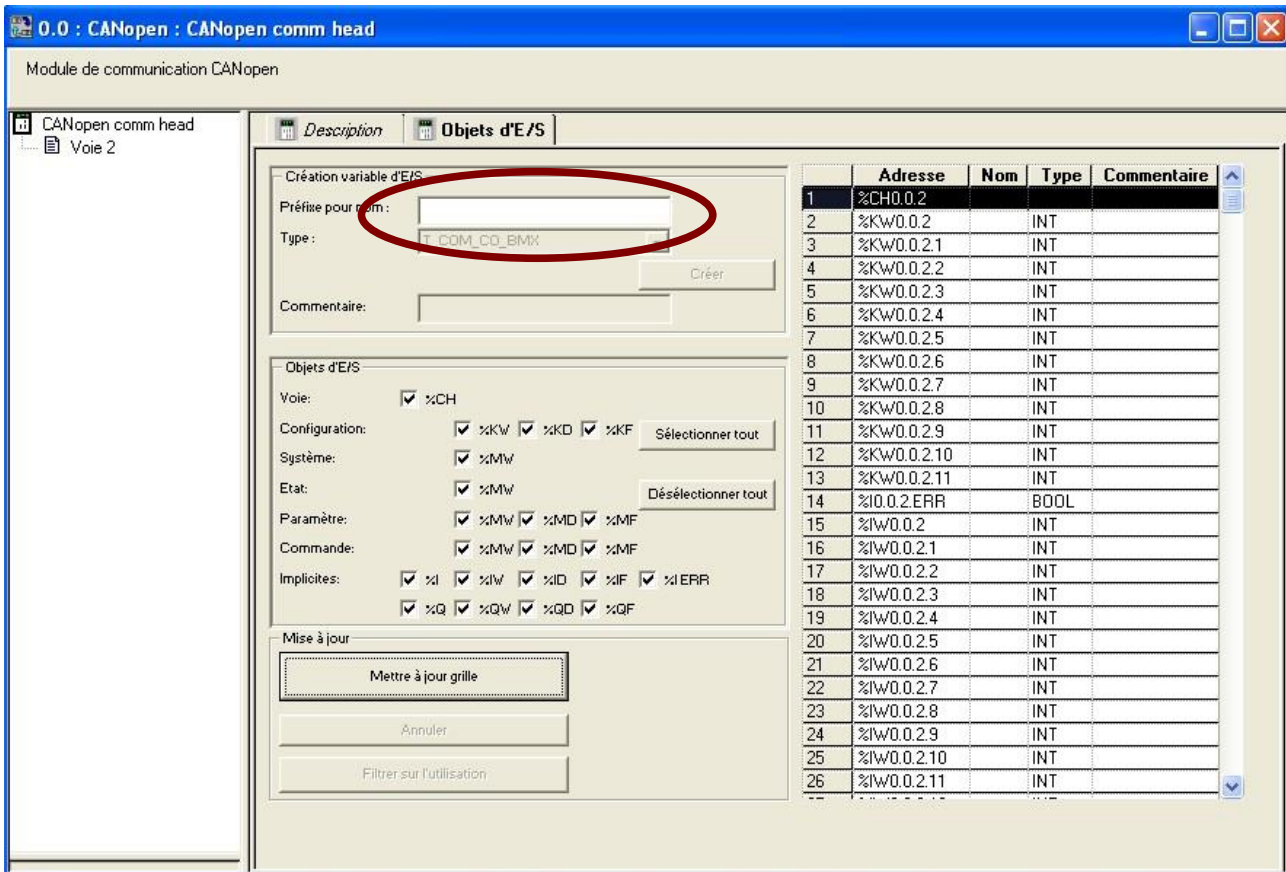
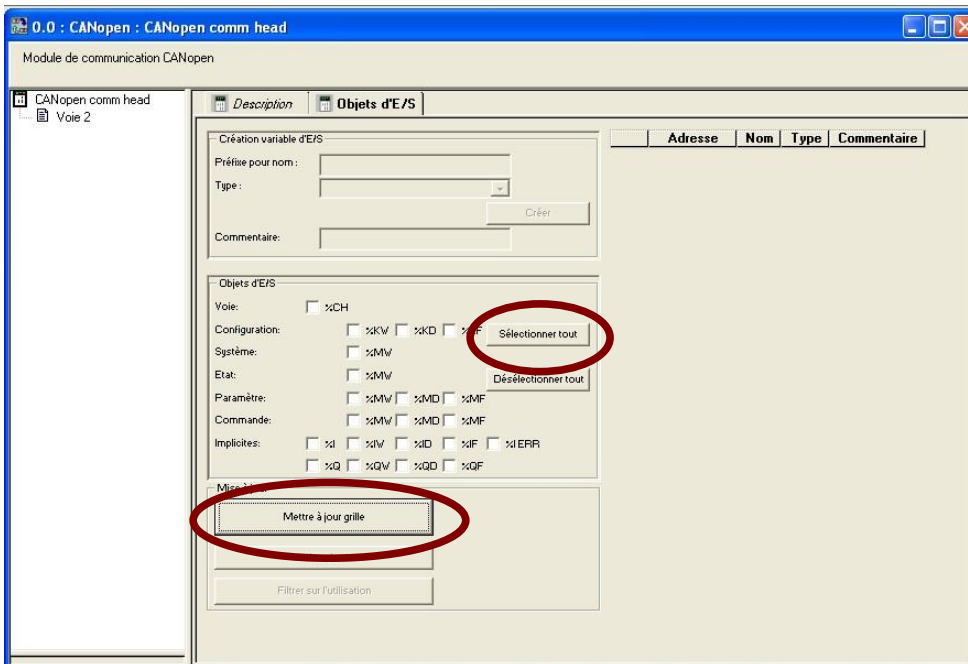
Configuration de la communication :



