REPUPLIQUE ALGERIENNE DEMOCRATIQUE ET POLPULAIRE

MINISTERE DE LœENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

ECOLE NATIONALE SUPERIEURE POLYTECHNIQUE

Département du Génie Electrique

Spécialité Automatique

Projet de fin dœétudes
En vue de lœobtention du diplôme dœingénieur dœétat en Automatique

Thème:

Gestion Et Supervision Dœune Station De Pompage à Base Dœautomate SCHNEIDER à Lœaide Des DFB

Etudié par :

Mr. DALI Ali
Mr. FIHAKHIR Amine Mahdi

Proposé et dirigé par :

Mr. A. BERKOUK
Mr.M.S.MAARADJI

Juin 2009

REMERCIEMENTS

Nous tenons à exprimer nos vifs remerciements à notre promoteur Pr.BERKOUK de l'Ecole Nationale Polytechnique pour nous avoir encadrer durant notre projet de fin d'études et nous conseillé tout le long de notre travail.

Nous remercions également notre co-promoteur Mr MAARAJI Samir, de SHNEIDER ELECTRIC, pour son encadrement, ces conseils, et sa confiance au sein de l'entreprise.

Nous remercions chaleureusement les membres du jury pour l'honneur qu'ils nous ont fait en acceptent d'évaluer notre projet.

Nos sincères remerciements aux ingénieurs de Schneider qui nous ont conseillé et éclairé sur notre travail tout le long de notre projet.

Nous souhaitons aussi remercier tous les enseignants de l'Ecole Nationale Polytechnique d'Alger, et en particulier, Nos professeurs d'Automatique qui nous ont encadrés auparavant et tous nos enseignants pour les connaissances qu'ils nous ont transmis, leur disponibilité et leurs efforts.

Nous remercions tout le personnel de l'école, de Schneider et tout les élèves de génie électrique.

Que tous ceux qui ont contribué de prés ou de loin à la réalisation de ce modeste travail trouvent ici l'expression de notre sincère gratitude.

Dédicaces

Je dédie ce modeste travail à mes chers parents qui ont fait de moi ce que je suis.

A mon frère Mohamed pour qui j'ai toujours tenu à donner le meilleur de moi-même.

A ma grand-mère, qui m'a toujours soutenu..

A mon binôme Ali pour ces deux années de travail pleines de souvenirs, ainsi qu'a

toute sa famille que je remercie pour son hospitalité et sa gentillesse.

A mes amis Dahmane ,Abdelghani ,Ahmed ,Mohamed ,Redwane ,Messaoud ,

Mahmoud ,Yazid,Brahim qui ont toujours été là pour moi

A mes amis et camarades de l'Ecole Nationale Polytechnique, et toute la promotion

Automatique de l'année 2009.

Aux anciens Automaticiens qui m'ont servis d'exemples

A tous ceux qui me sont chers, et qui me portent dans leurs cœurs.

FIHAKHIR Amine

Dédicaces

Je commence d'abord par remercier mon binôme « Amine FIHAKHIR ».

je dédie ce présent travail à ma mère et mon père qui m'ont soutenu et aider tout le long de mon parcourt et sans qui je ne serais pas ou j'en suis. À mes deux frères Hamid et Ayman.

À toute l'équipe Schneider Electric , surtout Samir , Ali, Djamel et Abderhman.

À toute la promotion 2009 , tout particulièrement Amine, Hakim et Badro.

À tout mes amis de l'Ecole Nationale Polytechnique avec qui j'ai eu d'agréable moments et apprit beaucoup de choses.

À tous mes amis que je n'ai pas cité et qui sont présent dans mes pensées.

DALI ALI

SOMMAIRE

CHAPITRE II Le langage de programmation Unity Pro

CHAPITRE III Description de la station de pompage

CHAPITRE IV Création des DFB

CHAPITRE V Le logiciel de construction des HMI Vijeo Designer

LISTE DES FIGURES

LISTE DES TABLEAUX

Symboles et abréviation

AC alternative current (courant alternatif)

API Automate Programmable industriel

DC direct current (courant continu)

DFB bloc fonction dérivé

CPU Central Processing Unit

EEPROM Electrically Erasable Programmable Read Only Memory

FAST tâche rapide

FBD langage bloc fonction

GEP groupe électropompe

IHM Interface Homme/Machine

LI langage liste instructions

LD langage à contacts LADDER

MAST tâche maître

PID régulateur proportionnel intégral dérivé

RAM Random Access Memory

ROM Read Only Memory

ST langage structuré

TOR tout ou rien

Introduction Générale

La compétitivité des entreprises impose un recours à la fois fréquent et intensif à des technologies de production avancées. La productique et la complexité des opérations à exécuter, conduisent à la mise en œuvre de dispositifs et systèmes pour l'automatisation des ateliers de fabrication ou de production.

L'automate programmable industriel A.P.I est aujourd'hui le constituant le plus répandu pour réaliser des automatismes. On le trouve pratiquement dans tous les secteurs de l'industrie car il répond à des besoins d'adaptation et de flexibilité pour un grand nombre d'opérations. Cette émergence est due en grande partie, à la puissance de son environnement de développement et aux larges possibilités d'interconnexions.

Le domaine de l'hydraulique parmi d'autres, est témoin de cette révolution, et de nombreuses sociétés algériennes (ANBT, SEEAL, Hydro-Aménagement, …etc.), orientées dans le stockage, l'épuration et la distribution de l'eau, cherchent à se procurer cette solution d'automatisme au niveau de leurs stations de pompages et de traitements. Schneider Electric est une firme compétitive, procurant ce type de service industriel.

Dans le but d'amélioration des applications (programmes) de gestion des stations de pompage Schneider Electric nous a proposé la réalisation des DFB (blocs fonctions dérivées) permettent de structurer et d'optimiser ces applications. Dans ce cadre : cinq DFB ainsi qu'un programme final vont être mises en place englobant l'essentiel des systèmes que nous puissions trouver dans une station de pompage réelle. Cette solution sera à base d'automate Schneider : Modicon M340 Ethernet.

Plan du travail

Le travail est présenté en cinq chapitres complémentaires :

Le premier chapitre traite les automates programmables : après une vue générale sur les API on abordera une description brève de toute la gamme des API Schneider Electric.

Le deuxième chapitre présente le logiciel de programmation Unity Pro : une présentation du logiciel (versions, fonctions, langages de programmations….) suivie d'une description de la méthode de configuration et on termine par un exemple de mise en œuvre d'une application.

Dans le troisième chapitre. On décrit la station de pompage : les différents équipements ainsi que l'Algorithme de process .Le quatrième est consacré pour la création des blocs DFB de notre application, pour chaque DFB on donne son cahier des charge, ce signaux, sa section de programmation, le programme de simulation et en fin l'écran d'exploitation.

Le cinquième chapitre présente le logiciel de construction des HMI Vejio designer : après une présentation générale on décrit les différents éléments permettant la mise en place de notre application qui sera traité à la fin de ce chapitre.

On termine par une conclusion générale et on mentionne les différentes perspectives.

Chapitre I

Les automates programmables Industriels SCHNEIDER

I.1 Historique

C'est Modicon qui créa en 1968, aux USA, le premier automate programmable. Son succès donna naissance à une industrie mondiale qui s'est considérablement développée depuis. L'automate programmable représente aujourd'hui l'intelligence des machines et des procédés automatisés de l'industrie, des infrastructures et du bâtiment.

Histoire des API SCHNEIDER-TELEMECANIQUE

1975 : Naissance du premier automate programmable TSP 100.

1978 : Mise sur le marché de la première génération d'automates avec le modèle TSX 80.

1980 : Poursuite du lancement automates avec le modèle TSX 60.

1984-1985 : Lancement de la deuxième génération d'automates, série 7, avec les modèles TSX 47, 67 et 87. Apparition du premier réseau local industriel Telway.

1986-1987 : Développement du terminal de dialogue homme-machine XBT avec écran et clavier.

1988 : Avec 14 500 personnes, 32 filiales hors de France et un chiffre d'affaires de 8 milliards de FF, Telemecanique entre dans le Groupe Schneider. Lancement du micro automate TSX 17.

1990-1992 : Commercialisation d'une nouvelle gamme de contacteurs série D, de l'atelier logiciel X-TEL pour la programmation des automates, des portables industriels FTX 417/507 compatibles PC.

1993-1995 : Fusion de Telemecanique et de Merlin Gerin dans Schneider Electric SA en 1994.

Développement du nano automate TSX 07.

Intégration de MODICON au sein du groupe SCHNEIDER en 1994.

1996-1999 : Lancements du micro automate TSX 37 et de la plate-forme d'automatisme Premium.

2000-2001 : Commercialisation du module programmable Zelio Logic.

2002-2003 : Lancement de nouvelles gammes de produits de contrôle industriel et d'automatisme embarquant de plus en plus d'intelligence, simples à mettre en î uvre et ouverts aux standards de communication du marché. Sa devise est « Simply smart », qui se traduit par « davantage d'ingéniosité et d'intelligence pour une utilisation toujours plus simple ». Parmi ces gammes : Global Detection (détecteurs photoélectriques et inductifs, interrupteurs de position et capteurs de pression), TeSys

modèle U, premier contrôleur démarreur de marché, Altivar 11, Altistart 48, Twido (nano automate), Magelis IPC (PC industriels), Vijeo Look (logiciel de supervison), etc.

En avril 2003 la marque Modicon, devient un nom de gamme de produits de la marque Telemecanique [1].

I.2 Définition des automates programmables industriels (API)

"Appareil électronique qui comporte une mémoire programmable par un utilisateur automaticien (et non informaticien) à l'aide d'un langage adapté, pour le stockage interne des instructions composant les fonctions d'automatisme comme par exemple :

- Logique séquentielle et combinatoire

- Temporisation, comptage, décomptage et comparaison

- Calcul arithmétique

- Réglage, asservissement, régulation, etc. Pour commander, mesurer et contrôler au moyen d'entrées et de sorties (logiques, numériques ou analogiques) différentes sortes de machines ou de processus, en environnement industriel." [2]

La force principale dôun automate programmable industriel API réside dans sa grande capacité de communication avec lôenvironnement industriel. Outre son unité centrale et son alimentation, il est constitué essentiellement de modules dôentrées/sorties, qui lui servent dôinterface de communication avec le processus industriel de conduite.

Et il a comme rôles principaux dans un processus :

ÅDôassurer lôacquisition de lôinformation fournie par les capteurs ;

ÅEn faire le traitement ;

ÅElaborer la commande des actionneurs ;

ÅAssurer également la communication pour lôéchange dôinformations avec lôenvironnement. [3]

I.3 Architecture des automates programmables industriels

De forme compacte ou modulaire, les automates sont organisés suivant lôarchitecture suivante :

Å Un module dôunité centrale ou CPU, qui assure le traitement de lôinformation et la gestion de lôensemble des unités. Ce module comporte un microprocesseur, des circuits périphériques de gestion des entrées/sorties, des mémoires RAM et EEPROM nécessaire pour stocker les programmes, les données, et les paramètres de configuration du système.

ÅUn module dôalimentation qui, à partir dôune tension 220V/50Hz ou dans certains cas de 24V fournit les tensions continues + /- 5V, +/-12V ou +/-15V.

ÅUn ou plusieurs modules dôentrées ôTout ou Rienô ou analogiques pour lôacquisition des informations provenant de la partie opérative (procédé à conduire).

ÅUn ou plusieurs modules de sorties ôTout ou Rienô (TOR) ou analogiques pour transmettre à la partie opérative les signaux de commande. Il y a des modules qui intègrent en même temps des entrées et des sorties [4].

Alimentation CPU Modules dôentrées/sorties

Figure I.1 *Plate-forme Modicon (API Schneider Electric)* [5]

I.4 Structure interne des automates programmables

La structure matérielle interne dôun API obéit au schéma donné sur la figure ci dessous.

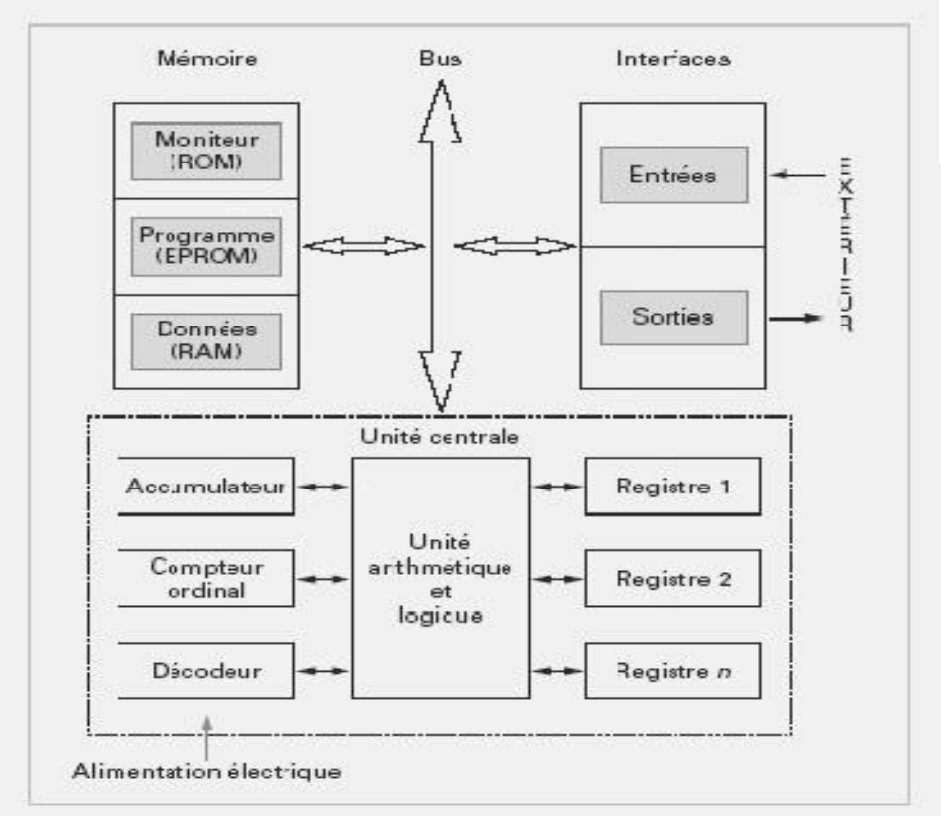


Figure I.2 *Structure interne dôun API* [3]

Détaillons successivement chacun des composants qui apparaissent sur ce schéma.

I.4.1 Le processeur

Il Constitue le cî ur de lôappareil dans lôunité centrale ; En fait, un processeur devant être automatisé, se subdivise en une multitude de domaine et processeur partiels plus petits, liés les uns aux autres.

I.4.2 Les modules dôentrées/sorties

Ils assurent le rôle dôinterface entre la CPU et le processus, en récupérant les informations sur lôétat de ce dernier et en coordonnant les actions.

Plusieurs types de modules sont disponibles sur le marché selon lôutilisation souhaitée :

ÅModules TOR : l'information traitée ne peut prendre que deux états (vrai/faux, 0 ou 1 é)

C'est le type d'information délivrée par une cellule photoélectrique, un bouton poussoir é etc.

ÅModules analogiques : l'information traitée est continue et prend une valeur qui évolue dans une plage bien déterminée. C'est le type d'information délivrée par un capteur (débit, niveau, pression, températureé etc.).

ÅModules spécialisés : l'information traitée est contenue dans des mots codés sous forme binaire ou bien hexadécimale. C'est le type d'information délivrée par un ordinateur ou un module intelligent.

I.4.3 Les mémoires

Un système de processeur est accompagné par un ou plusieurs types de mémoires. Elles permettent :

ÅDe stocker le système d'exploitation dans des ROM ou PROM,

ÅLe programme dans des EEPROM,

ÅLes données système lors du fonctionnement dans des RAM. Cette dernière est généralement secourue par pile ou batterie. On peut, en règle générale, augmenter la capacité mémoire par adjonction de barrettes mémoires type PCMCIA.

I.4.4 Lôalimentation

Elle assure la distribution d'énergie aux différents modules. L'automate est alimenté généralement par le réseau monophasé 230V-50 Hz mais d'autres alimentations sont possibles (110V é etc.).

I.4.5 Liaisons de communication

Elles Permettent la communication de l'ensemble des blocs de l'automate et des éventuelles extensions. Les liaisons sôeffectuent :

Åavec lôextérieur par des borniers sur lesquels arrivent des câbles transportant le signal électrique ;

Åavec lôintérieur par des bus reliant divers éléments, afin de dôéchanger des données, des états et des adresses. [3]

I.5 Présentation de lôoffre Schneider Electric

Schneider Electric propose une gamme dôautomates et de produit dôautomatisation complète par le biais de sa filiale Telemecanique. Les automates mis sur le marché sont :

I.5.1 Modules programmables Zelio Logic

I.5.1.1 Présentation

Figure I.3 *module Zelio Logic* [6]

Les modules logiques Zelio Logic sont destinés à la réalisation de petits équipements d'automatismes. Ils sont utilisés dans les secteurs d'activité de l'industrie et du tertiaire. [6]

Pour lôindustrie :

- automatismes de petites machines de finition, de confection, dôassemblage ou dôemballage
- automatismes décentralisés sur les annexes de grosses et moyennes machines dans les domaines du textile, du plastique, de la transformation de matériaux
- automatismes pour machines agricoles (irrigation, pompage, serre...).

Pour le tertiaire/bâtiment :

- automatismes de barrières, de volets roulants, de contrôle dôaccès
- automatismes dôéclairage
- automatismes de compresseurs et de climatisation

I.5.1.2 Modules compacts et modulaires

I.5.1.2.1 Modules logiques compactes

Les modules logiques compacts répondent aux besoins dôautomatismes simples.

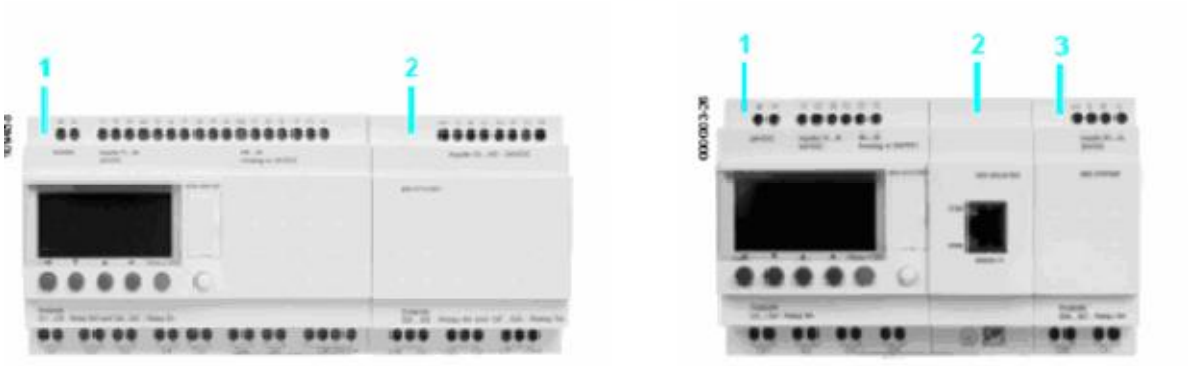
Les entrées/sorties sont au nombre de :

- 12 ou 20 E/S, alimentées en 24V AC ou 12V DC.
- 10, 12 ou 20 E/S, alimentées en 100é 240V AC ou 24V DC.

I.5.1.2.2 Modules logiques modulaires et extensions

Les entrées/sorties pour les modules logiques modulaires sont au nombre de :

- 26 E/S, alimentées en 12V DC,
- 10 ou 26 E/S, alimentées en 24V AC, 100é 240V AC ou 24V DC



1 Zelio logic Modulaire (10 ou 26 ES).

2 Extension E/S TOR (6,10 ou 14 E/S) ou analogiques (4 E/S).

1 Zelio logic Modulaire (10 ou 26 ES).

2 Extension de communication réseau Modbus ou Ethernet.

3 Extension E/S TOR (6,10 ou 14 E/S) ou analogiques (4 E/S)

Figure I.4 : *Module Zelio logic Modulaire* [3]

Pour plus de performance et de flexibilité, les modules Zelio Logic modulaires peuvent recevoir des extensions afin dôobtenir un maximum de 40 E/S :

- Extensions de communication réseau Modbus ou Ethernet, alimentées en 24V DC par le module Zelio Logic de même tension.
- Extension dôentrées/sorties analogiques avec 4 E/S, alimentées en 24V DC par le module Zelio Logic de même tension,
- Extensions dôentrées/sorties TOR avec 6, 10, ou 14 E/S, alimentées par le module Zelio Logic de même

I.5.1.3 Programmation

La simplicité de leur programmation, garantie par l'universalité des langages, satisfait aux exigences de l'automaticien et répond aux attentes de l'électricien. La programmation peut être effectuée :

- De façon autonome en utilisant le clavier du module Zelio Logic (langage à contacts),
- Sur PC avec le logiciel ñZelio Soft 2ò.

Sur PC, la programmation peut être réalisée soit en langage à contacts (LADDER), soit en langage blocs fonctions (FBD).Le rétro éclairage de lôafficheur LCD se fait par lôactivation de lôune des 6 touches de programmation du module Zelio Logic ou par programmation à lôaide du logiciel ñZelio Soft 2ò (exemple : clignotement lors dôun dysfonctionnement). Lôautonomie de lôhorloge, assurée par une pile lithium, est de 10 ans.

La sauvegarde des données (valeurs de présélection et valeurs courantes) est garantie par une mémoire Flash EEPROM (10 ans).