Dans le menu COM du lexium, configurez ADCO (Adresse CANopen) (ici 2) et BDCO (Baud CANopen)(ici 250).

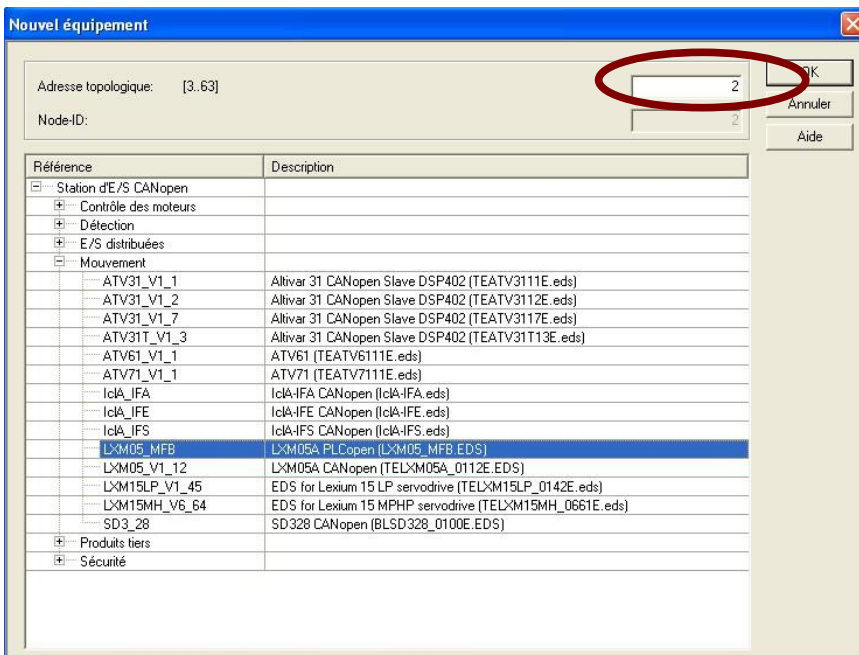
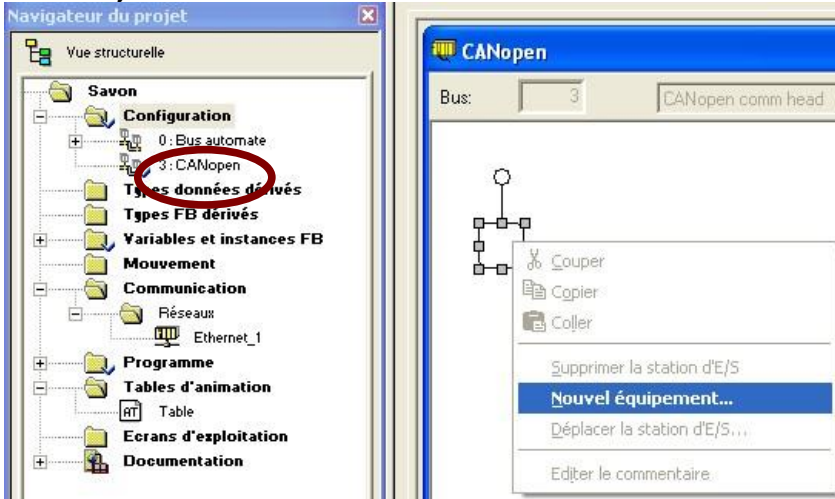
Dans Unity, configurez la vitesse du bus CAN de l'automate (ici 250 kBaud) :

Ensuite, on sélectionne toutes les variables du bus CAN et on leur donne un nom.

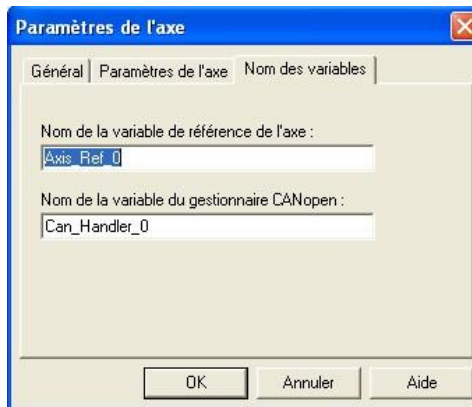
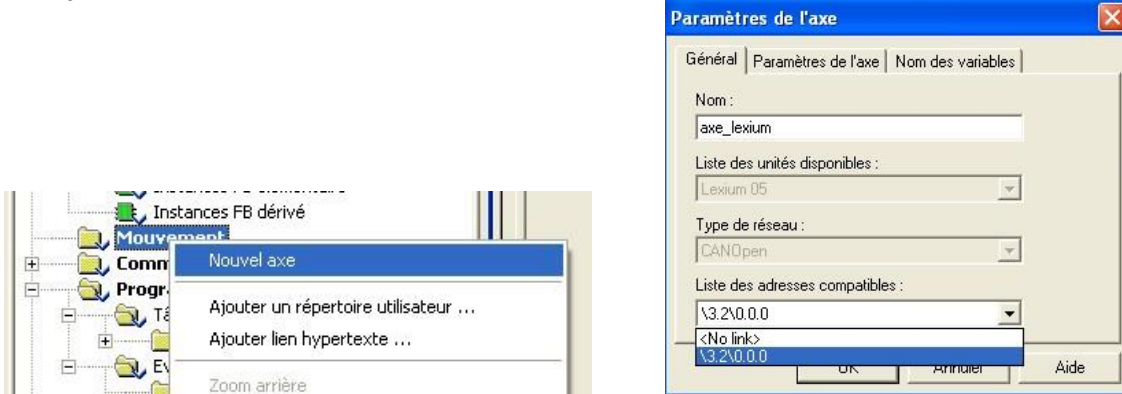


Sélectionner %CH0,0,2, tapez « bus_can » et cliquez sur « créer » pour que toutes les variables du bus CAN commencent pas « bus_can. ».

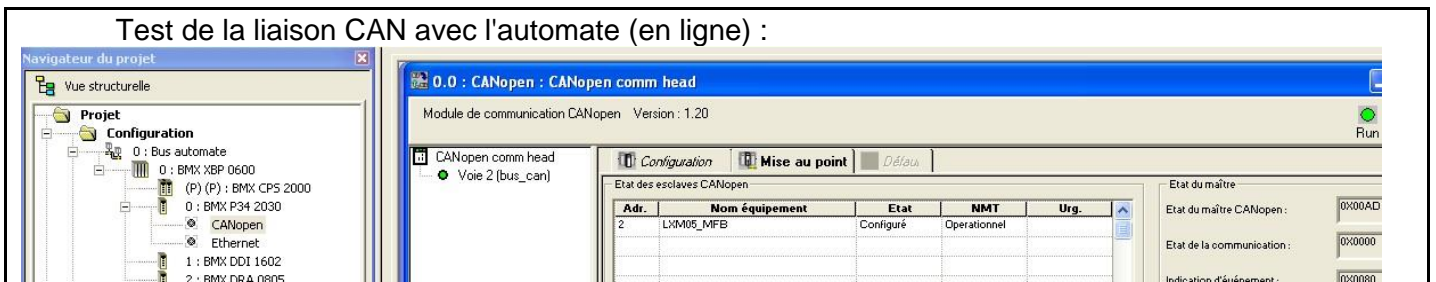
Ajout d'un LEXIUM 05A MFB sur le bus CAN à l'adresse 2



Ajout d'un nouvel axe :

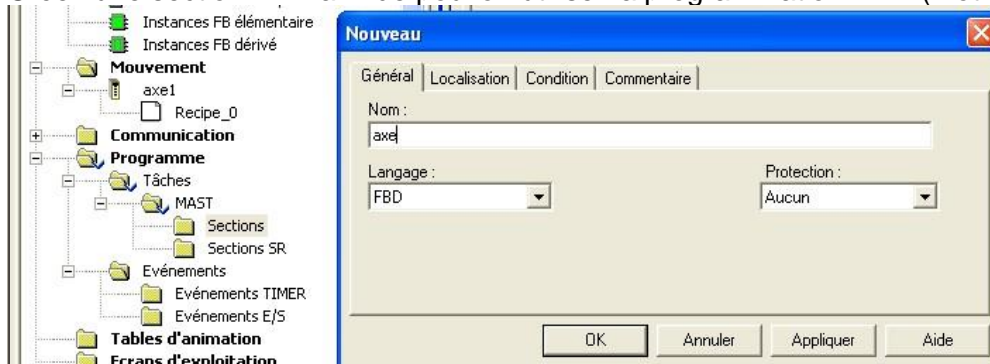


Test de la liaison CAN avec l'automate (en ligne) :