I.5.2 Contrôleurs programmables Twido

I.5.2.1 Bases compactes

La gamme des contrôleurs programmables compacts Twido offre une solution "tout en- un" dans un encombrement réduit de : 80 à 157 x 90 x 70 mm. Huit contrôleurs compacts sont disponibles, différents par leur capacité de traitement et leur nombre d'entrées 24V DC, de sorties à relais et à transistor (10, 16, 24 et 40 entrées/sorties). [6]

Figure I.5 : *TWD LDA 10 DRF* [6]

Les options afficheur et mémoire enfichables sur la base facilitent les opérations de réglage, de transfert et de sauvegarde des applications :

- Lôafficheur numérique peut être utilisé comme un outil de visualisation et de réglage local,
- La technologie EEPROM des cartouches mémoire permet les opérations de sauvegarde et de transfert de programme vers tout contrôleur compact ou modulaire Twido.

Les contrôleurs compacts possèdent 2 points de réglage analogique (un seul pour les bases 10 et 16 entrées/sorties) accessibles en face avant.

Base compacte	**Entrées 24 V DC**	**Sorties relais**	**Réglage Analogiques**	**Ports série**	**Expansion dôentrées/sorties**	**Module afficheur**	**Cartouche optionnelle**
TWD LCǒA 10DRF	6	4	1 point de 0é 1023	1×RS 485	NON	OUI	1 emplacement horodateur ou mémoire
TWD LCǒA 16DRF	9	7	1 point de 0é 1023	1×RS 485 en option 1×RS 232C /485	NON	OUI	1 emplacement horodateur ou mémoire
TWD LCǒA 24DRF	14	10	1 point de 0é 1023 1 point de 0é 511	1×RS 485 en option 1×RS 232C /485	OUI, 4 MAXI	OUI	1 emplacement horodateur ou mémoire
TWD LCǒA 40DRF	24	14+2 sorties à transistor source	1 point de 0é 1023 1 point de 0é 511	1×RS 485 en option 1×RS 232C /485	OUI, 4 MAXI	OUI	1 emplacement horodateur ou mémoire

Tableau I.1 : *Caractéristiques des différents modules Twido compact* [6]

I.5.2.2 Bases Modulaires

Lôoffre des contrôleurs programmables modulaires propose cinq bases, différentes par leur capacité de traitement et leurs nombre et type d'entrées/sorties (20 ou 40 E/S à raccordement par borniers à vis ou connecteur type HE 10, à sorties relais ou à transistor sink/source). Elles peuvent recevoir en expansion tous les modules d'entrées/sorties (18 modules TOR et analogiques). Toutes les bases modulaires utilisent une alimentation 24V DC. Ces bases modulaires offrent :

- Une modularité sôadaptant aux besoins de lôapplication à partir de base pouvant recevoir jusquôà 4 ou 7 modules d'expansion d'entrées/sorties TOR ou analogiques (selon modèle).
- Les bases modulaires intègrent :
- 1 entrée analogique tension 0...10 V, 9 bits (512 points),
- 1 point de réglage analogique accessible en face avant. Ce point peut être réglé sur une valeur comprise entre 0 et 1023.

Base modulaire	Entrée 24v DC	Sorties	Type de raccordement	Ports série	Expansion dôentrées /sorties	Extension module interface	Cartouche optionnelle
TWD LMDA 20DTK	12	8 à transistor source	Connecteur type HE 10	1×RS 485 en option 1×RS 232C /485	4 modules	1 module : afficheur ou liaison série	2 emplacement horodateur ou mémoire
TWD LMDA 20DUK	12	8 à transistor sink	Connecteur type HE 10	1×RS 485 en option 1×RS 232C /485	4 modules	1 module : afficheur ou liaison série	2 emplacement horodateur ou mémoire
TWD LMDA 20DRT	24	16 à transistor source	Connecteur type HE 10	1×RS 485 en option 1×RS 232C /485	7 modules	1 module : afficheur ou liaison série	2 emplacement horodateur ou mémoire
TWD LMDA 40DTK	24	16 à transistor source	Connecteur type HE 10	1×RS 485 en option 1×RS 232C /485	7 modules	1 module : afficheur ou liaison série	2 emplacement horodateur ou mémoire

TWD LMDA 40DUK	24	16 à transistor sink	Connecteur type HE 10	1×RS 485 en option 1×RS 232C /485	7 modules	1 module : afficheur ou liaison série	2 emplacement horodateur ou mémoire

Tableau I.2 : *Caractéristiques des différentes bases modulaires Twido* [3]

I.5.2.3 Programmation

Le logiciel TwidoSoft offre une programmation aisée à partir des instructions langage liste dôinstructions ou des éléments graphiques du langage à contacts.

I.5.3 Automate Modicon TSX Micro

I.5.3.1 Automates TSX 37 05 (et TSX 37 08)

L'automate TSX 37 05 (TSX 37 08) comprend un bac intégrant une alimentation à 100/240 V, un processeur incluant une mémoire RAM de 11 K mots (programme, données et constantes), 1 mémoire de sauvegarde Flash EPROM, un module d'entrées/sorties ñTout ou Rienò TSX DMZ 28DR (16 entrées et 12 sorties à relais) et un emplacement disponible.

L'emplacement disponible peut recevoir :

- 1 module d'entrées/sorties TOR au format standard de tout type.
- 2 modules demi format de type entrées/sorties TOR, sécurité, entrées/sorties analogiques et comptage. [6]

Description :

L'automate TSX 37 05 (TSX 37 08) comprend :

1 Un bac à 2 emplacements.

2 Un bloc de visualisation centralisé.

3 Une prise terminal repérée TER (protocole Uni-Telway maître/esclave, Modbus RTU esclave ou mode caractères).

4 Une trappe d'accès aux bornes d'alimentation.

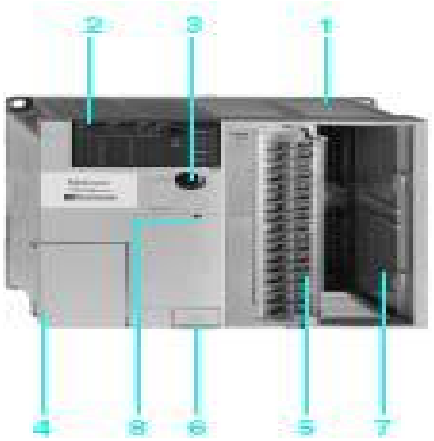


Figure I.6 : Automate TSX 35 05 [3]

5 Un module à 16 entrées et 12 sorties ñTout ou Rienò positionné dans le premier emplacement (positions 1 et 2). Inclut le bornier à vis de raccordement.

6 Une trappe d'accès à la pile optionnelle.

7 Un emplacement disponible pour module(s) dœntrées/sorties (1 au format standard ou 2 au demi-format).

8 Un bouton de réinitialisation.

I.5.3.2 Automates TSX 37 10

Les automates TSX 37 10 compacts et modulaires se différencient par leur tension d'alimentation et le type de module d'entrées/sorties ñTout ou Rienò implanté de base dans le premier emplacement.

Chaque configuration TSX 37 10 comprend un bac intégrant une alimentation (24V DC ou 100/240V AC), un processeur incluant une mémoire RAM de 14 K mots (programme, données et constantes), une mémoire de sauvegarde Flash EPROM, un horodateur, un module d'entrées/sorties ñTout ou Rienò (28 ou 64 entrées/sorties) et un emplacement disponible. Un mini bac d'extension TSX RKZ 02 permet d'augmenter le nombre d'emplacements de 2 (4 positions).

Chaque emplacement disponible peut recevoir :

- 1 module dœntrées/sorties TOR au format standard de tout type.
- 2 modules demi-formats de type entrées/sorties TOR, sécurité, entrées/sorties analogiques et comptage.

De plus, les automates TSX 37 10 peuvent se connecter au réseau Ethernet TCP/IP ou à un Modem via le coupleur autonome externe TSX ETZ 410/510.

Description

Les automates TSX 37 10 et le mini bac d'extension TSX RKZ 02 comprennent :

1 Un bac de base à 2 emplacements.

2 Un bloc de visualisation centralisée.

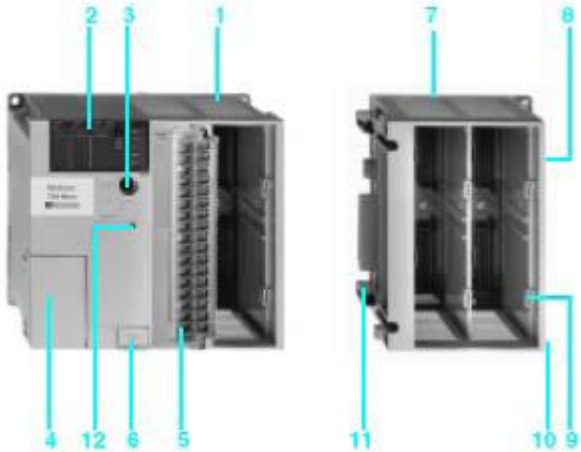


Figure I.7: *Automate TSX 37 10* [3]

3 Une prise terminal repérée TER (protocole Uni-Telway, Modbus RTU maître/esclave ou mode caractères).

4 Une trappe d'accès aux bornes d'alimentation.

5 Un module 28 ou 64 entrées/sorties ñTout ou Rienò positionné dans le premier emplacement (positions 1 et 2).

6 Une trappe d'accès à la pile optionnelle.

7 Un mini bac d'extension à 2 emplacements disponibles (positions 5 à 8).

8 Un voyant de présence de tension 24 V DC.

9 Des bornes d'alimentation protégées par un cache amovible, pour le raccordement d'une alimentation auxiliaire 24V DC dans le cas des automates alimentés en 100/240V AC.

10Une borne de masse.

11Des connecteurs de raccordement à l'automate de base.

12Un bouton de réinitialisation.

I.5.3.3 Automates TSX 37 21/22

Les automates TSX 37 21/22 modulaires se différencient entre eux par leur tension d'alimentation et/ou la possibilité d'effectuer sur la base, du comptage rapide et des fonctions analogiques.

Chaque automate comprend : un bac à 3 emplacements libres intégrant une alimentation (24V DC ou 100/240V AC), un processeur incluant une mémoire RAM de 20 K mots (programme, données et constantes), une mémoire de sauvegarde Flash EPROM, un horodateur, 2 emplacements pour carte PCMCIA (1 carte communication et 1 carte extension mémoire de 128 K mots maximum). Un mini bac d'extension TSX RKZ 02 permet d'augmenter le nombre d'emplacements de 2 (4 positions).

Chaque emplacement disponible peut recevoir :

- 1 module d'entrées/sorties TOR au format standard.
- 2 modules demi-formats de type entrées/sorties TOR, sécurité, entrées/sorties analogiques et comptage.

De plus, les automates TSX 37 21/22 peuvent se connecter au réseau Ethernet TCP/IP ou à un Modem via le coupleur autonome externe TSX ETZ 410/510.

Description

Les automates TSX 37 21/22 et le mini bac d'extension TSX RKZ 02 comprennent :

1 Un bac de base à 3 emplacements disponibles (positions 1 à 6).

2 Un emplacement réservé à un module au format standard.

3 Un bloc de visualisation centralisée.

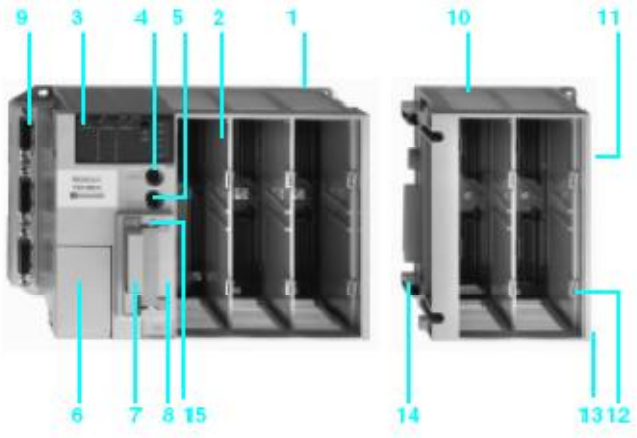


Figure 1.8 : *Automate TSX 37 22* [3]

4 Une prise terminal repérée TER (protocole Uni-Telway, Modbus RTU maître/esclave ou mode caractères).

5 Une prise de dialogue opérateur repérée AUX.

6 Une trappe d'accès aux bornes d'alimentation.

7 Un emplacement pour une carte PCMCIA d'extension mémoire.

8 Un emplacement pour une carte PCMCIA de communication.

9 Des connecteurs type SUB-D pour les fonctions analogiques et comptage intégrés (avec TSX 37 22).

10 Un mini bac d'extension à 2 emplacements disponibles (positions 7 à 10).

11 Un voyant de présence de tension 24V DC.

12Des bornes d'alimentation protégées par un cache amovible, pour le raccordement d'une alimentation auxiliaire 24V DC dans le cas des automates alimentés en 100/240V AC.

13 Une borne de masse.

14Des connecteurs de raccordement à l'automate de base.

15 Un bouton de réinitialisation.

I.5.4 Automates Modicon M340

I.5.4.1 Modules processeurs

Les processeurs Standard et Performance de la plate-forme dôautomatisme Modicon M340 gèrent l'ensemble d'une station monorack automate dont 11 emplacements maximum peuvent être équipés de :

- Modules d'entrées/sorties ñTout ou Rienò.
- Modules d'entrées/sorties analogiques.
- Modules métiers (comptage, communication Ethernet TCP/IP).[6]

Figure I.9 : *Plate forme dôautomatisme Modicon M340* [3]

Quatre processeurs sont proposés, ils se différencient par leurs capacités mémoire, vitesses de traitement, nombre dôE/S et nombre et type de ports .

De plus, selon le modèle, ils proposent au maximum et dôune manière non cumulative :

- De 512 à 1024 entrées/sorties ñTout ou Rienò,
- De 128 à 256 entrées/sorties analogiques,
- De 20 à 36 voies métiers comptage,
- De 0 à 2 réseaux Ethernet TCP/IP (avec ou sans port intégré et un module réseau).

Selon les modèles, les processeurs Modicon M340 intègrent :

- Un port Ethernet TCP/IP 10BASE-T/100BASE-TX,
- Un bus machines et installations CANopen,
- Une liaison série Modbus,
- Une prise TER de type USB (pour connexion dôun terminal de programmation).

Chaque processeur est fourni avec une carte mémoire permettant :

- La sauvegarde de lôapplication (programme, symboles et constantes)
- Lôactivation dôun serveur Web de base du port Ethernet intégré de classe Transparent Ready B10 (selon modèle).

Cette carte mémoire peut être remplacée par un autre type de carte mémoire, à commander séparément, supportant :

- Egalement la sauvegarde de lôapplication et lôactivation du serveur Web de base,
- Une zone de 16 Mo pour stockage de données additionnelles organisées en système de fichiers (répertoires et sous-répertoires).

I.5.4.2 Programmation

La mise en oeuvre de processeurs de la plate-forme dôautomatisme Modicon M340 nécessite soit :

- Le logiciel de programmation Unity Pro Small.
- Le logiciel de programmation Unity Pro Medium, Large ou Extra Large identique à celui permettant la mise en î uvre des plates-formes dôautomatisme Modicon Premium et Modicon Quantum,.
- Avec éventuellement, selon les besoins :

➢ le logiciel Unity EFB toolkit pour le développement en langage C de librairies de blocs fonction EFs et EFBs,

➢ le logiciel Unity SFC View pour la visualisation et le diagnostic des applications écrites en langage diagramme fonctionnel en séquence (SFC) ou Grafcet.

Les librairies logicielles de blocs fonctions donnent la puissance aux processeurs Modicon M340 afin de répondre aux métiers spécialisés dans les domaines de :

- La régulation de procédés via des boucles de régulation programmables (bibliothèque de blocs fonctions EFs et EFBs)
- La commande de mouvement avec de multiples fonctions dôaxes indépendants (bibliothèque MFB ñMotion Function Blocksò). Les axes sont pilotés par des variateurs de vitesse Altivar 31/71 ou des servo variateurs Lexium 05/15 connectés sur le bus machines & installations CANopen.

I.5.5 Modicon Premium

I.5.5.1 Présentation

Les processeurs des plates-formes d'automatisme Premium TSX P57 ŏŏ3M/3AM et 23M/23AM gèrent l'ensemble d'une station automate constituée de modules d'entrées/sorties ñTout ou Rienò, modules de sécurité Preventa, de modules d'entrées/sorties analogiques et de modules métiers qui

peuvent être répartis sur un ou plusieurs racks connectés sur le bus X ou peuvent être distribués sur bus de terrain.

I.5.5.2 Les Processeurs TSX P57

Les processeurs proposés sont segmentés par des capacités différentes au niveau de la mémoire, des entrées/sorties ñIn rackò, des communications ainsi que par leurs vitesses de traitement. Selon le modèle :

- De 4 à 16 racks.
- De 512 à 2040 entrées/sorties ñTout ou Rienò.
- De 24 à 256 entrées/sorties analogiques.
- De 8 à 64 voies métiers. Chaque module métier (comptage, commande de mouvement, liaison série ou pesage) compte pour 1 ou plusieurs voies métiers.
- De 1 à 4 réseaux (Ethernet TCP/IP, Fipway, Ethway, Modbus Plus), de 2 à 8 bus capteurs/actionneurs AS-Interface, de 1 à 2 bus de terrain (CANopen, INTERBUS, Profibus DP), 0 ou 1 bus de terrain Fipio, des liaisons séries (Modbus, Uni-Telway).
- De 10 à 20 voies de régulation.

La conception et mise en oeuvre des applications sur les Automates Premium se fait à lôaide du logiciel PL7 Pro ou Unity Pro. [6]

Type de processeurs			TSX P57 103M	TSX P57 153M	TSX P57 203M	TSX P57 2623M	TSX P57 253M	TSX P57 2823M
Configuration maximale	Nb de racks	4/6/8 Emplacements	4		16			
		12 emplacements	2		8			
	Nb d'emplacements maximal pour modules		32		128			
Fonctions	Nb maximal	E/S TOR	512		1024			
		E/S analogiques	24		80			
		Voies de régulation	-		10 (jusqu'à 30 boucles simples)			
		Voies métiers	8		24			
	Connexions intégrées	Ethernet TCP/IP	-			1	-	1
		Fipio gestionnaire	-	1 (63 agents)	-		1 (127 agents)	
		Liaison série	1 liaison avec 2 connecteurs (TER et AUX) 19.2 Kbit/s					
	Nb max de connexions	Réseaux	1		1	1, aucun si Ethernet intégré utilisé	1	1, aucun si Ethernet intégré utilisé
		Bus AS-i	2		4			
		Bus CANopen	1	-	1			
		Bus InterBus ou Profibus DP	-		1, aucun si CANopen utilisé			
Mémoires	Capacité maximale	Sans carte PCMCIA(Kmots)	32, programme et données		48, programme et données		64, programme et données	
		Avec carte PCMCIA(Kmots)	64, programme 32, données		160,programme 48, données		160, programme 64, données	
		Stockage de données (Kmots)	128		2688			
	Taille max des zones objets	Bits internes localisés (%Mi) (bits)	4096		8132			
		Données internes Localisées (Kmots)	30.5 pour mots internes %M•i 32 pour mots constants %K•i					

Tableau I.3 : *Caractéristiques spécifiques à chaque processeur TSX premium* [3]

I.5.6 Modicon Quantum

I.5.6.1 Présentation

Les unités centrales de la plate-forme d'automatisme Modicon Quantum sont basées sur des processeurs haute performance 486 et Pentium, et sont compatibles avec le logiciel Unity Pro.

De nombreuses fonctionnalités sont incluses de base dans les processeurs Quantum :

- Temps de cycle réduit avec acquisition rapide des entrées/sorties.
- Traitement d'interruption sur événement de temps ou en provenance d'entrées.
- Traitement possible en tâche rapide comme en tâche maître.
- Extension des capacités mémoire par cartes mémoire PCMCIA.
- Multiples ports de communication intégrés au processeur.

Figure I.10 : *Automate Modicon Quantum* [6]

- Diagnostic et maintenance aisés grâce au bloc de visualisation LCD en face avant des processeurs haut de gamme.

Les processeurs proposés se différencient par leurs capacités mémoire, leurs vitesses de traitement et leurs possibilités de communication.

I.5.6.2 Mémoire sauvegardée protégée

Les processeurs supportent de base leur programme application en mémoire RAM interne sauvegardée par pile. Cette pile est logée en face avant du processeur et peut-être remplacée, processeur en fonctionnement.

Pour protéger le programme application en cas de mauvaise manipulation, les processeurs sont équipés en face avant d'un commutateur à clé destiné à protéger la mémoire. Ce commutateur peut être également utilisé pour autoriser la commande Run/Stop d'exécution du processeur. Le processeur 140 CPU 311 10 ne dispose que d'un commutateur pour la protection de la mémoire. Un bit de protection mémoire, à positionner en mode configuration, est également disponible pour le verrouillage de toute modification de programme (via PC de programmation ou par téléchargement de programme) [6].

I.5.6.3 Ports de communication

Tous les processeurs sôintègrent dans les architectures de réseaux Modbus et Modbus Plus. Des commutateurs rotatifs en face arrière des modules permettent de définir lôadresse du (des) port(s) Modbus Plus. Chaque équipement réseau Modbus Plus doit avoir une adresse unique dans la plage 1...64. Les réglages des ports Modbus comprennent : la vitesse, la parité, le nombre de bits de données, le nombre de bits de stop, le protocole et lôadresse de lôEsclave. Par défaut, ces réglages sont 9600 bit/s, parité paire, 8 bits de données, 1 bit de stop, mode RTU et adresse 1.

Un commutateur en face avant des processeurs permet de paramétrer le port Modbus comme support de communication modem (2400 bit/s, parité paire, 7 bits de données, 1 bit de stop, mode ASCII et adresse 1).

Les processeurs 140 CPU 434 12 A et 140 CPU 534 14B disposent de 2 ports séries Modbus:

- Port 1 Modbus, paramétrable comme modem.
- Port 2 Modbus, gestion de flux RIS/CTS (ne supporte pas la liaison modem).

I.5.6.4 Coprocesseur mathématique

Pour des applications justifiant des traitements mathématiques conséquents, un coprocesseur mathématique est intégré sur certains processeurs. Ce coprocesseur réduit considérablement les temps dôexécution de la Bibliothèque de Régulation 984 (PCFL) et de lôEditeur dôEquation, ainsi que ceux des opérations mathématiques écrites au moyen de langages IEC. Cette réduction des temps dôexécution des opérations en virgule flottante signifie plus de puissance pour le traitement dôalgorithmes de régulation et de calculs mathématiques [6].

I.6 Conclusion

Lôétude des différents automates de la même gamme nous aide à mieux classer les produits dôautomatisation quôon aura à manipuler. Mais aussi à bien choisir lôautomate qui convient à notre application. Côest le Modicon M340 Ethernet .(une brève description de cet automates est illustré dans lôannexe D).

Chapitre II

Le langage de programmation Unity Pro

II.1 Présentation de logiciel unity pro

Le logiciel Unity Pro est un atelier logiciel destiné à programmer les automates Telemecanique Modicon M340, Premium, Atrium ou Quantum à l'aide des langages de programmation conformes à la norme CEI 61131-3 : langage à blocs fonction (FBD), langage à contacts (LD), diagramme fonctionnel en séquence (SFC), liste d'instructions (IL) et littéral structuré (ST). Dans ce chapitre on décrit la structure générale du logiciel avec mise en ouvre d'une application.[8]

II.2 Les versions

Unity pro est proposé en 5 versions appelées prologiciel. Les progiciels disponibles sont les suivants :

- Unity Pro S.
- Unity Pro M.
- Unity Pro L.
- Unity Pro XL.
- Unity Pro XLS.
- Unity Developers Edition (UDE).

Le tableau suivant présente les propriétés principales des différents progiciels :

II.3 Fonctions d'Unity Pro

II.3.1 Plates-formes matérielles

Unity Pro prend en charge les plates-formes matérielles suivantes :

- Modicon M340
- Premium
- Atrium
- Quantum

II.3.2 Langages de programmation

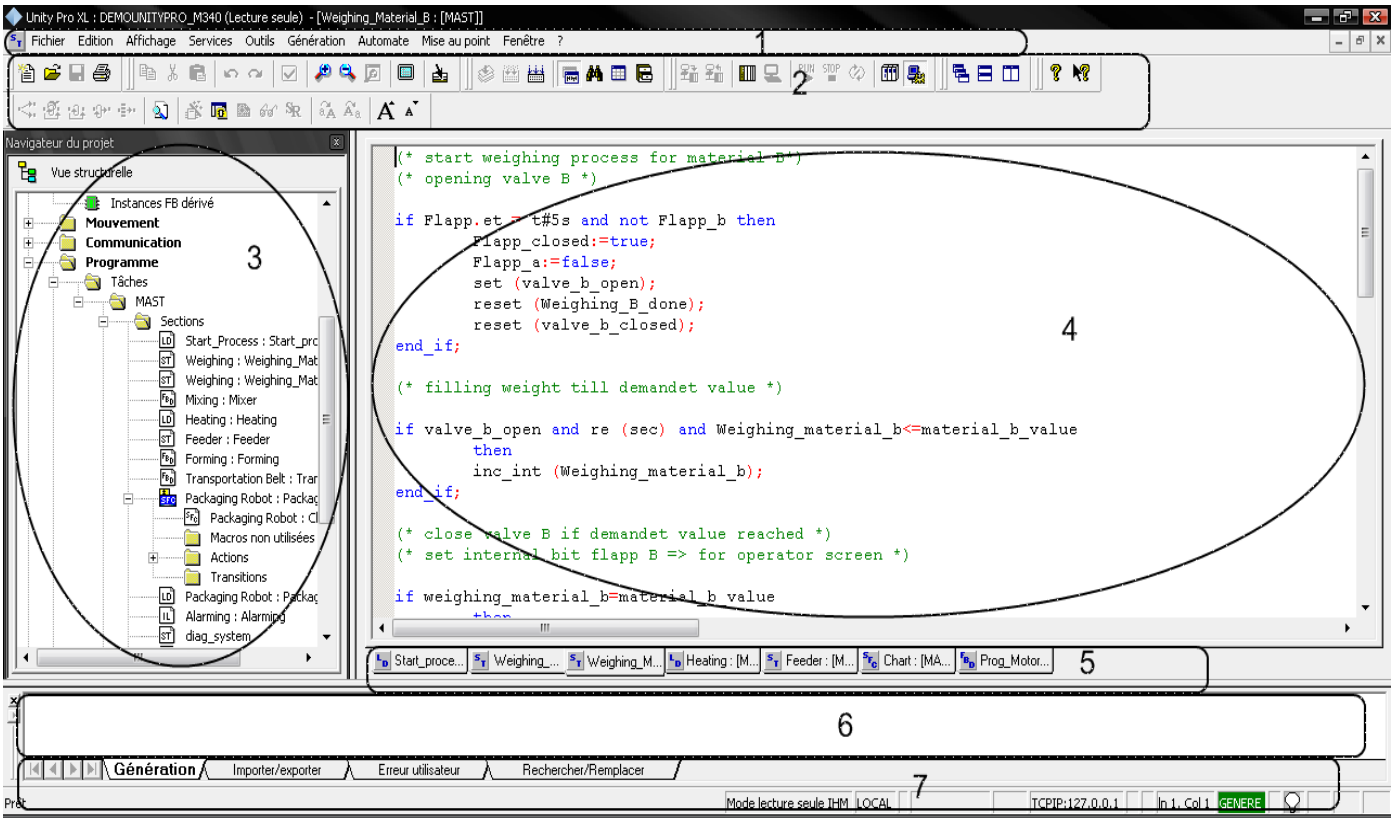
Unity Pro propose les langages suivants pour la création du programme utilisateur :

- Langage à blocs fonction (FBD)
- Langage à contacts (LD)
- Liste d'instructions IL

- Littéral structuré ST
- Diagramme fonctionnel en séquence SFC

II.4 Interface utilisateur

L'interface utilisateur se compose de plusieurs fenêtres et barres d'outils pouvant être positionnées librement.[8]



Légende : 1 Barre de menus 2 Barre d'outils 3 Navigateur du projet 4 Fenêtre de l'éditeur (éditeurs de langages, éditeur de données, etc.) 5 Onglets d'accès direct aux fenêtres de l'éditeur

6 Fenêtre d'information 7 Ligne d'état

Figure II.1 : *Interface utilisateur* [9]

II.5 Technique d'adressage

Avant de parler de la technique d'adressage on définit les variables

- Une variable est une entité mémoire de type BOOL, WORD, DWORD, etc., dont le contenu peut être modifié par le programme au cours de l'exécution.

- Une variable affectée est une variable affectée à un module d'E/S ou associée à une référence mémoire. Par exemple :la variable Pression_eau est associée au mot mémoire %MW102. Pression_eau est considérée comme localisée.
- Une variable non affectée est une variable non affectée à l'E/S ou non associée à une référence mémoire (impossible de déterminer sa position en mémoire). Une variable qui n'a pas d'adresse affectée est dite non affectée.
- Une variable publique est une variable disponible avec certains blocs fonctions. Ces variables transfèrent des valeurs statistiques (valeurs qui ne sont pas influencées par le processus) au bloc fonction. Elles sont utilisées pour régler les paramètres du bloc fonction.
- Une variable privée est une variable utilisée par certains blocs fonctions. Ces variables ne sont pas accessibles par le programme d'application
- I/ODDT est l'abréviation d'Input/Output Derived Data Type (type de données dérivées d'entée/sortie). L'élément I/ODDT désigne un type de données structurées représentant un module ou une voie d'un module d'automate. Chaque module expert de l'application possède ses propres I/ODDT.
- Les constantes sont des variables de type INT, DINT ou REAL affectées dans le champ constant (%K) ou des variables utilisées dans l'adressage direct (%KW, %KD ou %KF). Le contenu de ces constantes ne peut pas être modifié par le programme pendant l'exécution. [10]

Noms de Variables :

- Non affectées : Nom d'étiquette sans adresse matérielle ; Implémentation dans le code, donc plus rapide qu'une variable affectée.
- Affectées : Nom d'étiquette avec une adresse matérielle (RAM d'état)

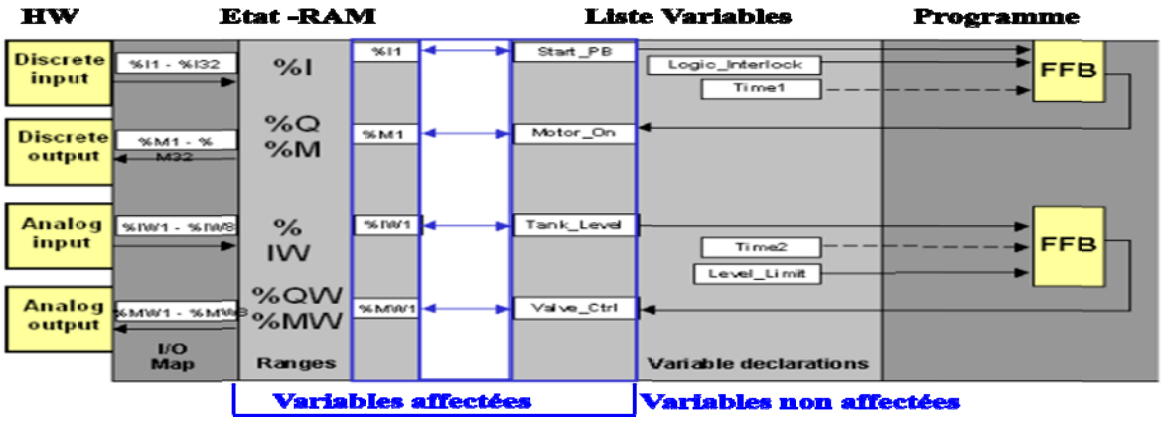


Figure II.2 : *Configuration, adressage, affectations* [10]

%	I / Q	X / W / D	r	.	m	.	c	.	d	.	j
Symbole	Type I = Entrée Q = Sortie	Format X = Booléen W = Mot D = Double mot	Rack		Module		Voie		Rang		Bit

Figure II.3 : *Adressage physique des entrées / sorties* [10]

Adresse	Adresse topologique	Donnée utilisée
0xxxxx	%Qr.m.c.d / %Mi	Bits de module de sortie et bits internes
1xxxxx	%Ir.m.c.d / %Ii	Bits de module d'entrée
3xxxxx	%IWr.m.c.d / %IWi	Mots d'entrée de module d'entrées / sorties
4xxxxx	%QWr.m.c.d / %MWi	Mots de sortie de module d'entrées / sorties et mots internes

Figure II.4 : *Adressage topologie des données localisées Quatum* [10]

II.6 Configuration matérielle

Configuration de bus local (Bus X)

Lors de la création d'un projet, une configuration par défaut est créée automatiquement suite à des choix imposés par le logiciel de programmation. Le rack par défaut est sélectionné. Son adresse est la suivante :

- 0 pour un automate de la famille Premium/Atrium ou Modicon M340,
- 1 pour un automate de la famille Quantum.

1. A partir du navigateur de projet, ouvrez le répertoire Configuration.

2. unity pro affiche la Configuration par défaut.

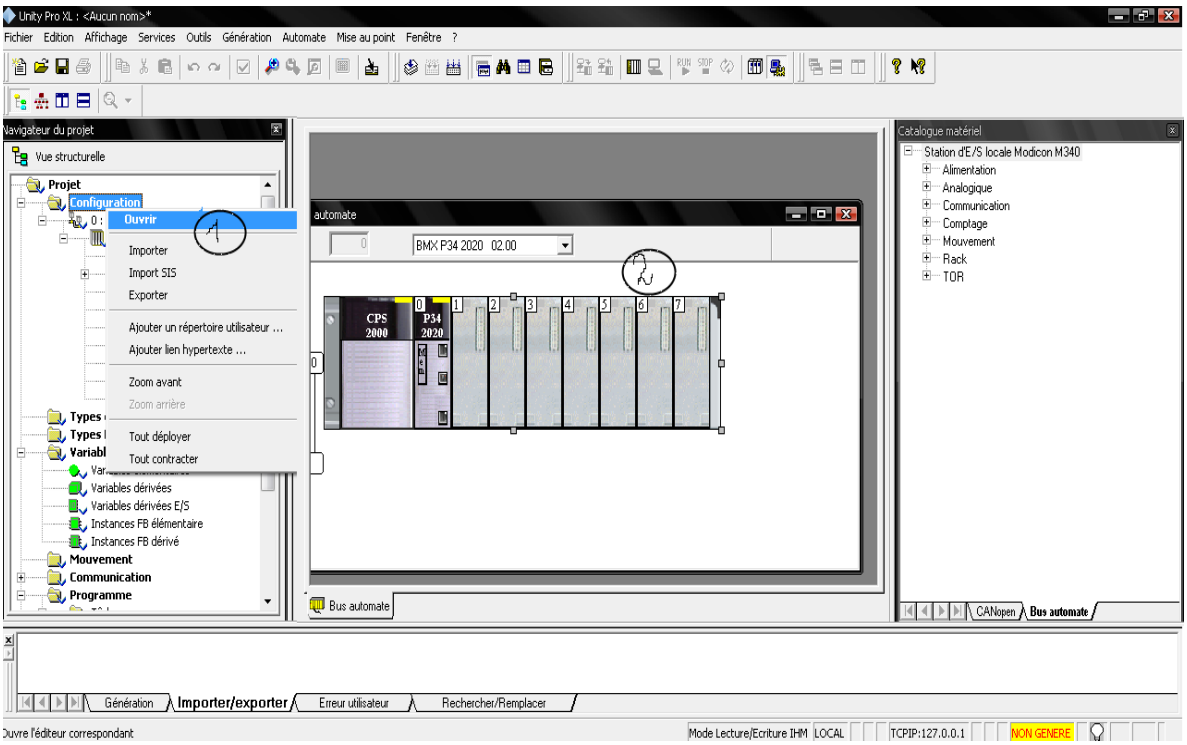


Figure II.5 : *Configuration matérielle* [9]

La Configuration par défaut

Pour une station Modicon M340 :

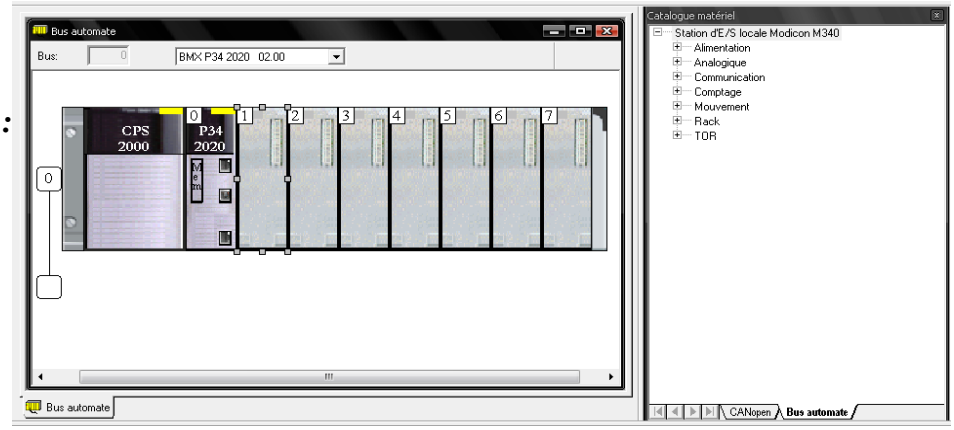


Figure II.6 : *Configuration matérielle pour Modicin M340* [9]

Pour une station Premium

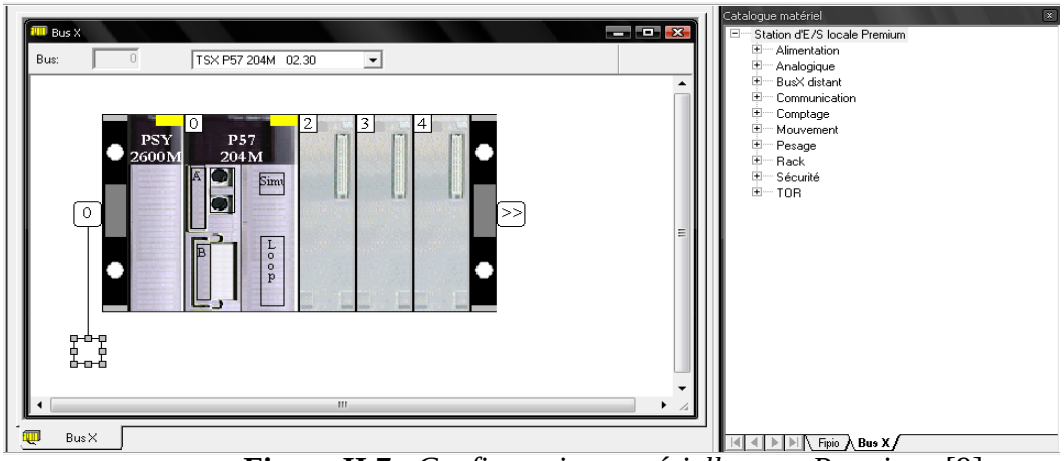


Figure II.7 : *Configuration matérielle pour Premium* [9]

Pour une station Quantum

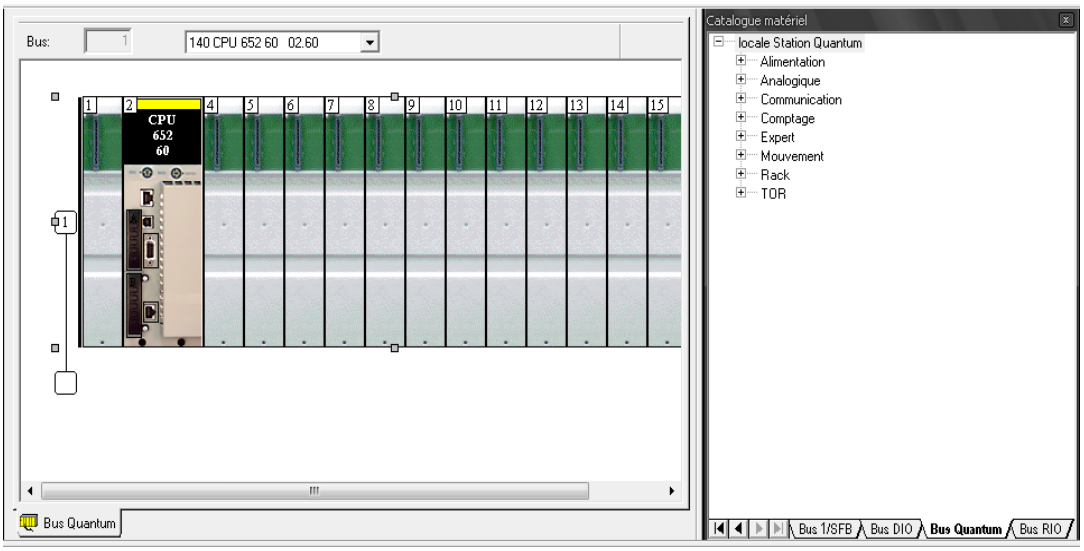


Figure II.8 : *Configuration matérielle pour Quantum* [9]

A ce stade les éléments physiques configurés constituant la station sont:

- le rack.
- le module d'alimentation (sauf pour les stations Quantum).
- Le processeur.

II.6.1 Configuration des racks sur bus local

Vous pouvez manipuler les racks d'une station automate :

Manipulation des racks est bien détaillées a l'Annexe A

II.6.2 Configuration des modules d'alimentation

Lorsque vous créez une application

- dans une station Modicon M340 ou Premium, un module d'alimentation est configuré par défaut,
- dans une station Quantum, aucun module d'alimentation n'est configuré par défaut.

Les Manipulations des modules d'alimentation est décrit sur l'Annexe A.

II.6.3 Configuration du processeur

Choix du processeur

Le choix du processeur intervient lors de la création du projet, ce choix n'est pas irréversible.