Programmation :

Créez une section FBD afin de pouvoir utiliser la programmation MFB (Motion Function Block)

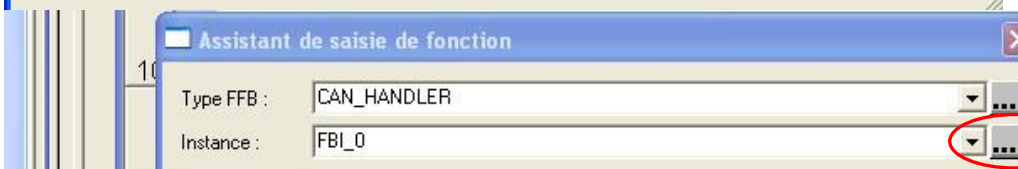
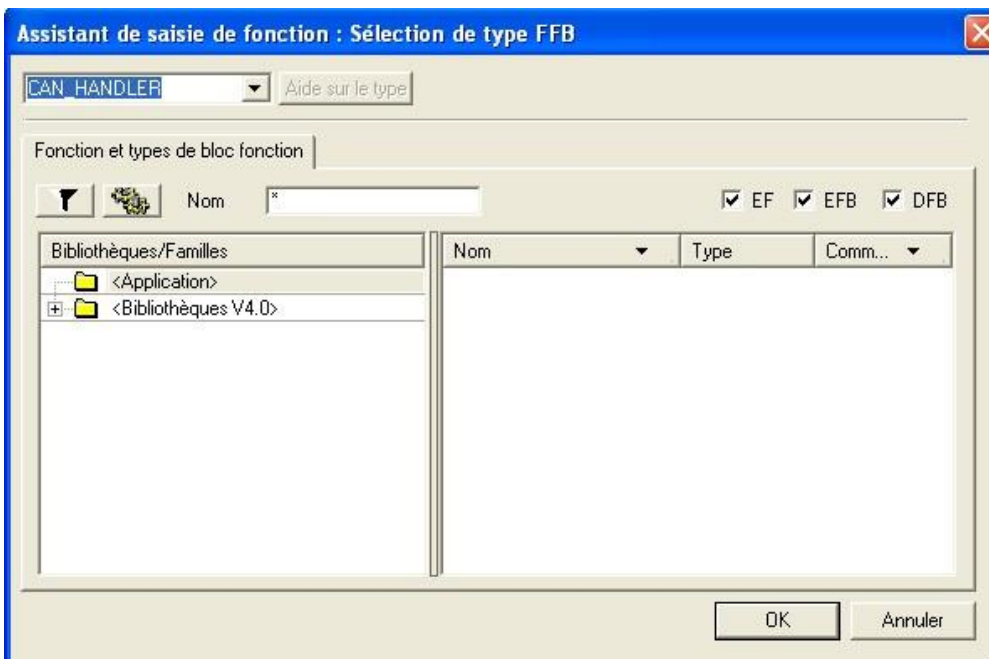
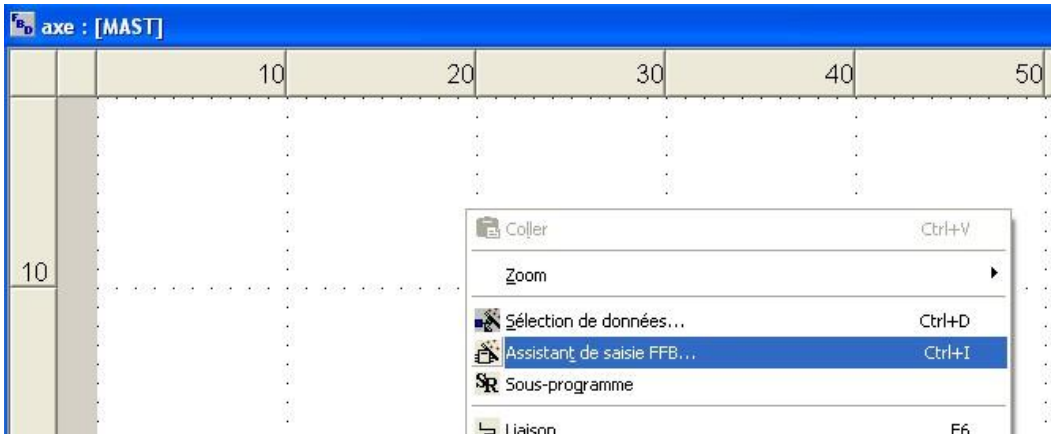


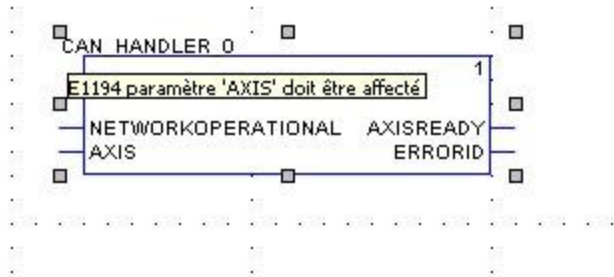
Le premier bloc à créer est le CAN_HANDLER.