Exécutez les actions suivantes

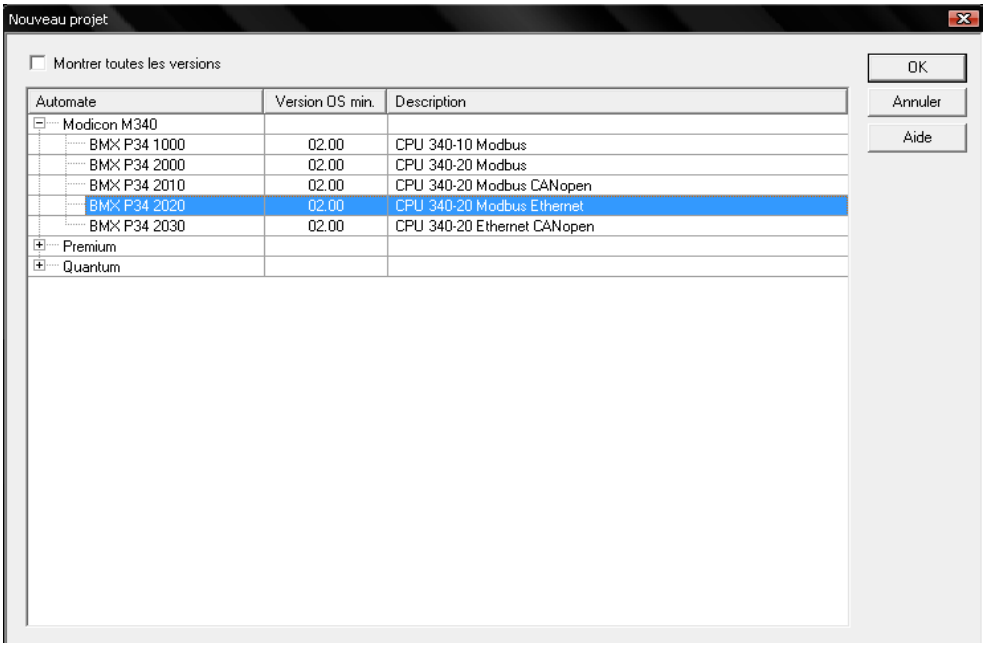
Etape	Action
1	Dans l'écran de bienvenue du logiciel, sélectionnez Nouveau dans le menu Fichier.
2	Ouvrez le type d'automate requis. Exemple : 
3	Si vous voulez voir toutes les versions d'automates, cliquez sur la case Montrer toutes les versions.
4	Sélectionnez le processeur.
5	Validez par OK.

Tableau II.1 : *Choix d'un processeur* [8]

Remplacement du processeur

L'éditeur de configuration vous aide si vous souhaitez remplacer le processeur. Si un remplacement n'est pas autorisé, un message vous avertit. Le nouveau processeur doit obligatoirement appartenir à la même famille d'automate que le processeur précédemment configuré. Si certains modules d'entrée/sortie précédemment configurés ne sont plus pris en charge par le nouveau processeur, des messages d'erreur s'affichent lors de l'analyse du projet. Il est possible de remédier à ces incompatibilités

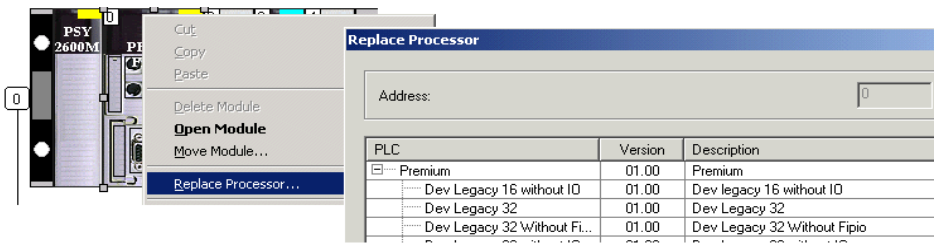


Figure II.9 : *Remplacement d'un processeur* [9]

Note : Cette opération ne peut s'effectuer qu'en mode local (automate non connecté)

Configuration du processeur Modicon M340

Accès à l'écran de configuration :

1. Sélectionnez le processeur.
2. Par le menu contextuel sélectionnez la commande Ouvrir.
3. Choisissez l'onglet Configuration.

Ecran Configuration :

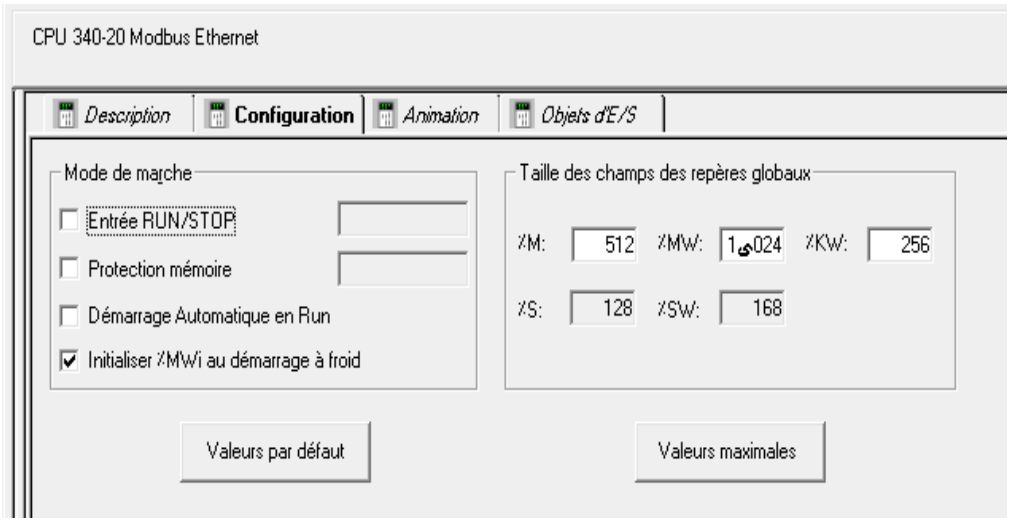


Figure II.10 : *Ecran de configuration Modicon M340* [9]

Entrée RUN/STOP

L'entrée %Ir.m.c peut être paramétrée pour commander le passage RUN/STOP de l'automate de la façon suivante :

%Ir.m.c à 1 -> l'automate bascule en mode RUN (exécution du programme)

%Ir.m.c à 0 -> l'automate bascule en mode STOP (arrêt de l'exécution du programme).

Protection de mémoire

L'entrée %Ir.m.c peut être paramétrée pour protéger la mémoire vive de l'application et la carte mémoire de la façon suivante :

%Ir.m.c à 0 -> l'application interne et la carte mémoire ne sont pas protégées,

%Ir.m.c à 1 -> l'application interne et la carte mémoire sont protégées.

Démarrage Automatique en Run

L'activation de cette option fait automatiquement passer l'automate en mode RUN lors d'un démarrage à froid.

Initialiser %MWi

- Si vous cochez la case (état par défaut), lors d'un démarrage à froid ou d'un téléchargement :

les valeurs %MWi sont traitées comme les autres variables globales (initialisées sur la valeur 0 ou sur la valeur initiale, selon l'application) dans tous les cas de démarrage à froid ;

- Si vous décochez la case, lors d'un démarrage à froid ou d'un téléchargement :

Les mots internes %MW sont restaurés à partir de la flash mémoire interne s'ils ont été préalablement enregistrés dans cette mémoire (à l'aide du mot %SW96) ;

Sinon :

- si le démarrage à froid est lié à une mise hors tension ou à une pression sur le bouton de réinitialisation, les mots internes %MW sont initialisés ;
- si ce n'est pas le cas, les valeurs actuelles des mots internes %MW sont conservées.

Configuration du processeur Premium

- Définir les objets globaux de l'application : nombre de bits et de mots

II.6.4 Configuration des modules d'E/S

Positionnement des modules

La station locale pour Modicom M340 propose Les modules d'E/S suivants:

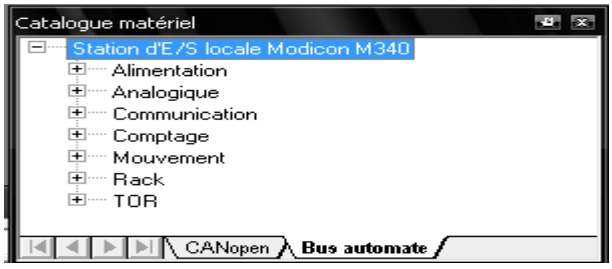


Figure II.11 : *Les différents modules de Modicon M340*[9]

La station locale pour premium propose Les modules d'E/S suivants:

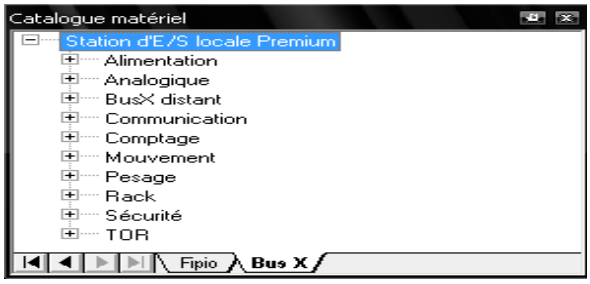


Figure II.12 : *Les différents modules du Premium*[9]

La station locale pour Quantum propose Les modules d'E/S suivants:

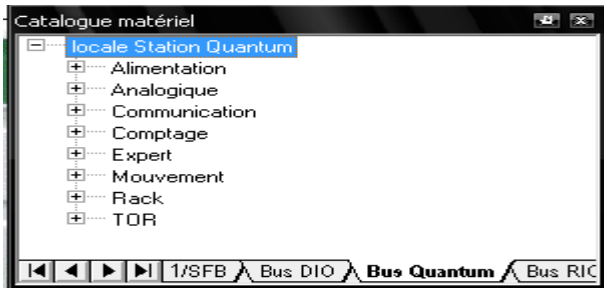


Figure II.13 : *Les différents modules du Quantum*[9]

Pour positionner un module Exécutez les actions suivantes:

1) Sélectionnez avec la souris l'emplacement du module à insérer, huit poignets entourent le module.

2) Par le menu contextuel sélectionnez la commande Nouvel équipement.

3) Sélectionnez le métier et le module souhaité.

Exemple :

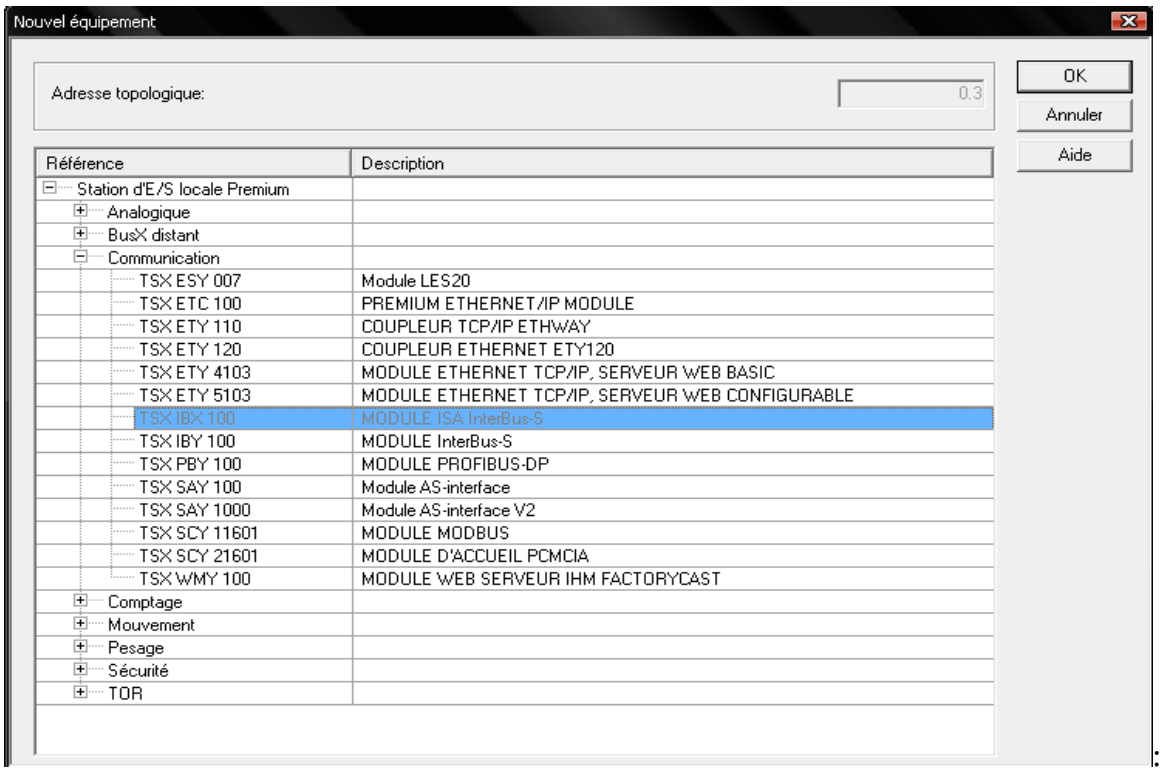


Figure II.14 : *Comment positionner les modules*[9]

4) Validez par OK.

Configuration des écrans des modules d'E/S

Pour configurer l'écran :

1. Sélectionnez avec la souris le module inséré dans le rack.

2. Par le menu contextuel sélectionnez la commande ouvrir le module.

La configuration des différents écrans pour différent type de module est la suivante (on traite l'exemple d'un écran d'un module d'E/S communication pour les autres voir annexe B)

module d'E/S Communication

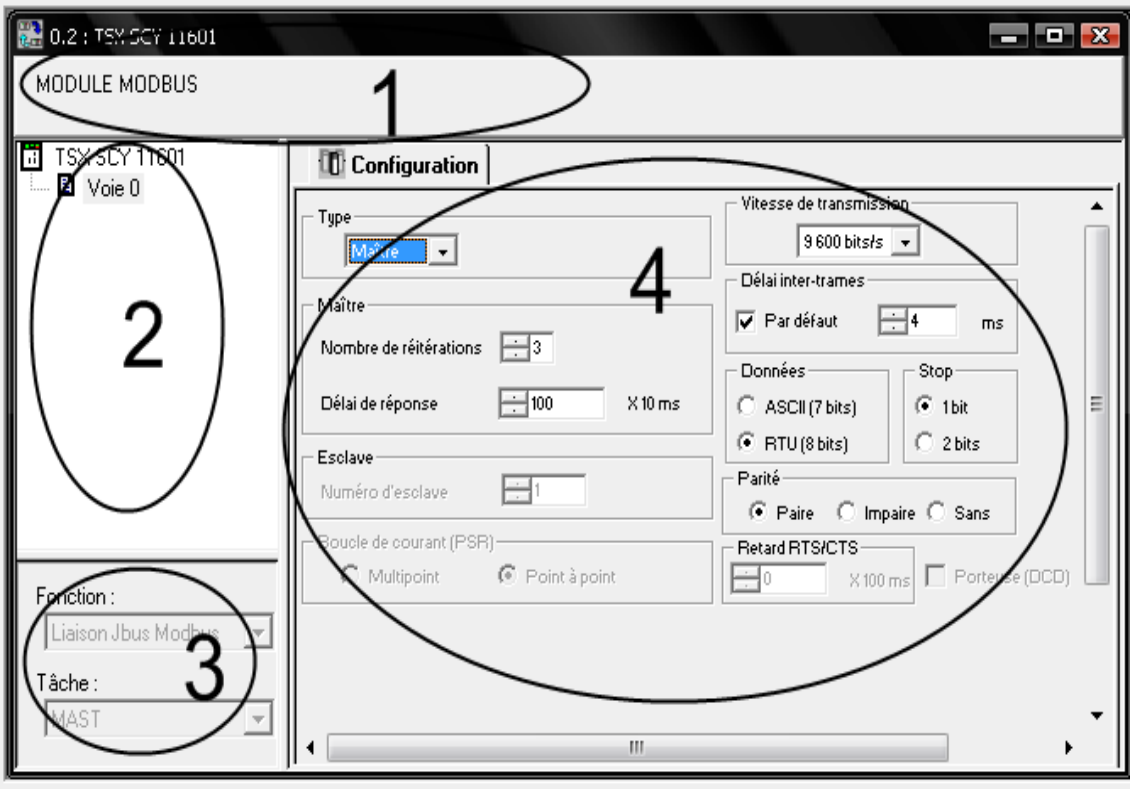


Figure II.15 : *module d'E/S communication (l'écran de configuration)*[9]

Adresse	Elément	Fonction
1	Zone Module	Rappelle l'intitulé abrégé du module
2	Zone Voie	Permet : ➢ de choisir la voie ; ➢ d'afficher le Symbole, nom de la voie défini par l'utilisateur (via l'éditeur de variables).
3	Zone Paramètres généraux	Permet de choisir les paramètres généraux associés à la voie : ➢ Fonction : selon la voie, les fonctions disponibles sont Modbus, Mode caractère et Uni-Telway. Par défaut, aucune fonction n'est configurée. ➢ Tâche : définit la tâche MAST dans laquelle seront échangés les objets à échange implicite de la voie.
4	Zone	Permet de configurer les paramètres de configuration de la voie. Certaines

	Configuration	sélections peuvent être verrouillées. Elles apparaissent alors grisées. Elle est divisée en deux types d'informations : ➢ paramètres d'application ; ➢ paramètres de transmission.

Tableau II.2 : *les différents éléments de l'écran de configuration et leurs fonctions* [8]

Description de la zone configuration :

➢ Paramètres d'application :

Ils sont répartis dans quatre fenêtres différentes :

- la fenêtre Type.
- la fenêtre Maître.
- la fenêtre Esclave.
- la fenêtre Boucle de courant (PSR).

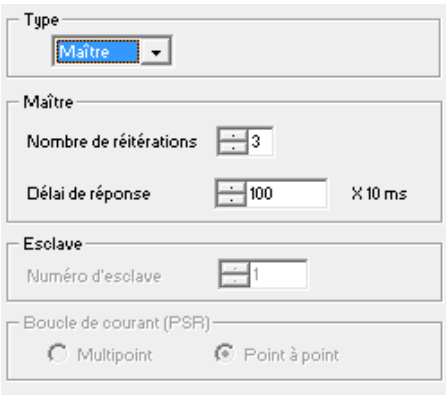


Figure II.16 : *module d'E/S communication (zone configuration)*[9]

Fenêtre Type

Elle vous permet de sélectionner le type de protocole Modbus utilisé par le module :

- Maître : sélectionne le Modbus maître où la station est maître ;
- Esclave : sélectionne le Modbus esclave où la station est esclave ;

Fenêtre Maitre

Vous pouvez alors renseigner :

- Nombre de réitérations : nombre de tentatives de connexion qu'effectue le maître avant de déclarer l'esclave absent.

➢ La valeur par défaut est 3.

➢ Les valeurs possibles sont comprises entre 0 et 15.

➢ La valeur 0 indique qu'il n'y a pas de réitération du maître.

- Temps de réponse : délai écoulé entre la requête émise par le maître et sa réitération en cas de non réponse de l'esclave. Il correspond au temps maximum entre l'émission du dernier caractère de la requête émise par le maître et la réception du premier caractère de la requête renvoyée par l'esclave.

- La valeur par défaut est 1 s (100*10 ms).
- Les valeurs possibles sont comprises entre 10 ms et 10 s.

Fenêtre Esclave

Elle permet de remplir le Numéro d'esclave de l'équipement :

- TSX SCY 21601 :
- La valeur par défaut est 98.
- Les valeurs possibles sont comprises entre 1 et 98.
- TSX SCY 11601 :
- La valeur par défaut est 247.
- Les valeurs possibles sont comprises entre 1 et 247.

Fenêtre Boucle de courant

Elle permet de sélectionner une communication :

- Multipoint (Boucle de courant) ;
- Point à point (Boucle de courant).

Paramètres de transmission :

Ils sont présents dans six fenêtres différentes :

- la fenêtre Vitesse de transmission.
- la fenêtre Délai intercaractères.
- les fenêtres propres aux données et à l'arrêt.
- la fenêtre Parité.
- la fenêtre Retard RTS/CTS.

Fenêtre Vitesse de transmission.

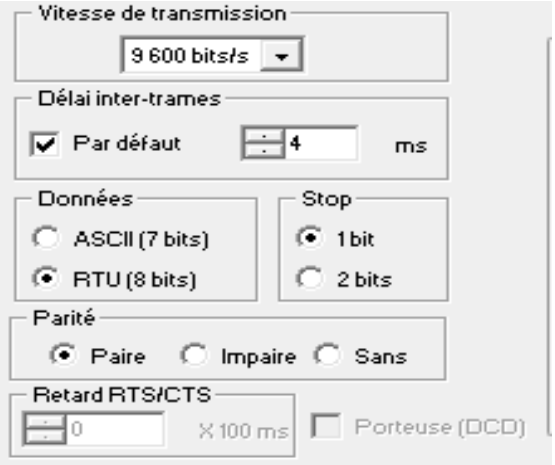


Figure II.17 : *module d'E/S communication (paramètres de transmission)*[9]

Elle permet de sélectionner la vitesse de transmission du protocole Modbus utilisé par le module. Cette dernière est conforme aux autres équipements :

- La vitesse par défaut est de 9 600 bits/s.
- Les vitesses disponibles sont 300, 600, 1 200, 2 400, 4 800, 9 600, 19 200, 38 400 et 57 600 bits/s.
- Les vitesses de 300 et 600 bits/s sont disponibles uniquement avec la carte PCMCIA TSX SCP 111.
- Les vitesses de 38 400 et 57 600 bits/s sont disponibles uniquement avec la carte PCMCIA TSX SCP 1114.

Fenêtre Délai intercaractères

Il s'agit du délai de détection de fin de trame et du temps maximal séparant deux caractères en réception. Il est géré quand l'automate reçoit des messages, qu'il soit maître ou esclave. Il est préconisé d'utiliser les valeurs par défaut dans des configurations sans modem et sans équipement intermédiaire. Sinon, il est nécessaire d'utiliser des valeurs supérieures.

fenêtres propres aux données et à l'arrêt.

- Données

Le champ Données comporte le type de codage utilisé pour communiquer en mode Modbus. Il doit être configuré en fonction des autres équipements :

- ✓ *En mode RTU :*

Les caractères sont codés sur 8 bits.

Le début et la fin de trame sont détectés par un silence d'au moins 3,5 caractères.

L'intégrité de la trame est contrôlée à l'aide de la somme de contrôle CRC contenue dans celle-ci.