L'utilisation du bloc fonction MFB CAN_HANDLER est primordiale et obligatoire dans la programmation de l'axe. La section de programme contenant ce MFB doit être associée à la même tâche que le maître du bus CANopen.

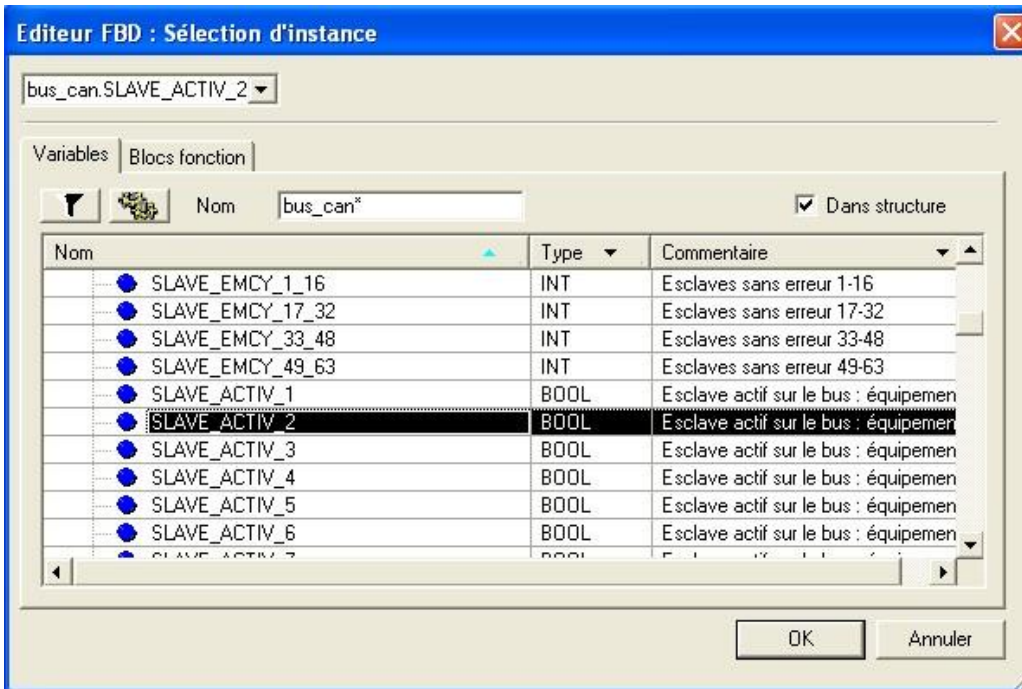
Il permet de vérifier la communication CANopen et la cohérence entre la configuration logicielle et l'équipement physique connecté.

Ce bloc utilise les deux variables appartenant au répertoire de l'axe. La variable Can_Handler_Z est à utiliser comme instance et la variable Axis_Ref_Z est à affecter au paramètre d'entrée AXIS du bloc.

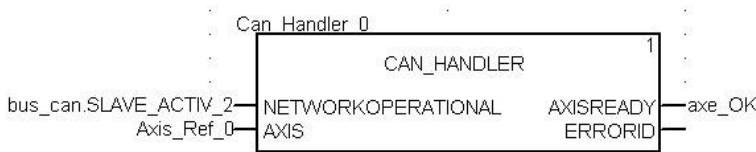
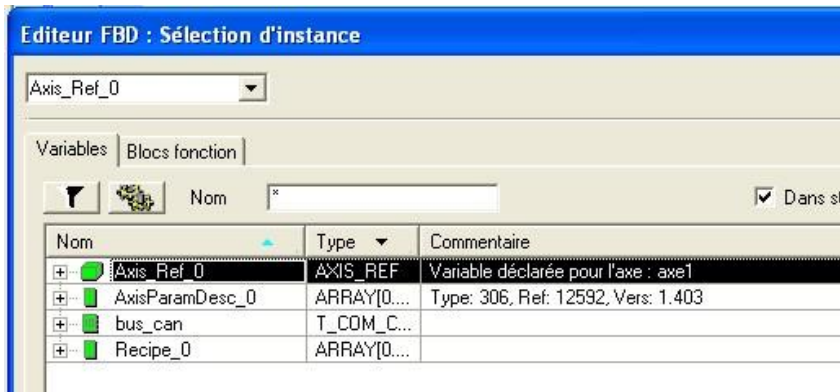




Entrée Networkoperational : Elle correspond à l'adresse de l'esclave (ici 2).