- ✓ *En mode ASCII :*

Les caractères sont codés sur 7 bits.

Le début de la trame est détecté par réception des caractères ":" ou par un silence plus long que le délai intercaractères.

La fin de la trame est détectée par CR et LF (retour chariot et retour à la ligne) ou par un silence plus long que le délai intercaractères.

- Stop

Le champ Stop permet d'indiquer le nombre de bits d'arrêt utilisés pour communiquer en Modbus. Les valeurs possibles sont 1 ou 2 bits d'arrêt. Ce champ est défini en fonction des autres équipements

Fenêtre Parité.

Ce champ permet de définir l'adjonction ou non d'un bit de parité ainsi que son type. Les valeurs possibles sont Paire, Impaire ou Sans (par défaut, Paire). Ce champ est défini en fonction des autres équipements.

Fenêtre Retard RTS/CTS.

Avant chaque émission d'une chaîne de caractères, le coupleur active le signal RTS (Requête à émettre) et attend l'activation du signal CTS (Prêt à émettre).

Vous pouvez alors indiquer :

- le temps d'attente maximal entre les deux signaux. Si ce temps est dépassé, la requête n'est pas envoyée sur le bus.
- ➢ La valeur est exprimée en centaines de millisecondes.
- ➢ La valeur par défaut est 0 ms.
- ➢ La valeur est comprise entre 0 s et 10 s.
- ➢ La valeur 0 spécifie l'absence de gestion du retard entre les deux signaux.

- La gestion de la porteuse (signal DCD - Porteuse détectée) est utilisée uniquement dans le cas d'une communication avec un modem à porteuse commandée :

➢ Si l'option est sélectionnée, la réception des caractères est valide uniquement si le signal de la porteuse DCD est détecté.

➢ Si l'option n'est pas sélectionnée, tous les caractères reçus sont pris en compte.

Note :

- La configuration des écrans des autres modules est détaillée sur Annexe A
- Configuration d'équipements sur le bus de terrain est détaillée sur Annexe A

II.7 Description des langages de programmation

II.7.1 Langage à blocs fonction (FBD)

Le langage à blocs fonction FBD permet la programmation graphique de blocs fonction conformément à la norme CEI 61131-3.

Représentation d'une section FBD

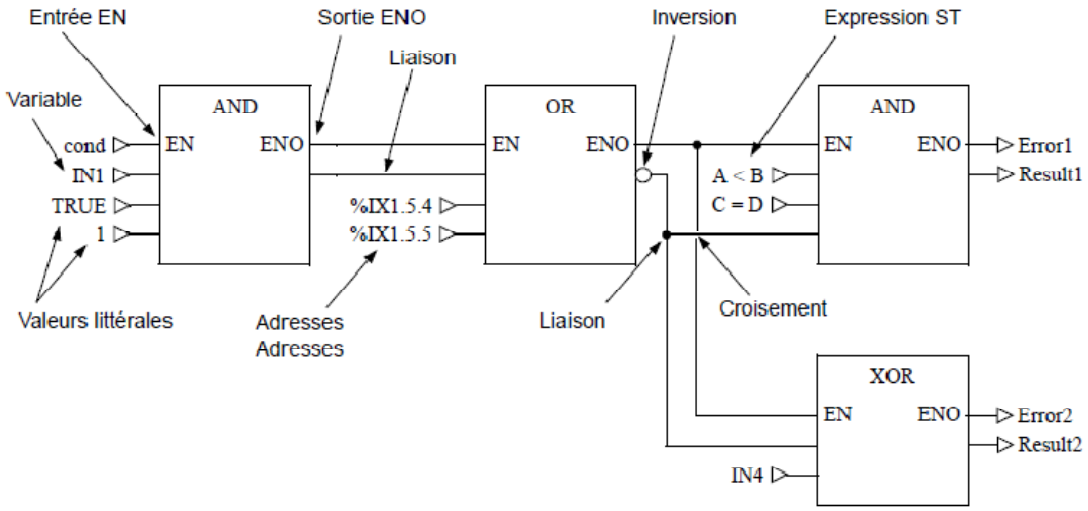


Figure II.18 : *Représentation d'une section FBD* [9]

Les objets du langage FBD (diagramme de blocs fonctionnels) offrent des aides permettant de structurer une section en un ensemble de :

➢ EF et EFB fonctions élémentaires et blocs fonction élémentaires

➢ DFB (blocs fonction dérivés)

➢ procédures

- contrôles.

Ces objets,(Fonctions élémentaires, blocs fonction élémentaires, blocs fonction dérivés et procédures) regroupés sous l'abréviation générique FFB, peuvent être liés les uns aux autres par :

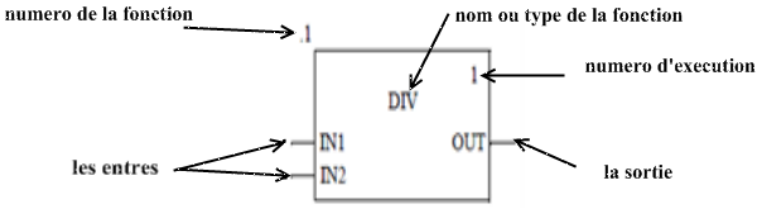
- des liaisons
- des paramètres réels.

Les différents objets du langage DFB

1) Fonction élémentaire :

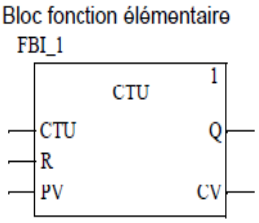
Une fonction élémentaire est représentée graphiquement sous forme de cadre avec des entrées et une sortie.

- Le nom de la fonction, c'est-à-dire le type de fonction, est affiché au centre du cadre.
- Le numéro d'exécution de la fonction apparaît à droite du type de fonction.



2) Bloc fonction élémentaire :

Les blocs fonction élémentaires (EFB) ont des états internes. Pour des valeurs égales aux entrées, la valeur à la sortie peut être différente pour toutes les exécutions de la fonction. Par exemple, pour un compteur, la valeur à la sortie augmente.



Les blocs fonction peuvent avoir plusieurs sorties.

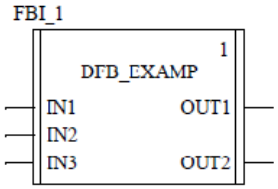
3) Les blocs fonction dérivés :

Les blocs fonction dérivés (DFB) ont les mêmes caractéristiques que les blocs

fonction élémentaires. Ils sont cependant créés par l'utilisateur dans les langages

FBD, LD, IL et/ou ST.

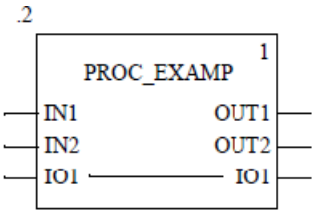
L'unique différence par rapport aux blocs fonction élémentaires est que le bloc fonction dérivé est représenté graphiquement sous forme de cadre avec deux lignes verticales.



4) Procédure :

Techniquement, les procédures sont des fonctions.

L'unique différence par rapport aux fonctions élémentaires est que les procédures peuvent comprendre plus d'une sortie et qu'elles prennent en charge le type de données VAR_IN_OUT.



Il n'y a pas de différence physique entre les procédures et les fonctions élémentaires

Ordre d'exécution des FFB : L'ordre d'exécution est défini par la position des FFB dans la section (exécution de gauche à droite et de haut en bas). Lorsque, par la suite, les FFB sont liés à des liaisons graphiques, l'ordre d'exécution est alors déterminé par le flux de signaux.

Le numéro d'exécution (numéro figurant dans le coin supérieur droit du cadre de FFB) indique l'ordre d'exécution.

Exemple :

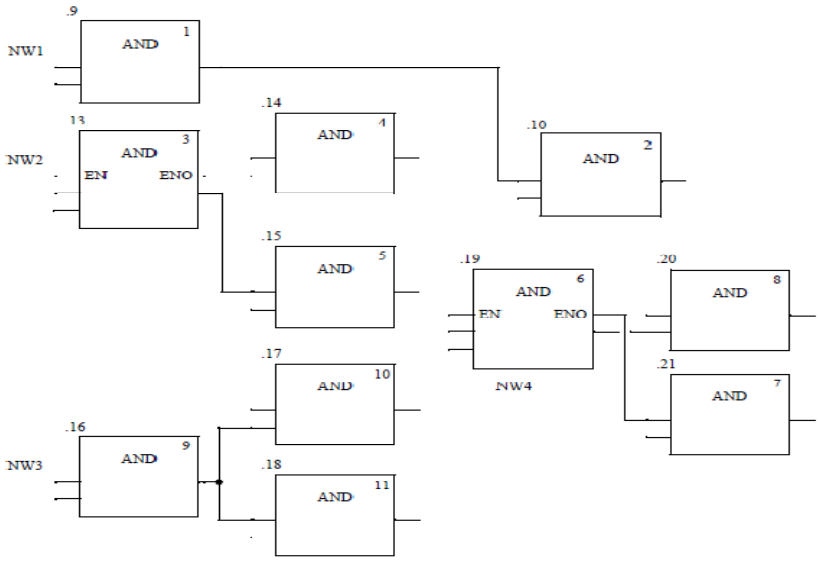


Figure II.19 : *Exemple de l'ordre d'exécution d'objets dans une section FBD* [9]

II.7.2 Langage à contacts (Ladder)

Le langage à contacts ou bien le Ladder (LD) est un langage graphique. Il permet la transcription des schémas à relais, il est excellent pour le combinatoire et les problématiques d'interverrouillage.

Il utilise les symboles graphiques standards :

- Contacts,
- Bobine,
- Blocs (EF , EFB ,DFB , procédures, contrôles et blocs d'opération et de comparaison)

Ces objets peuvent être liés les uns aux autres par :

- des liaisons .
- des paramètres réels (FFB uniquement).

Une section de programme écrite en langage à contacts se compose d'une suite de réseaux de contacts exécutés séquentiellement par l'automate.

Les objets du langage LD :

Contacts :

Un contact est un élément LD permettant de transférer un état de la liaison horizontale vers la droite.

Cet état est le résultat d'une opération booléenne AND sur l'état de la liaison horizontale de gauche avec l'état du aramètre booléen réel associé.

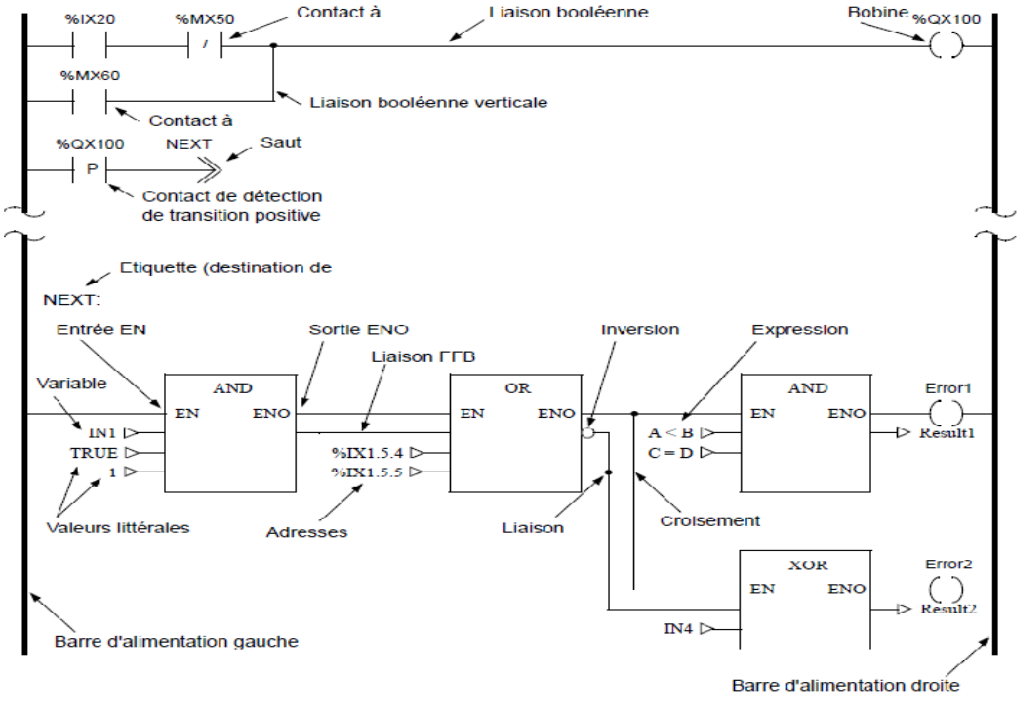


Figure II.20: *Représentation d'une section en langage contact* [9]

Ordre d'exécution des réseaux :

l'ordre d'exécution de réseaux reliés ensemble uniquement par la barre gauche d'alimentation est déterminé par l'ordre graphique (du haut vers le bas) dans lequel ces réseaux sont reliés à la barre gauche d'alimentation. Cela ne s'applique pas si l'ordre a été influencé par des éléments de commande.

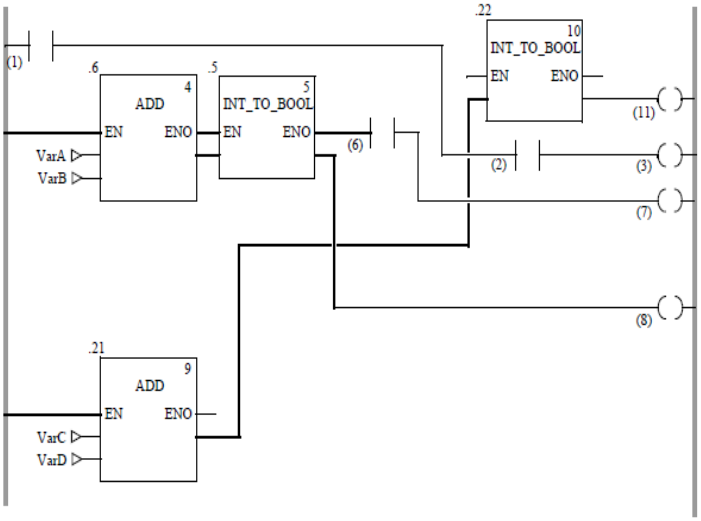


Figure II.21: *Exemple de l'ordre d'exécution des objets dans une section LD* [9]

II.7.3 Langage séquentiel SFC

le langage séquentiel SFC c'est un langage à Diagramme fonctionnel en séquence.

Un diagramme fonctionnel en séquence conforme à CEI se compose dans Unity Pro de sections SFC (niveau supérieur), de sections de transition et de sections d'action.

Ces sections SFC ne sont admises que dans la tâche maître du projet. Dans d'autres tâches ou DFB, les sections SFC ne peuvent pas être utilisées.

Chaque section SFC contient exactement un réseau SFC (séquence) dans le jeton unique.

Les jetons multiples d'une section SFC peuvent contenir un ou plusieurs réseaux SFC indépendants les uns des autres.

Objets :

Une section SFC contient les objets de création de programme suivants :

- Étape
- macro étape (séquence de sous-étapes imbriquées)
- transition (condition de transition)
- saut
- liaison

- divergence en OU
- convergence en OU
- divergence en ET
- convergence en ET

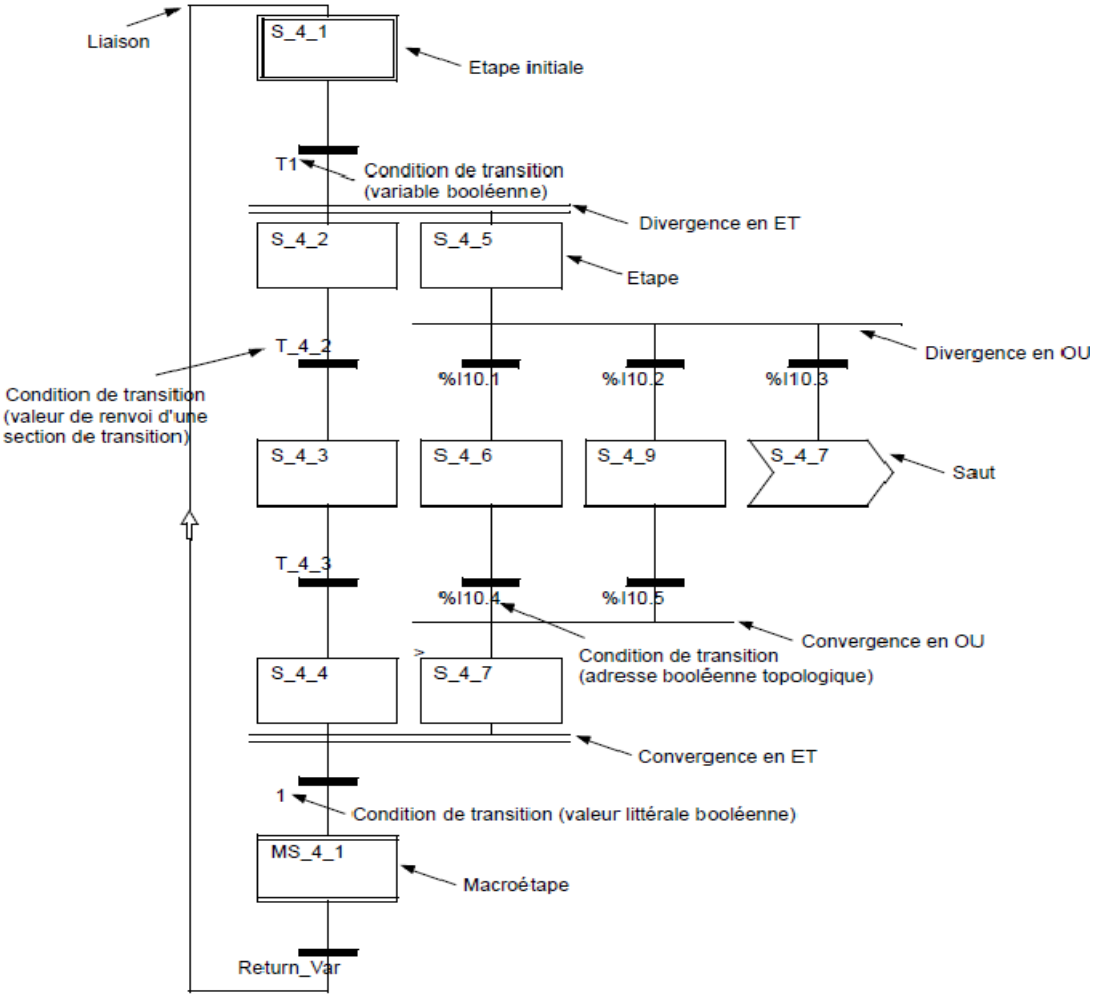


Figure II.22: *Représentation d'une section SFC* [9]

II.7.4 Langage Liste d'instructions (IL)

Le langage de programmation Liste d'instructions (IL) vous permet par exemple d'appeler des fonctions et des blocs fonction de façon conditionnelle ou inconditionnelle, de lancer des affectations et d'exécuter des sauts conditionnels ou inconditionnels au sein de la section.[10]

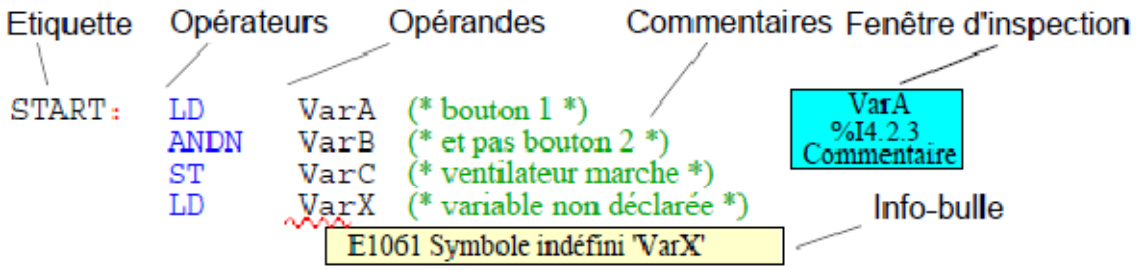


Figure II.23: *Représentation d'une section IL* [10]

II.7.5 Langage Texte structuré (ST)

Présentation

Le langage Littéral structuré (ST) vous permet par exemple d'appeler des blocs fonction, d'exécuter des fonctions, de lancer des affectations, d'exécuter des instructions conditionnelles et de répéter des instructions.

Expression : Le langage ST utilise ce que l'on appelle des "expressions".Les expressions sont des constructions comprenant opérateurs et opérandes qui livrent une valeur lors de leur exécution.

Opérateur : Les opérateurs sont des symboles pour les opérations à exécuter.

Opérande : Les opérateurs sont utilisés sur les opérandes. Les opérandes sont par exemple des variables, des valeurs littérales, des entrées/sorties FFB, etc.

Instructions : Les instructions permettent d'affecter les valeurs retournées par des expressions à des paramètres réels et de structurer et commander les expressions.

Taille de la section

La taille d'une ligne d'instruction est limitée à 300 caractères.

La longueur d'une section ST n'est pas limitée au sein de l'environnement de programmation. La longueur d'une section ST n'est limitée que par la taille de la mémoire de l'automate.