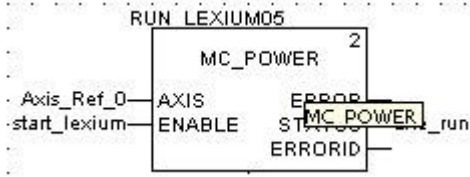


entrée AXIS : variable de référence de l'axe

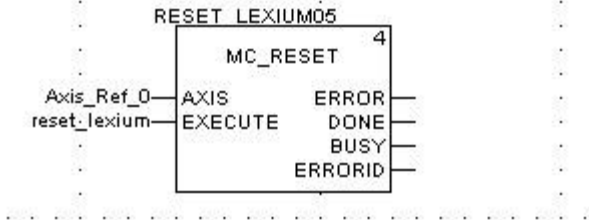
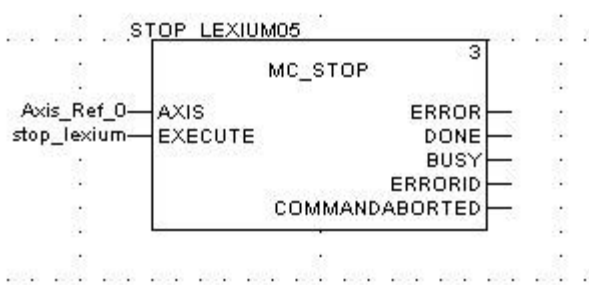


Exploitation de l'axe :

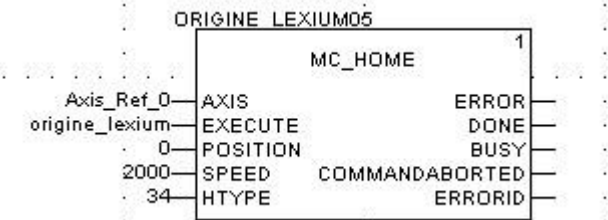
Mise en RUN du Lexium



Stop mouvement et Reset Lexium



Prise d'origine



Type de prise d'origine (extrait doc Lexium)

8.5.8.4 Course de référence sur l'impulsion d'indexation

Description Une course de référence sans impulsion d'indexation peut être réglée via le paramètre HMmethod = 33 et 34, voir page 8-52.

Course de référence sur l'impulsion d'indexation Les courses de référence sont représentées ci-après sur l'impulsion d'indexation (HMmethod = 33 et 34).

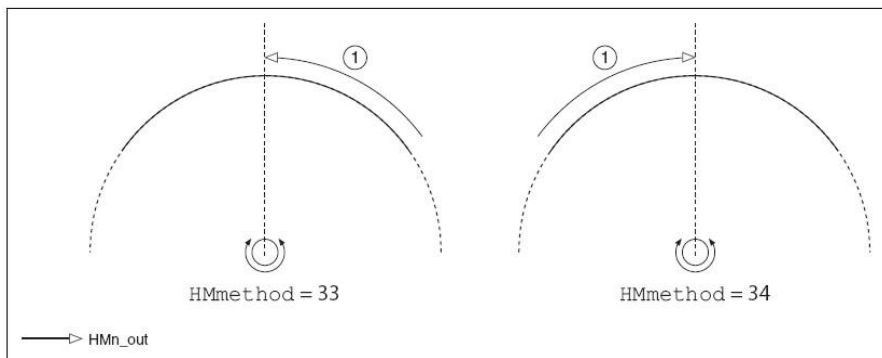


Illustration 8.36 Course de référence sur l'impulsion d'indexation

- (1) Course de référence sur l'impulsion d'indexation à la vitesse de retour en zone de positionnement

Parameter Name Menu HMI	Description	Unité Valeur minimale Valeur par défaut Valeur maximale	Type R/W persistant expert	Adresse de para- mètre par bus de terrain
HMmethod	Méthode de la course de référence8-51	-	INT8	CANopen 6098:0h
-	1 : LIMN avec impulsion d'indexation	1	INT16	Modbus 6936
-	2 : LIMP avec impulsion d'indexation	18	R/W	
-	7 : REF+ avec impulsion d'indexation, inv., dehors	35	-	
	8 : REF+ avec impulsion d'indexation, inv., dedans			
	9 : REF+ avec impulsion d'indexation, non inv., dedans			
	10 : REF+ avec impulsion d'indexation, non inv., dehors			
	11 : REF- avec impulsion d'indexation, inv., dehors			
	12 : REF- avec impulsion d'indexation, inv., dedans			
	13 : REF- avec impulsion d'indexation, non inv., dedans			
	14 : REF- avec impulsion d'indexation, non inv., dehors			
	17 : LIMN			
	18 : LIMP			
	23 : REF+, inv., dehors			
	24 : REF+, inv., dedans			
	25 : REF+, non inv., dedans			
	26 : REF+, non inv., dehors			
	27 : REF-, inv., dehors			
	28 : REF-, inv., dedans			
	29 : REF-, non inv., dedans			
	30 : REF-, non inv., dehors			
	33 : Impulsion d'indexation direction nég.			
	34 : Impulsion d'indexation direction pos.			
	35 : Définition des coordonnées			
	Signification des abréviations :			
	REF+ : Course de recherche dans la direc- tion pos.			
	REF- : Course de recherche dans la direc- tion nég.			
	inv. : Inverser la direction dans le commuta- teur			
	non inv. : Direction non inversée dans le commutateur.			
	dehors : impulsion d'indexation/distance hors du commutateur			
	dedans : impulsion d'indexation/distance dans le commutateur :			

Voir doc Lexium pour plus de précision sur toutes les méthodes d'indexation.

Fonctionnement :

« start_lexium » doit rester à 1 durant tout le fonctionnement : affichage de RUN sur le Lexium. Lorsqu'elle est à 0, affichage de RDY sur le LEXIUM.

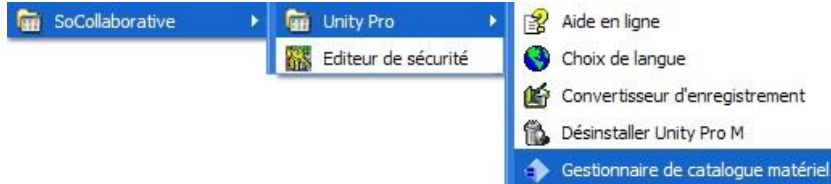
« stop_lexium » permet de stopper le mouvement en cours sur le lexium.

« reset_lexium » permet d'acquitter les défauts sur le lexium.

Annexe 4 : Ajout d'un nouveau modèle de composant tiers CAN open :

Un composant Can open est défini par son fichier EDS.

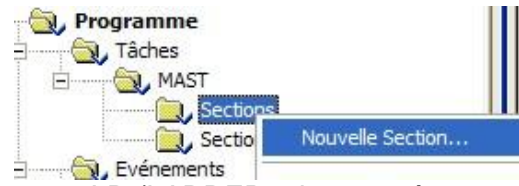
Ajout d'un fichier EDS sous UNITY :



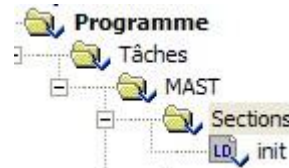
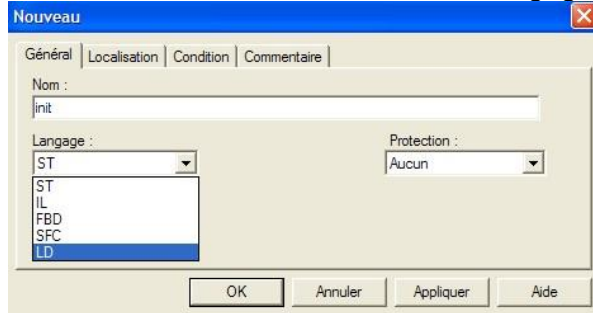
Choisir le fichier EDS concerné.

Annexe 5 : programmation en langage LD d'un grafcet

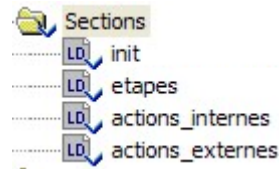
Double cliquez sur "tâche MAST". Cliquez droit sur « section » et choisissez "Créer".



Donnez le nom "init" et choisissez le langage LD (LADDER = langage à contacts)



Recommencez en créant les sections "étapes", "actions internes" et "actions externes".



Ecriture du programme dans une section :

Double cliquez sur « init ». Dessinez vos réseaux en utilisant les icônes en haut de l'écran. Vous programmez directement en mnémotechnique (sauf pour les variables systèmes).

Recommencez pour les autres sections.

Test du programme :

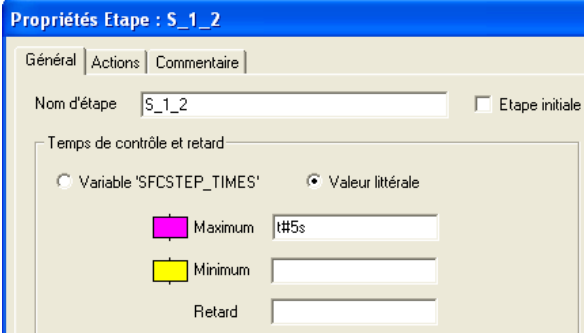
Vous devez créer une table d'animation en faisant un clic droit sur "table d'animation" et en cliquant sur "Nouvelle table d'animation".