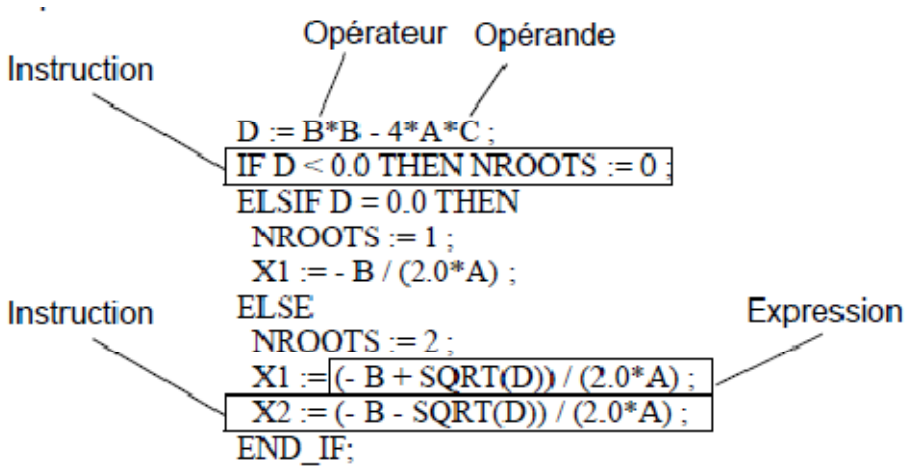


Figure II.24: *Représentation d'une section ST* [8]

Les instructions

Le littéral structuré utilisé aussi des instructions :

- le contrôle conditionnel IF…THEN…END_IF,
- le contrôle conditionnel REPEAT…END_REPEAT,
- le contrôle conditionnel WHILE…END_WHILE,
- le contrôle conditionnel FOR…END_FOR,

II.8 Mise en œuvre d'une application sous Unity Pro

Cette annexe décrit la mise en œuvre d'une application basée sur l'utilisation des différents types de variables, de langages de programmation et d'un écran d'exploitation décrivant son fonctionnement.

II.8.1 Présentation de l'application

L'application consiste à gérer le niveau d'un liquide dans une cuve. Le remplissage de la cuve se fait par l'intermédiaire d'une pompe et la vidange est gérée par une vanne. Les différents niveaux de la cuve sont mesurés par des capteurs disposés sur la cuve. Le volume de la cuve est donné par un afficheur numérique. Les moyens de contrôle du fonctionnement de l'application sont basés sur un écran d'exploitation qui doit fournir l'état des différents capteurs, actionneurs et le volume de la cuve. Suivant l'état du niveau de la cuve et de l'application il faut avertir l'utilisateur par des alarmes et archiver les informations nécessaires à chaque déclenchement.

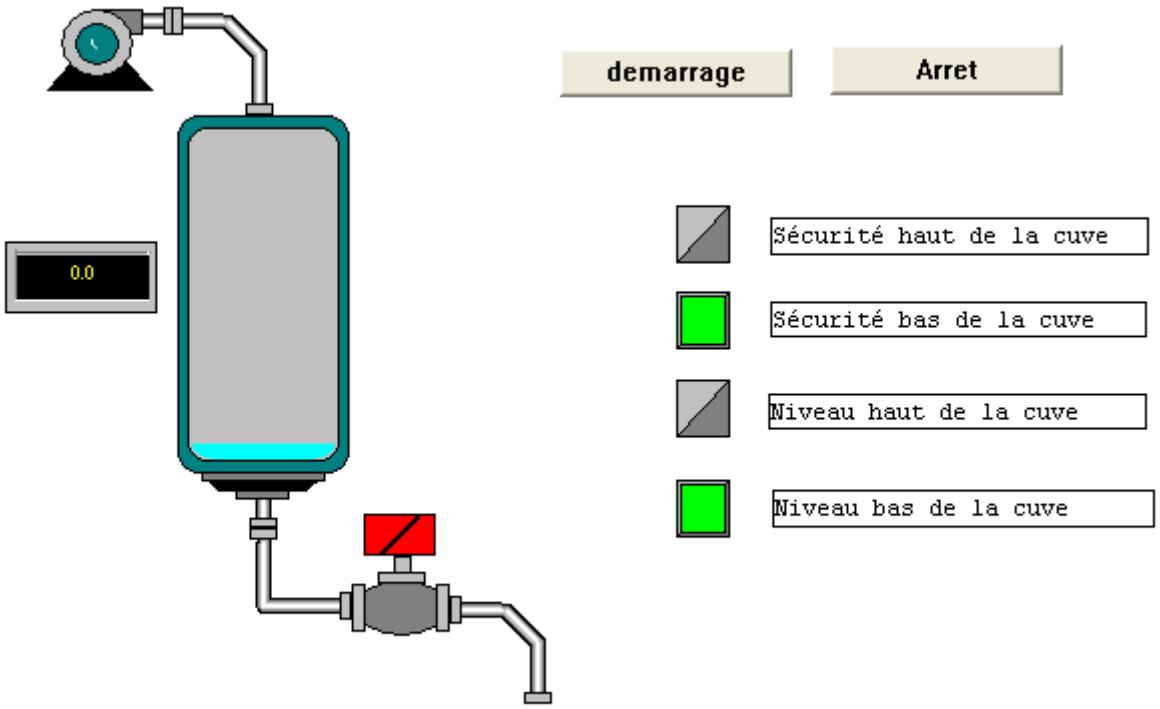


Figure II.25 *vue générale de l'écran d'exploitation* [9]

II.8.2 Mode de marche

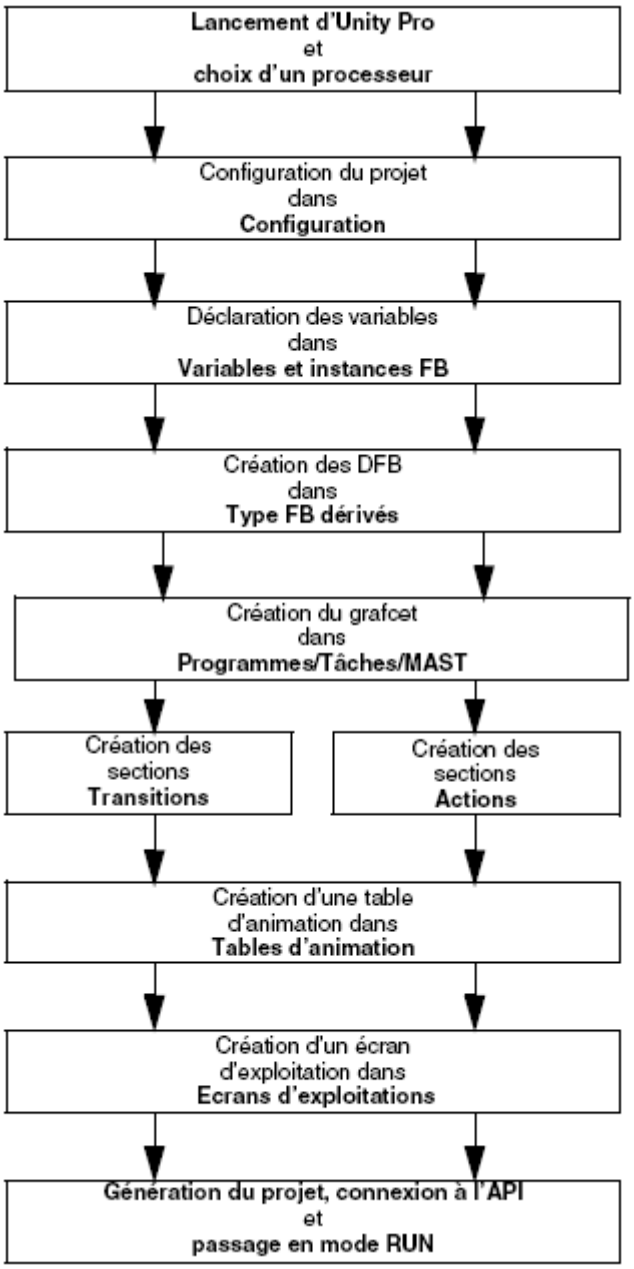
Le mode de marche est le suivant :

- un bouton Démarrage cycle permet de lancer les cycles de remplissage,
- lorsque le niveau haut de la cuve est atteint la pompe s'arrête et la vanne s'ouvre. Lorsque le niveau bas de la cuve est atteint, la vanne se ferme et la pompe se met en marche jusqu'a atteindre le niveau haut.
- un bouton Arrêt cycle permet d'interrompre les cycles de remplissage. Une action sur ce bouton permet de mettre le système en sécurité. La pompe s'arrête, la vanne s'ouvre jusqu'à atteindre le niveau "Sécurité bas" (cuve vide). La vanne se ferme et le cycle s'arrête.
- la pompe a un débit variable, la valeur de ce débit pourra être accessible par l'écran d'exploitation. Le débit de la vanne est égal à celui de la pompe.
- des sécurités doivent être mises en place.
- perte du niveau haut de la cuve : un autre niveau dit "Sécurité haut" se déclenche, le système se met en sécurité. Dans ce cas, la pompe s'arrête, la vanne s'ouvre jusqu'à atteindre le niveau "Sécurité bas" (cuve vide). La vanne se ferme et le cycle s'arrête.

- perte du niveau bas de la cuve : un autre niveau dit "Sécurité bas" se déclenche, le système se met en sécurité. Dans ce cas, la vanne se ferme et le cycle s'arrête.
- pour les deux sécurités, il faut afficher un message de défaut.
- les temps d'ouverture et de fermeture de la vanne sont surveillés, un message de défaut est affiché en cas de dépassement.

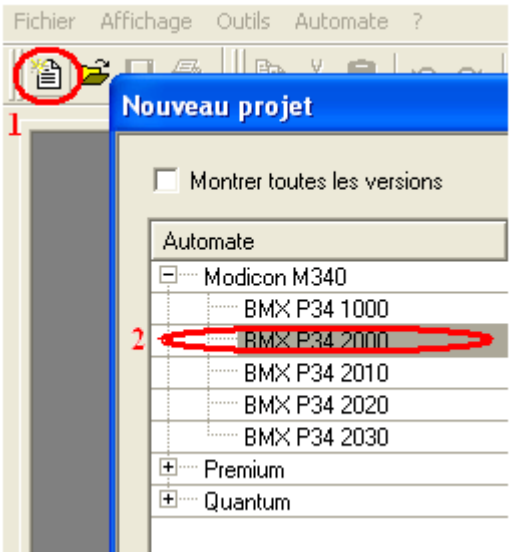
II.8.3 Les différentes étapes du process dans Unity Pro

Le logigramme ci-dessous est destiné à donner les différentes étapes à suivre pour créer l'application. Un ordre chronologique doit être respecté afin de définir correctement tous les éléments de l'application.



Création du projet

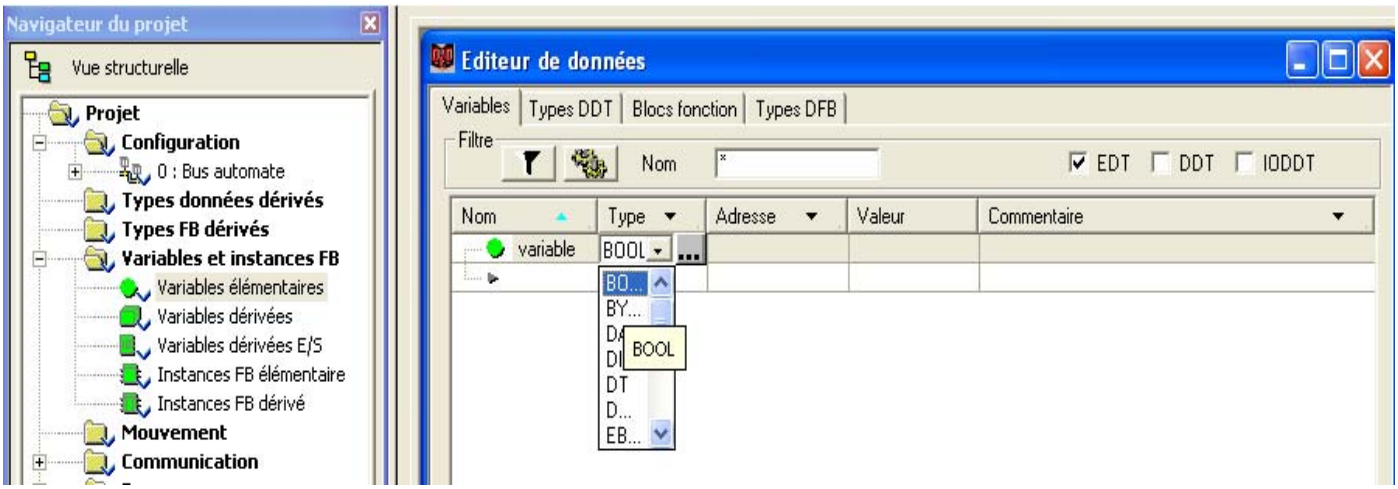
Le développement d'une application sous Unity Pro passe par la création d'un projet associé à un automate : Lancez le logiciel Unity Pro, Cliquez sur Nouveau puis choisissez un automate,



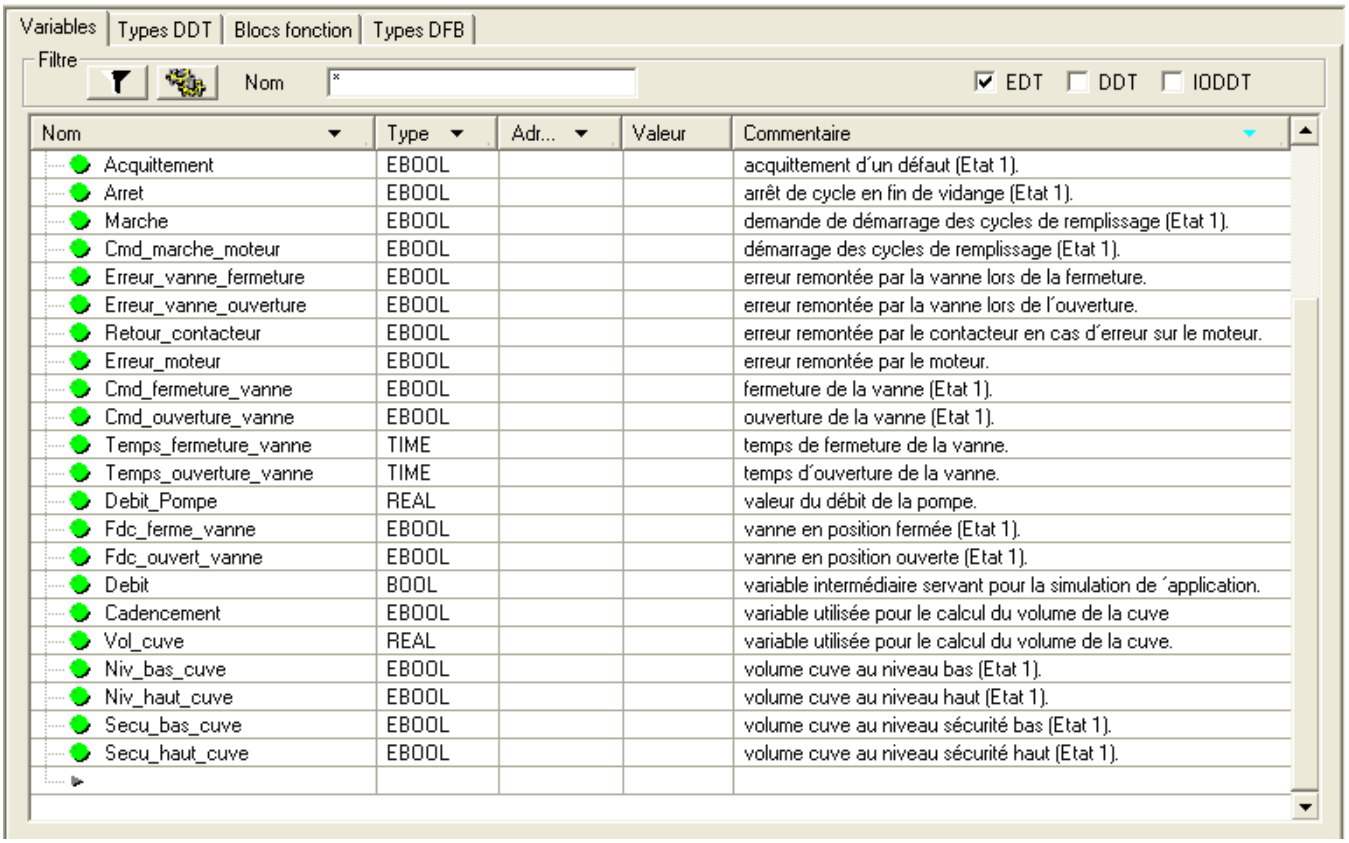
Déclaration des variables

Dans le Navigateur de projet \ Variables et instances FB, double cliquez sur Variables élémentaires. Dans la fenêtre Editeur de données sélectionnez la case dans la colonne Nom puis entrez le nom de votre première

variable. Ensuite, sélectionnez le Type de cette variable. Lorsque toutes vos variables sont déclarées, vous pouvez fermer la fenêtre.



Variables utilisées pour l'application



Nom	Type	Adr...	Valeur	Commentaire
Acquittement	EBOOL			acquittement d'un défaut (Etat 1).
Arret	EBOOL			arrêt de cycle en fin de vidange (Etat 1).
Marche	EBOOL			demande de démarrage des cycles de remplissage (Etat 1).
Cmd_marche_moteur	EBOOL			démarrage des cycles de remplissage (Etat 1).
Erreur_vanne_fermeture	EBOOL			erreur remontée par la vanne lors de la fermeture.
Erreur_vanne_ouverture	EBOOL			erreur remontée par la vanne lors de l'ouverture.
Retour_contacteur	EBOOL			erreur remontée par le contacteur en cas d'erreur sur le moteur.
Erreur_moteur	EBOOL			erreur remontée par le moteur.
Cmd_fermeture_vanne	EBOOL			fermeture de la vanne (Etat 1).
Cmd_ouverture_vanne	EBOOL			ouverture de la vanne (Etat 1).
Temps_fermeture_vanne	TIME			temps de fermeture de la vanne.
Temps_ouverture_vanne	TIME			temps d'ouverture de la vanne.
Debit_Pompe	REAL			valeur du débit de la pompe.
Fdc_ferme_vanne	EBOOL			vanne en position fermée (Etat 1).
Fdc_ouvert_vanne	EBOOL			vanne en position ouverte (Etat 1).
Debit	BOOL			variable intermédiaire servant pour la simulation de 'application.
Cadencement	EBOOL			variable utilisée pour le calcul du volume de la cuve
Vol_cuve	REAL			variable utilisée pour le calcul du volume de la cuve.
Niv_bas_cuve	EBOOL			volume cuve au niveau bas (Etat 1).
Niv_haut_cuve	EBOOL			volume cuve au niveau haut (Etat 1).
Secu_bas_cuve	EBOOL			volume cuve au niveau sécurité bas (Etat 1).
Secu_haut_cuve	EBOOL			volume cuve au niveau sécurité haut (Etat 1).

II.8.4 Création et utilisation des DFB

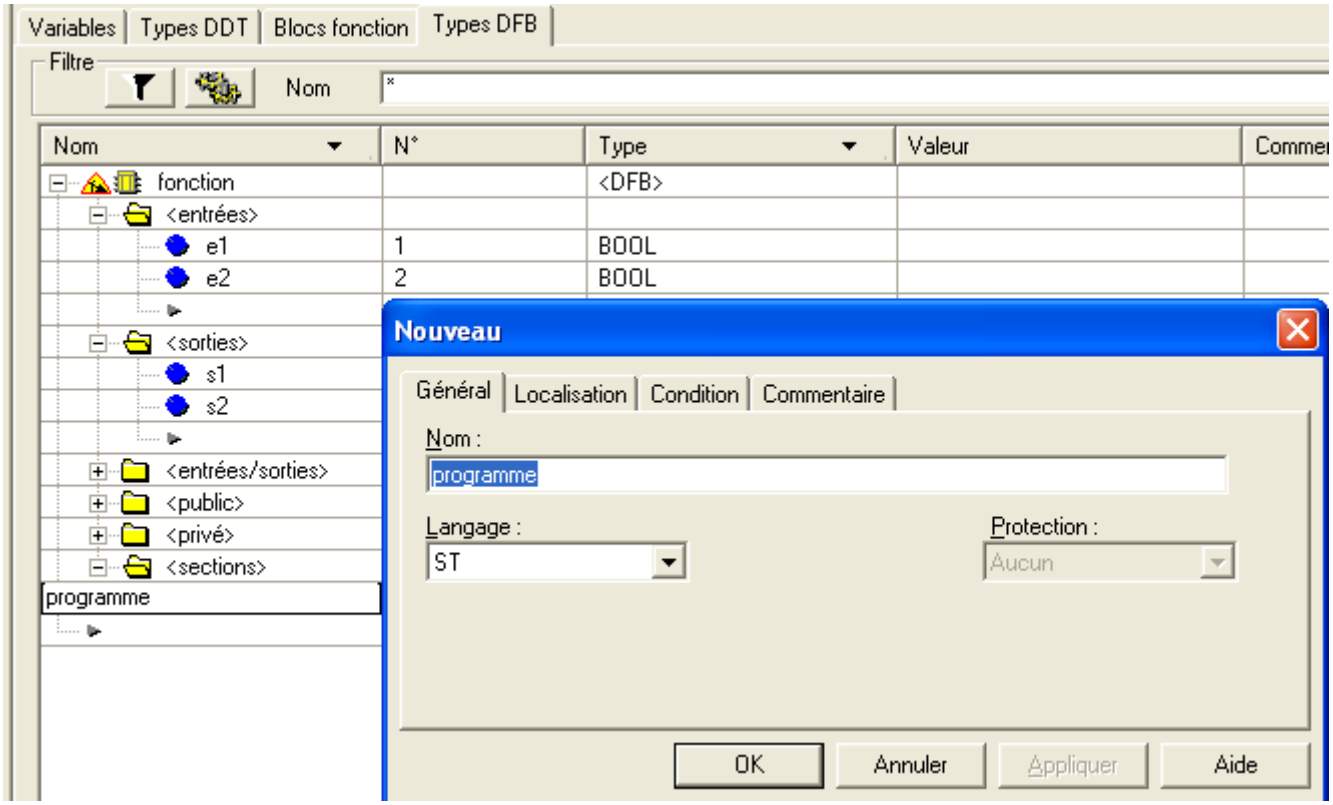
Les types DFB sont des blocs fonction programmables par l'utilisateur en langage ST, IL, LD ou FBD. Notre application doit utiliser un DFB moteur et un DFB Vanne. Nous allons utiliser également des

DFB existants dans la bibliothèque pour surveiller des variables. Notamment, les variables "sécurité" pour le niveau de la cuve et les variables "erreur" remontées par la vanne. L'état de ces variables sera visible dans Visualisation du diagnostic.

Marche à suivre pour créer un DFB

Dans le Navigateur de projet, faîtes un clic droit sur Types FB dérivés puis choisissez Ouvrir. Dans la fenêtre Editeur de données sélectionnez la case dans la colonne Nom puis entrez le nom de votre DFB et validez par Entrée. Le nom de votre DFB apparaît avec le signe "Travaux" (DFB non analysé).

Ouvrez la structure de votre DFB (voir la figure ci-dessous) puis ajoutez les entrées, sorties et les autres variables propres à votre DFB. Lorsque vos variables du DFB sont déclarées, analysez votre DFB (le signe "Travaux" doit disparaitre). Pour analyser votre DFB, sélectionnez le DFB et cliquez dans le menu Génération puis sur Analyser. Vous venez de créer les variables de votre DFB, il faut maintenant créer la section associée. Donnez un nom à votre section puis choisissez le type de langage et validez par OK. Editez votre section en utilisant les variables déclarées

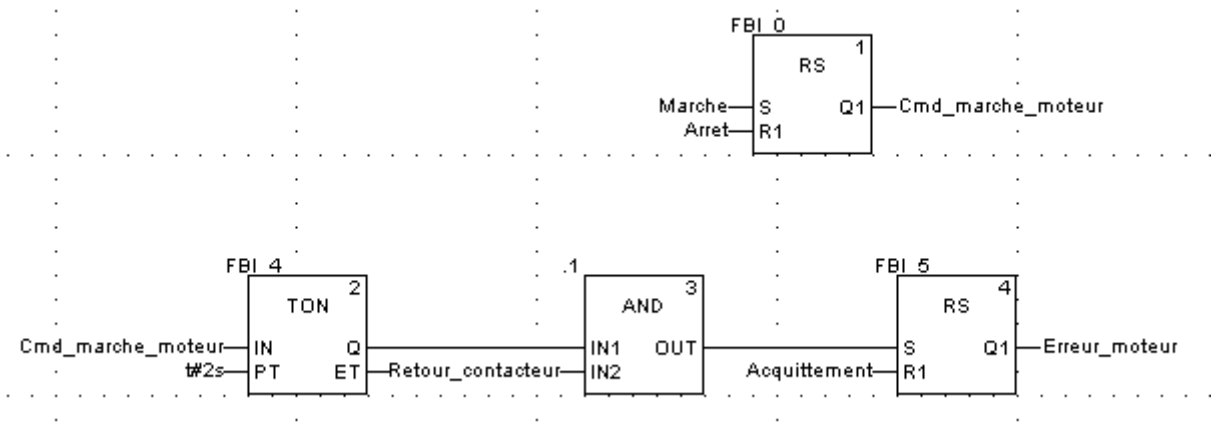


Voici la liste des différentes entrées sorties de la DFB moteur :

<entrées>				
Marche	1	BOOL		commande du démarrage du moteur.
Arret	2	BOOL		commande de l´arrêt du moteur.
Retour_contacteur	3	BOOL		retour du contacteur en cas de problème de démarrage du mote...
Acquittement	4	BOOL		acquittement de la variable de sortie erreur_moteur.
<sorties>				
Cmd_marche_moteur	1	BOOL		démarrage du moteur.
Erreur_moteur	2	BOOL		erreur moteur

Cette DFB réalise les taches suivantes :

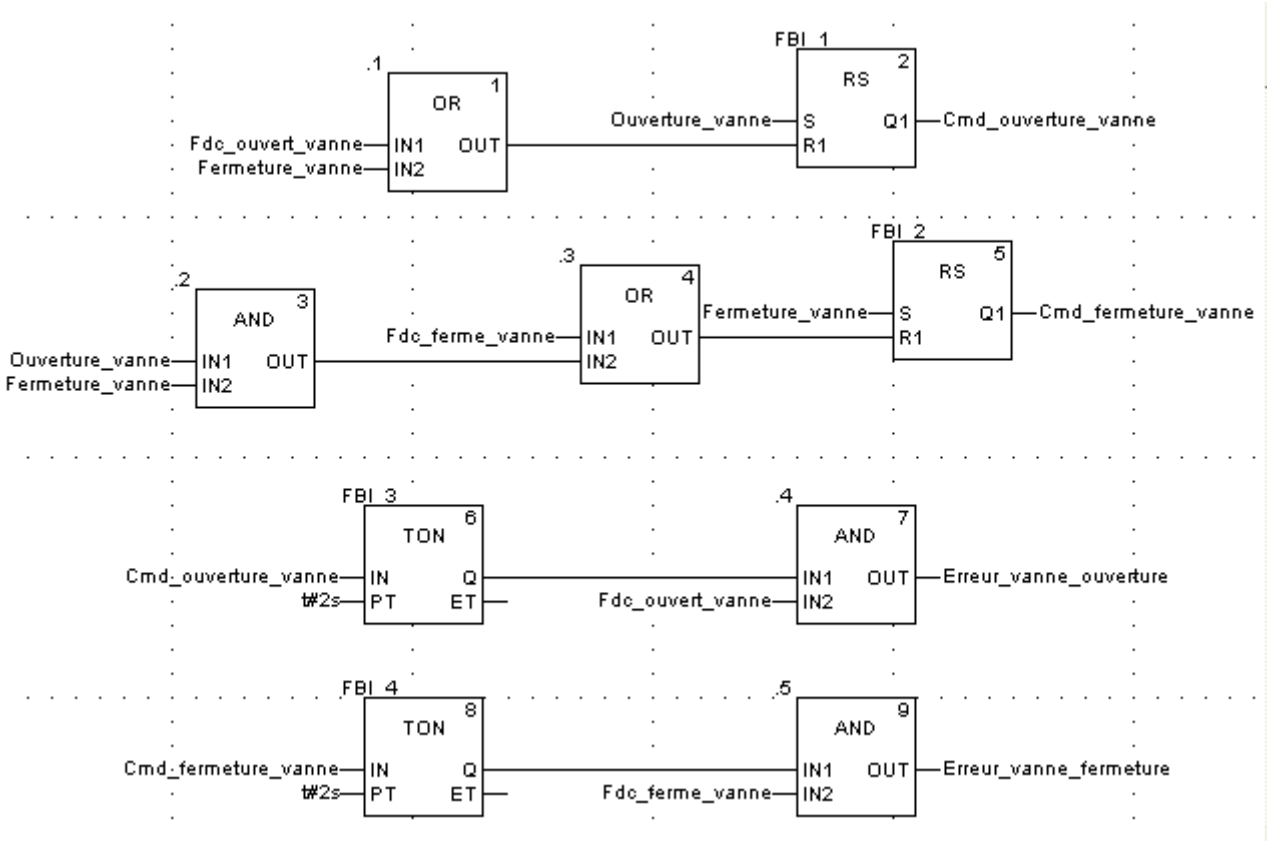
Lorsque Marche = 1et Arrêt = 0, on peut commander le moteur (Cmd_marche_moteur = 1). L'autre partie surveille la variable Retour_contacteur. Si Retour_contacteur n'est pas à "1" avant les deux secondes décompter par le compteur TON, la sortie Erreur_moteur passe à "1".



Voici la liste des différentes entrées sorties de la vanne :

DFB_vanne		<DFB>		
<entrées>				
Ouverture_vanne	1	BOOL		commande de l´ouverture de la vanne.
Fermeture_vanne	2	BOOL		commande de la fermeture de la vanne.
Fdc_ouvert_vanne	3	BOOL		état du fin de course de la vanne.
Fdc_ferme_vanne	4	BOOL		état du fin de course de la vanne.
Acquittement	5	BOOL		acquittement des variables Erreur_fermeture_vanne ou Erreur_...
<sorties>				
Cmd_ouverture_vanne	1	BOOL		ouverture de la vanne.
Cmd_fermeture_vanne	2	BOOL		fermeture de la vanne.
Erreur_vanne_ouverture	3	BOOL		erreur ouverture
Erreur_vanne_fermeture	4	BOOL		erreyr fermeture

Ce DFB autorise la commande de l'ouverture de la vanne (Cmd_ouverture_vanne) lorsque les entrées Fermeture_vanne et Fdc_ouvert_vanne sont à "0". Le principe est le même pour la fermeture avec une sécurité supplémentaires si on demande la fermeture et l'ouverture de la vanne en même temps (priorité sur l'ouverture). Afin de surveiller le temps d'ouverture et de fermeture on utilise le temporisateur TON pour retarder le déclenchement d'un défaut. Dés que l'ouverture de la vanne est validée (Cmd_ouverture_vanne = 1) le temporisateur se déclenche. Si dans les deux secondes Fdc_ouvert_vanne n'est pas à "1" la variable de sortie Erreur_vanne_ouverture monte à "1". Dans ce cas un message sera affiché.



II.8.5 Création du programme en SFC pour la gestion de la cuve

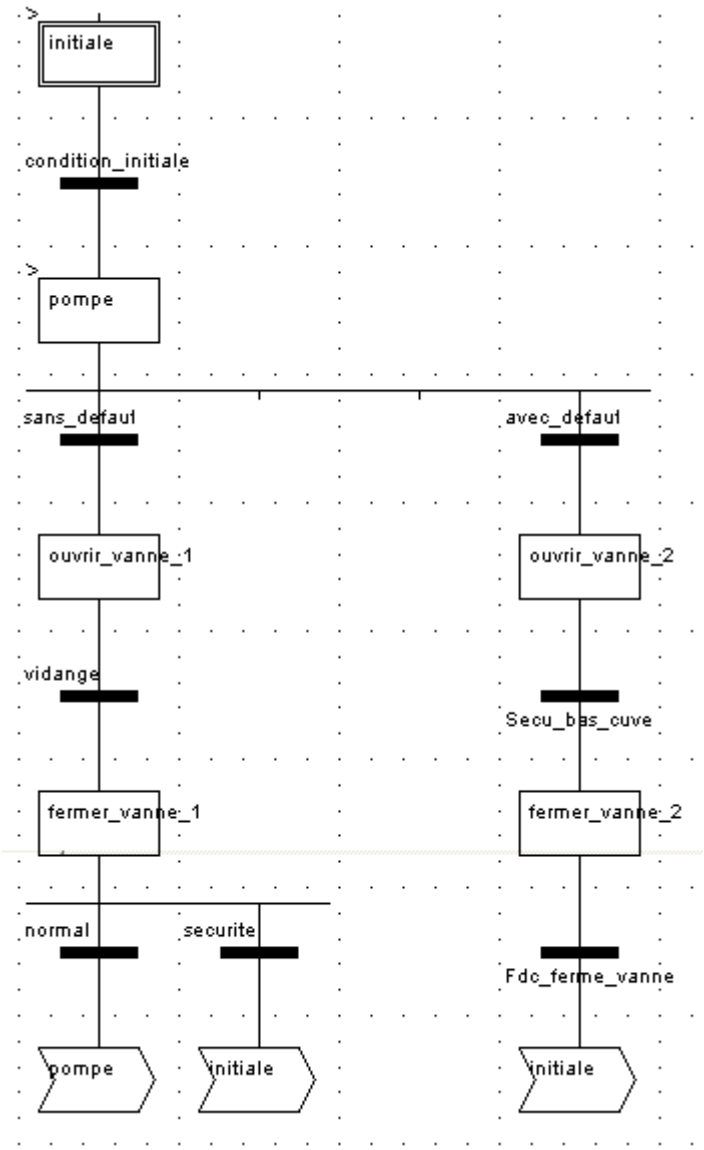
Le programme principal est écrit en SFC (Grafcet). Ce programme est déclaré dans une tâche MAST, il sera dépendant de l'état d'une variable booléenne. Le principal avantage du langage SFC est de pouvoir suivre en temps réel l'exécution de l'application grâce à son animation graphique.

Plusieurs sections sont déclarées dans la tâche MAST :

- la section Gestion_cuve écrite en SFC décrivant le mode opératoire,

- la section Application écrite en LD exécutant le démarrage de la pompe en utilisant le DFB moteur, la fermeture et l'ouverture de la vanne.
- la section Simulation écrite en LD simulant l'application, cette section est à supprimer dans le cas d'une connexion à un automate.
- la section Diagnostic pour le diagnostic de l'application écrite en FBD pour remonter les erreurs de l'application au visualisateur de diagnostic.

L'écran ci-dessous représente le grafcet de l'application :



Marche à suivre pour créer la section SFC

Dans le Navigateur de projet \Programme\ Tâches, double-cliquez sur MAST.

Faîtes un clic droit sur Section puis choisissez Nouvelle section. Donnez un nom à votre section puis sélectionnez le langage SFC. Le nom de votre section apparaît, vous pouvez l'éditer en double-cliquant dessus.

Les outils d'édition du SFC apparaissent dans la fenêtre, vous pouvez ainsi créer votre grafcet.

Exemple pour créer une étape avec une transition :

Pour créer une étape, cliquez sur puis placez-la dans l'éditeur, pour créer une transition, cliquez sur puis placez-la dans l'éditeur (généralement sous l'étape qui la précède).

Description de la section Gestion_cuve

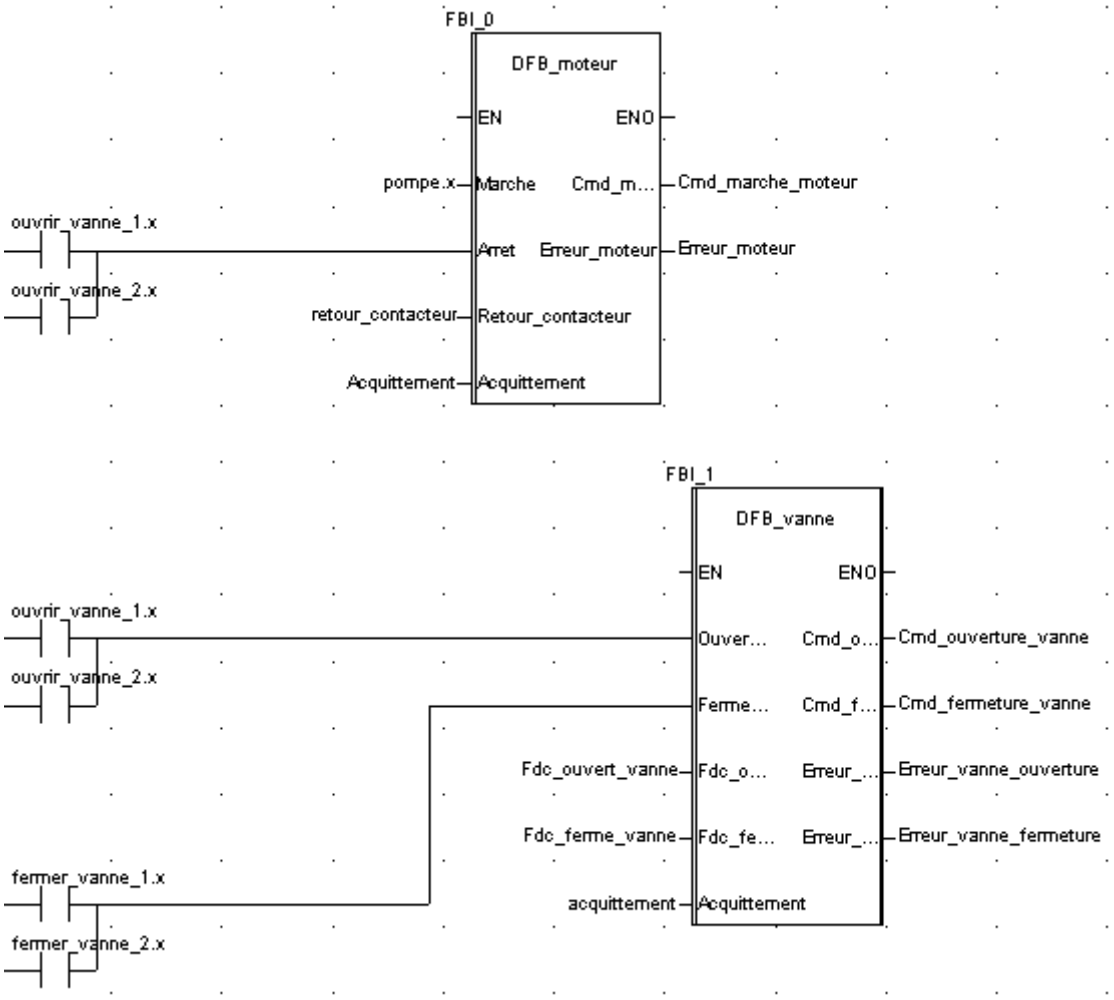
Le tableau suivant décrit les différents étapes et transitions du grafcet Gestion_cuve

Etape / Transition	**Description**
Initiale	C'est l'étape initiale.
Condition_initiale	C'est la transition qui va activer le démarrage de la pompe. La transition sera valide lorsque les variables : • Arret_cycle = 0, • Marche_cycle = 1, • Secu_haut_cuve = 0, • Fdc_ferme_vanne = 1
Pompe	C'est l'étape de démarrage de la pompe et de remplissage de la cuve jusqu'au niveau haut. Cette étape activera l'entrée du DFB moteur dans la section Application pour commander le démarrage de la pompe.
Sans_defaut	Cette transition est active lorsque le niveau haut de la cuve est atteint et que le niveau sécurité haut est à 0.
Ouvrir_vanne1	C'est l'étape de vidange de la cuve et d'ouverture de la vanne. Cette étape activera l'entrée du DFB vanne dans la section Application pour commander l'ouverture de la vanne.
Vidange	Cette transition est active lorsque le niveau bas de la cuve ou le niveau sécurité bas est à 1.
Fermer_vanne1	C'est l'étape de fermeture de la vanne. Cette étape activera l'entrée du DFB vanne dans la section Application pour commander la fermeture de la vanne.
Normal	Cette transition est valide lorsque le niveau bas de la cuve et Fdc_ferme_vanne sont à 1. Dans ce cas on fait un saut vers l'étape S_1_2.
Sécurité	Cette transition est valide lorsque la sécurité niveau bas de la cuve et Fdc_ferme_vanne sont à 1. Dans ce cas on revient en début de cycle et on attend une initialisation de la variable sécurité et un redémarrage du cycle.
Avec_defaut	Cette transition est active lorsque la Securite niveau haut de la cuve est atteinte ou que le bouton Arret_cycle a été activé (Arret_cycle = 1).
Ouvrir_vanne2	Cette étape est identique à Ouvrir_vane1.
Securite_bas_cuve	Cette transition est active lorsque la sécurite basse de la cuve est à 1 (après une vidange de la cuve suite à un arrêt du cycle ou à une activation de la sécurite haute de la cuve).

Fermer_vanne2	Cette étape est identique à la Fermer_vanne1.
Fdc_ferme_vanne	Cette transition est valide lorsque le Fdc_ferme_vanne est à 1. Dans ce cas on revient en début de cycle et on attend une initialisation de la variable sécurité et un redémarrage du cycle.

II.8.5 Création du programme en LD pour l'exécution de l'application

La section ci-dessous fait partie de la tâche MAST. Elle n'a pas de condition, donc elle est exécutée en permanence :



Description de la section Application

- lorsque l'étape Pompe est active, l'entrée Marche du DFB moteur est à 1, si l'entrée Arrêt du DFB moteur est à 0, la sortie Cmd_marche_moteur passe à "1" et la pompe est alimentée.

- même principe pour les étapes Ouvrir_vanne1 et Ouvrir_vanne2 et pour le reste de la section.

Marche à suivre pour créer la section LD

Dans le Navigateur de projet\Programme\Tâches, double-cliquez sur MAST. Faîtes un clic droit sur Section puis choisissez Nouvelle section. Donnez le nom Application à cette section puis choisissez le langage LD. La fenêtre d'édition s'ouvre. Pour créer le contact Ouvrir_vanne1.x, cliquez sur puis placez la dans l'éditeur. Double-cliquez sur ce contact puis écrivez le nom de l'étape avec un

".x" à la fin (signifiant une étape d'une section SFC) et validez par OK.

Pour utiliser le DFB Moteur il faut l'instancier. Faites un clic droit dans l'éditeur puis cliquez sur Sélection de données et sur . Cliquez sur l'onglet Fonction et types de bloc fonction et sélectionnez votre DFB puis validez par OK et placez votre DFB. Pour relier le contact Ouvrir_vanne1.x à l'entrée Arrêt du DFB, alignez horizontalement le contact et l'entrée, enfin cliquez sur et placez-le entre le contact et l'entrée.

(les parties simulation, diagnostique et création des tables et des écrans d'exploitation sont décrite dans l'annexe B)

II.9 conclusion

Intégrant la totalité des langages et les outils nécessaires pour une programmation complète et souple, la gamme des logiciels Unity Pro offre une facilité d'utilisation et de compréhension.