Nom	Valeur	Type
● etape1	1	EBOOL
● etape2	0	EBOOL
● etape3	0	EBOOL
● e0	0	EBOOL
● e1	0	EBOOL

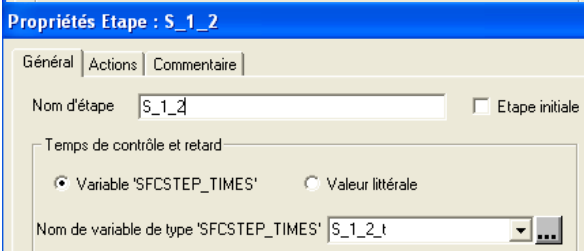
Nom	Valeur	Définit la valeur à 1
● etape1	1	EBOOL
● etape2	0	EBOOL
● etape3	0	EBOOL
● e0	0	EBOOL
● e1	0	EBOOL
● e2	0	EBOOL

Annexe 6 : Contrôle de la durée d'une étape (programmée en SFC) :

Le contrôle de la durée d'une étape permet de surveiller le bon fonctionnement d'une machine et de générer des défauts si ces temps sont trop importants.



Dans l'écran Propriétés, on peut définir le temps maximum d'une étape (ici 5s). Cette méthode est fastidieuse car lorsqu'on veut changer ce temps, il faut modifier la propriété de l'étape et générer à nouveau le programme.



Méthode recommandée :
 On définit une variable durée de l'étape S_1_2_t.
 Durée max de l'étape : S_1_2_t.max
 Durée min de l'étape : S_1_2_t.min
 Ces valeurs sont modifiables dans une table d'animation (pas besoin de régénération).

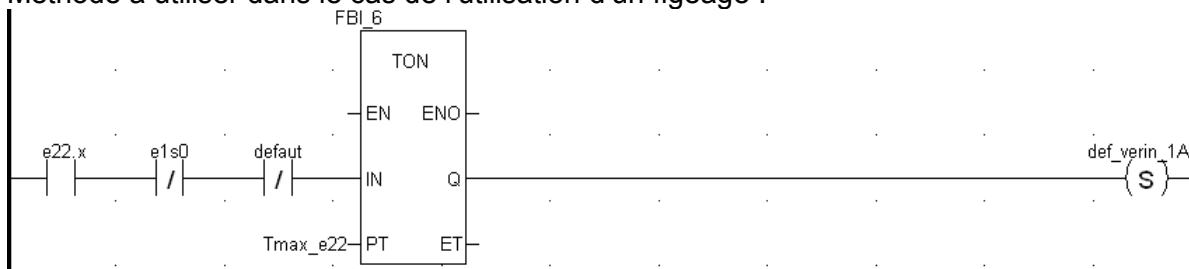
La variable S_1_2.tmaxErr passe à 1 lorsque la durée de l'étape est supérieure à la durée définie dans S_1_2_t.max.

Il suffit donc de tester la variable S_1_2.tmaxErr pour générer un défaut.

The screenshot shows a ladder logic chart on the left and a table of variables on the right. The chart has rungs 2 through 8. Rung 2 has a timer T#32s_200ms with event e1. Rung 3 has a timer T#8s_300ms with event e2. Rung 4 has a timer T#2s_900ms with event e3. Rung 5 has a timer T#56s_950ms with event e4. The table on the right lists variables: e1 (0, BOOL), e2 (0, BOOL), e3 (0, BOOL), e4 (0, BOOL), S_1_2_t.max (5s, TIME), S_1_2.tmaxErr (1, BOOL), default (1, BOOL), and reset_defaut (0, BOOL). Below the table is a 'defaults : [MAST]' window showing a ladder logic diagram for the default state, with a set coil (S) for 'defaut' and a reset coil (R) for 'reset_defaut'.

Attention : dans le cas d'un figeage, le temps continue à s'écouler. A partir d'un premier défaut, le système sera toujours en défaut.

Méthode à utiliser dans le cas de l'utilisation d'un figeage :



Fiche de programmation Unity.....	1
1.Ouverture d'un fichier existant :	1
2.Création d'une programmation :.....	1
3.Ecriture des mnémoniques :	1
4.Ecriture du programme :	1
5.Test du programme avec le simulateur intégré :.....	3
6.Configuration de l'automate :	5
7.Attribution d'adresses réelles :	5
8.Test réel :	6
9.Impression :	6
10.Sauvegarde et sortie :	7
Annexe 1 : programmation d'un bloc fonctionnel en langage structuré	10
Annexe 2 : Configuration de l'interface Ethernet :.....	12
Annexe 3 : Configuration d'un axe lexium sur bus CAN	14
Configuration de la communication :	14
Ajout d'un nouvel axe :	17
Programmation :	17
Exploitation de l'axe :.....	20
Annexe 4 : Ajout d'un nouveau modèle de composant tiers CAN open :	22
Annexe 5 : programmation en langage LD d'un grafcet.....	23
Annexe 6 : Contrôle de la durée d'une étape (programmée en SFC) :.....	24