Chapitre III
Description de la station de pompage

III.1 Introduction

En général, on essaie de faire véhiculer les eaux gravitairement, si éventuellement la topographie et la nature du terrain le permettent, parfois cette solution devient difficile à cause de certaines contraintes topographiques et géotechniques (exemples : terrains accidentés ou trop plats etc.…). Donc on fait appel à des stations de pompage qui sont des installations, y compris des pompes et d'équipements pour le pompage de liquides d'un endroit à l'autre. Notre objective et de faire l'automatisation d'une telle station .

III.2 Notre objectif

Le but c'est d'automatisé un système de pompage d'eau potable de la Banlieue Est d'Alger qui se situé à Kouba (poste 107). La station est conçue pour un réservoir de stockage d´aspiration de pompe intermédiaire. La pompe fonctionne à 1300 m /h respectivement 2600 m /h avec une pression de refoulement de 58 m. Sur le collecteur de refoulement principal des pompes, un réservoir anti-bélier protège les pompes en cas d´une éventuel vague de pression.

Les vannes de décharge des pompes sont motorisées, et les vannes d´aspiration sont manuelles. Les moteurs des pompes sont munis de démarreur progressif et de variateur de vitesse. Un débitmètre, installé sur la ligne d´aspiration qui indique et enregistre le débit en temps réel quand une ou plusieurs pompes sont en service.

De la ligne d´alimentation de Garidi, il y a deux lignes (l'une principale et l'autre de secours) qui alimentent la station de pompage en eau. Une ligne DN 1000 séparée par une vanne alimente le réservoir(R100) et l'autre ligne DN 700 séparée avec une vanne qui peut alimenter directement le collecteur d´aspiration de la pompe (système By Pass).

Le niveau du réservoir R100 intermédiaire est contrôlé par un indicateur de niveau à ultrasons LC 01 et des capteurs de l'état du niveau (niveau Bas :LLL, niveau Moyen : LLA, niveau Haut :HL). Sur l´écran de la salle de contrôle, l´opérateur voit le niveau du réservoir en temps réel qui est sera schématisé par une jauge du niveau du réservoir R100, indiquant le contenu de 0 à 100%. De surcroît, un flotteur arrête les pompes au niveau très bas (indiqué par le capteur à ultrason) et informe l´opérateur par une alarme sonore et visuel sur l'écran de supervision quand le niveau haut de (HL) est aussi atteint.

Le débit d´eau vers le réservoir est contrôlé par une vanne altimétrique. Un pilote de niveau haut et un pilote de niveau bas sont ajustés pour une ouverture réglable. Le remplissage du réservoir est effectué à

partir du fond. Le réservoir est équipé d´un trop-plein et d´une vanne de drainage. La sortie du réservoir DN 1000 alimente le collecteur d´aspiration de la pompe.

En cas d´une panne du réseau électrique dans la station de pompage, le groupe électrogène diesel se met en marche pour assurer le fonctionnement d´une pompes afin de garantir la continuité de service en ressources en eau.

Donc l'objectif principale de cette réalisation est :

- Apporter une optimisation de la station de pompage, qui est absorbé en fonction des besoins en temps réel pour un fonctionnement des pompes qui est affiché sur le pupitre de commande.
- le fonctionnement de la station prévois un fonctionnement en mode dégradé en cas de panne sur les éléments d'automatisme ou d'une éventuel pompe (à remplacé ou en cas de maintenance) pour assurer une continuité de service.

III.3 les équipements utilisés

III.3.1 Equipement MT

III.3.1.1 Le câble MT

Le câble MT fait la liaison entre la cellule (déjà existante sur le site) et le transformateur. Sa fonction proprement dit est le transport d'énergie électrique d'un endroit vers un autre. Ce câble sera placé dans la cellule existante jusqu'au transformateur qui lui sera à proximité (c'est-à-dire dans le même local, très peu distant l'un de l'autre)

III.3.1.2 La Cellule de protection Auxiliaire Transformateur

La protection des transformateurs de puissance permettra de préserver notre installation MT, contre tout risque électrique du type (court-circuit, surintensité, etc.). Ces cellules seront placées à proximité du transformateur au coté des cellules existantes. Le nombre de cellules correspond au nombre de source, c'est-à-dire qu'il y deux arrivées (SN1 et SN2).

III.3.1.3 Le transformateur

Le transformateur a pour rôle essentiel d'abaisser ou l'augmenter la tension, ici dans notre installation le transformateur sera d'abaisser la tension donc généralement on dit que c'est un transformateur abaisseur. Il aura du coté primaire (entré du transfo) une tension de 10KV et du coté secondaire (en sorti du transfo) il aura comme valeur une tension de 400V.

Il sera placé au plus proche des cellules SM6 QM et sera équipé de relais Bucholz et de thermomètre à 2 seuils.

Nombre de transformateur : 2 transformateurs (il y aura un transformateur par arrivé réseau, d'ou l'installation dispose de 2 arrivées SN1 et SN2 donc 2 transformateurs pour assuré l'alimentation en énergie électrique du site).

III.3.2 Equipement Electrique BT

III.3.2.1 Tableau Electrique

Enveloppe

Le tableau électrique recouvre les divers organes de commande et de puissance. Elle sera placée dans le local électrique de la station. Il y aura placé sur ses armoires des étiquettes afin de pouvoir les identifiées (arrivées, départ groupe électro-pompe1, 2 et 3, réserve, auxiliaires, compensation).Les armoires sont juxtaposable (extensible), fabriquées à partir d'une feuille d'acier, protégée à l'intérieur et à l'extérieur par une peinture résine polyester époxy et grise.

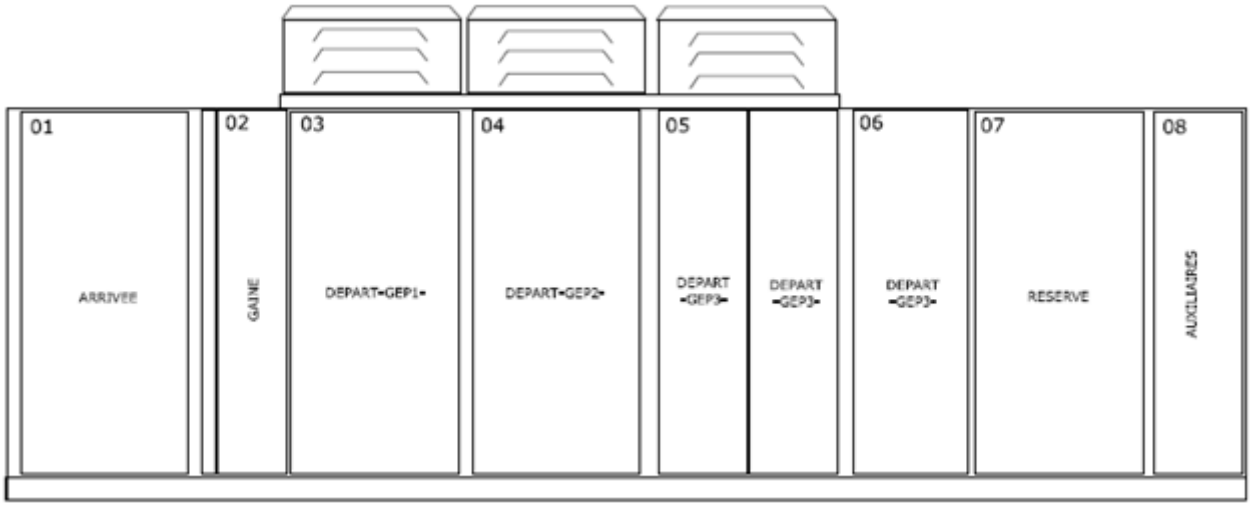


Figure III.1 *Vue de face avant du tableau basse tension* [1]

III.3.2.2 Arrivée générale

- Un inverseur de source avec verrouillage électrique et mécanique, il sera équipé de:
- Deux (02) disjoncteurs de protection Masterpact 1000A 4Pole chacun, ils seront débrochables sur châssis avec des contacts auxiliaires OF, des contacts signal défauts SD, des bobines d'ouverture MX et des bobines de fermeture électrique, ils seront équipés des protections différentielles.
- Un (01) disjoncteur de protection Masterpact 1000A 4Pole, il sera débrochable sur châssis avec un contact auxiliaire OF, un contacts signal défaut SD, une bobine d'ouverture MX et une bobine de fermeture électrique, il sera équipé d'une protection différentielle. .
- Des jeux de barre verticaux et horizontaux supportant 2500A, 4 Pôles, ils seront montés sur des isolateurs.

Dimension: 80 x 20 mm

- Trois (03) Centrales de mesure PM710:
- Sous comptage/allocation des coûts
- Surveillance à distance de l'installation avec un port RS458 Modbus.
- Surveillance des harmoniques (THD)
- Des entrées et des sorties logique.
- Trois (03) transformateurs d’intensité.
- Trois (03) relais de phase pour la protection contre l’inversion des phases, déséquilibre des hase, manque phase, les sous-tension et surtension.

- Un jeu de barre de terre.

III.3.3.2 Armoire Départ 1 et 2

Elles sont aménager de la même façon, cette à dire qu'elles contiennent la partie Puissance ainsi que les Démarreurs progressif (ATS). Les départs groupes électropompes 315 kW chacun avec les démarreurs progressifs électroniques Altistart48, les inductances de ligne et les contacteurs de by-pass pour limiter les dégagements calorifiques.

III.3.2.4 Armoire Départ 3

Cette armoire se compose de la partie Puissance pour le départ 3, où il figure le départ groupe électropompe 355 kW avec le variateur de vitesse Altivar61.

Protection et sécurités du variateur et du moteur :

Protection thermique :

- Contre les échauffements excessifs.
- Protection thermique intégrée dans le variateur par calcule permanent du I^2t.
- Protection par sonde PTC.

Protection contre :

- Les courts-circuits entre les phases moteurs.
- Les coupures des phases moteurs.
- Les surintensités entre les phases de sortie.
- Les surtensions.
- Les sous-tensions.
- Rotor du moteur bloqué.
- Les défauts d'isolement terre.
- Les défauts au niveau des entrées analogique et logique.
- Un dépassement de la vitesse limite.
- Défaut de communication.

Sécurité :

- De surtension et de sous-tension du réseau.
- D'absence de phase réseau, en triphasé.

III.3.2.5 Cellule Réserve

Elle sera vide, c'est-à-dire qu'elle est prévue pour une future extension du site.

III.3.2.6 Cellule Auxiliaire

Cette armoire se compose de la partie départs auxiliaires, tel que :

- Eléments chauffants et ventilateurs pour les groupes électropompes
- Départs puissance vers les vannes motorisées
- Départs vers les coffrets auxiliaires, etc.

Armoires batteries de compensation:

Pour chaque départ groupe électropompe, une armoire batterie de condensateur est proposée, elle sera constituée de condensateur, de contacteurs spécifiques pour commander les condensateurs

III.3.3 Le Pupitre de commande

Sur le pupitre de commande, on y trouve sur l'extérieur des voyants (marche/arrêt/défaut, etc.) et des boutons (Marche/arrêt/arrêt d'urgence, etc.) ainsi que les commutateurs (Mode locale/Mode distant/Modes Manuel dégradé, etc.,) ainsi que l'écran de supervision qui affichera l'évolution du Process (Marche/arrêt/Défauts etc.).

Figure III.2 *Vue de face avant de la HMI* [1]

A l'intérieur du pupitre de commande, il y aura l'automate ainsi que son alimentation et le bornier on l'on retrouve les Entrées/Sorties et un répartiteur pour une éventuelle connexion a l'automate, ce répartiteur dispose d'un mode de protocole qui est le Modbus avec des emplacements disponible pour un futur raccord avec d'autre appareillage communicant (Utilisant le Modbus).

III.3.4 Le groupe électropompe

Notre station de pompage contient 3 groupes électropompes , chaque groupe se compose des éléments nécessaire au pompage de l'eau (pompe, vanne ,démarreur progressif ,variateur de vitesse)

la pompe

La pompe utilisée **OMEGA 250-480 A** est une pompe centrifuge se caractérise par :

Débit 1300 m^3/h, puissance 237,3KW, Rendement 86,6%

Figure III.3 *Vue de générale des groupes électropompes* [1]

III.3.5 la vanne

La vanne joue un rôle très important dans la station, en effet elle garantie un démarrage et un arrêt sécurisé de la pompe (la pompe démarre et s'arrête toujours avec une vanne fermée : cela la protège contre le refoulement de l'eau à des vitesses très importantes pouvant détruire toute la station).

III.3.6 le démarreur progressif Altistart48

Le démarreur progressif ALTISTART 48 est un gradateur à 6 thyristors assurant le démarrage et l'arrêt progressif en couple des moteurs asynchrones triphasés à cage. Il intègre les fonctions de démarrage et ralentissement en douceur, les fonctions de communication avec les automatismes. Ces fonctions répondent aux applications les plus courantes des pompes. L'Altistart 48 intègre toutes les protections nécessaires au bon fonctionnement du moteur et l'application :

- la protection thermique.
- la protection contre l'inversion des phases et manque de phase.
- La protection contre les démarrages longs.
- La protection contre les démarrages consécutifs
- Signalisation et affichage de tous les défauts.
- Réduction des pointes de courant et des chutes de tensions.

Aussi, les démarreurs électroniques ALTISTART 48 remplacent les systèmes de démarrage classique avec beaucoup d'avantages :

- Réduction des coups de bélier et des coups de clapet.
- Réduction des volumes des armoires de plus de 50%.
- Optimisation de l'énergie installée.

III.3.7 variateur de vitesse Altivar 61

Le variateur de vitesse Altivar 61 est un convertisseur de fréquence pour moteur asynchrone triphasé de 0,75 kW à 630 kW.

Il est destiné au secteur de l'industrie et du bâtiment pour les applications à couple variable. Il vise principalement les applications :

- Pompage (régulation pression, débit..)
- Distribution d'eau (relevage, sur-presseur, multi pompe.)
- Ventilation (traitement d'air, désenfumage..)

Il intègre de nombreuses fonctionnalités dédiées aux applications ainsi que différents types de commandes moteurs. Il est livré avec un terminal graphique en standard. C 'est un produit communicant offrant une large palette de protocole dont Modbus et CAN open en standard.

III.3.8 le système anti-bélier

ANTI-BELIER : Prévention ou absorption des coups de bélier, élimination des chocs hydrauliques qui se produit lorsque l'écoulement est interrompu plus rapidement que la décélération possible par les pertes de charge dues au frottement du liquide sur les parois. Ceci peut être causé soit par la fermeture trop brusque d'une vanne, soit par l'arrêt d'une pompe. Dans ce dernier cas, un vide se crée derrière la colonne de liquide en mouvement et le choc hydraulique est subi par la pompe elle-même quand l'écoulement s'inverse pour remplir le vide.

Dans notre station on utilise un système anti-bélier : un ballon dont le niveau est indiqué par 5 détecteurs (poires de niveau) contrôlés par :

- Un compresseur qui compresse l'air dans l'anti-bélier afin diminuer le niveau d'eau en cas de surpression d'eau
- Une vanne qui sert à purger l'eau afin d'augmenter le niveau d'eau dans le ballon en cas de dépression d'eau

III.3.9 Les capteurs

III.3.9.1 Les capteurs de niveau (poire de niveau)

Les capteurs de type TOR sont destinés pour indiquer les niveaux comme suite :

• 3 poires de niveau (niveau Bas : LLL, niveau Moyen : LLA, niveau Haut : HL) dans la bâche (Réservoir source du bas), pour la gestion des pompes et pour éviter la marche a sec de la pompe.

• 5 poires de niveau (niveau très Bas : LLA, niveau Bas : LL, niveau Moyen : NL, niveau Haut : HL

Niveau très Haut : HHL) dans le ballon de système antibélier pour

III.3.9.2 Les capteurs de température

Dans notre station on utilise pour chaque pompe 3 capteurs de température pour les bobines et 2 capteurs de température pour les paliers. Ces capteurs sont utilisés pour signaler des alarmes en cas de dépassement des seuils.

III.3.9.3 Les capteurs de vibration

Ces capteurs sont utilisés pour mesurer le taux de vibration des bobines et des paliers pour signaler des alarmes en cas de dépassement des seuils limites.

III.3.10 Débitmètre

Il mesure le débit et le transmet vers l'automate**.** Ainsi on peut régler la vitesse nécessaire pour éviter la marche à sec et cela dans le cas ou le niveau du réservoir soit très bas.

III.4 Algorithme de procès

L'Algorithme du process est géré par l'automate, cependant l'opérateur peut intervenir durant le Process pour changer sa configuration. Pour modifier l'état du Process l'opérateur dispose soit des boutons et commutateurs placés sur le pupitre soit sur le superviseur (Magelis), il y aura une interface pour éventuellement changer et observer l'état du Process.

L'Automate Modicon M340 aura pour rôle essentiel de gèrer les actionneurs tel que les groupes électropompe, les vannes motorisées, Il assurera la sécurité du process afin d'évité toutes fausse manipulation de la part de l'opérateur, mais aussi il permettra la communication entre le variateur de vitesse (ATS 61), les démarreurs progressives (ATV 48) et le superviseur (Magelis).

L'automate va tester les démarreurs progressif et le variateur en les initialisent pour réinitialisé leur états et pour vérifier si il y aurait une possibilité d'une anomalie.

Par mesure de sécurité pour les moteurs, le démarrage des pompes se fait à vide (c'est-à-dire sans charge, ici la charge est l'eau à travers la canalisation) pour éviter toutes éventuelles de surchauffe des pompes. Le démarrage d'une pompe se passe pendant un court instant c'est le régime transitoire qui n'est rien d'autre que : le temps que met la dynamique du système a passe d'une valeur initiale (ici pompe à l'arrêt) à une valeur finale (la pompe est arrivé a sa vitesse nominale).

Une fois que la pompe est atteint sur régime permanent (c'est-à-dire que l'appel de courant à disparut et le courant c'est stabiliser à sa valeur nominale), l'automate actionnera alors l'ouverture des vannes.

L'indicateur de niveau (LS 01) et le contrôleur de niveau à Ultrason (LC 01) donneront l'image du contenu du Réservoir R100.

- Si le capteur indique un niveau très bas(LLL) alors les pompes seront toutes actionnées
- Si le capteur indique un niveau moyen(LLA) alors on activera soit une ou deux pompe(s)
- Si le capteur indique un niveau haut(HL) alors on activera une seule pompe.

Les indicateurs de niveaux seront affichés sur le superviseur sous la forme d'une jauge en temps réel ainsi que l'état de l'activation des pompes.

Par mesures de sécurité l'automate gérera le fonctionnement des pompes, si une pompe décèle un défaut de vibration (VBS) ou de température du bobinage (TW) ou la température du pallier moteur (TB) ou autre défaillance technique alors l'automate arrêtera la pompe en question et le superviseur affichera le nature de la défaillance de la (ou des) pompe(s) en question.

Le superviseur

Le superviseur est un écran tactile de contrôle de l'ensemble de l'application du process, l'application est programmable avec des logiciels de type SCADA (Vijeo Designer). On peut à partir de cet écran de superviseur observé les fonctions en temps réel suivantes qui seront disponibles :

- Etat du Process
- Etat de la distribution électrique
- Valeurs des mesures électriques provenant des tableaux de distribution MT
- Valeurs des mesures analogiques de niveau débit pression etc.…
- Choix et visualisation du mode de fonctionnement
- Signal d'anomalie sonore et visuelle
- Consignation et impression des états et des défauts
- Edition de courbes historiques
- Edition de rapports journaliers, mensuels et annuels

Gestion des taches du système

La tâche de POMPAGE se décompose en plusieurs autres tâches :

- Tâche de chargement par le Réservoir R100 avec le contrôle de ses niveaux
- Tâche gestion des Groupes de pompage (SP107-A, SP107-B et SP107-C).
- Tâche gestion des Sécurités (Compresseur C101 et Anti-Bélier V101, etc.).
- Tâches gestion des Défauts et Supervision Process.

Les différentes tâches sont gérées par l'automate, on y trouve des signaux

- Des entrées en TOR (Tout ou Rien)
- Des entrées Ana (Analogique)
- Des sorties TOR
- Des sorties Ana

L'automate, via la carte Modbus communique avec les Démarreurs et le Variateur (ATS et ATV).

Gestion des Anomalies

Défaut, Alarmes, définition:

On appelle défaut, un événement dont l'origine est une anomalie physique. On définit l'alarme comme une représentation visuelle du défaut. L'alarme est l'information issue du défaut. Elle est transmise à l'opérateur par l'intermédiaire du terminal IHM.

On distingue plusieurs types de défauts :

- **Les défauts câblés** (ou défauts électriques). Ces défauts sont acquis directement par câblage depuis les armoires électriques. Ils correspondent à des dysfonctionnements liés à la sécurité des hommes : arrêts d'urgences etc. et la protection des machines : les défauts des départs actionneurs (thermiques). Ces défauts sont surveillés par le système en permanence. Une alarme de ce défaut est générée.
- **Les défauts générés par le programme**, ils peuvent être de plusieurs natures :

- Ceux liés à une anomalie physique d'un ou plusieurs constituants de l'équipement : exemple discordance entre la commande et le retour d'état du contacteur, rupture de fils d'une mesure…
- Les défauts système. Ils correspondent à des anomalies physiques liées aux organes du système (par exemple : défaut de carte).

Procédure d'acquittement des défauts

Lorsqu'un défaut apparaît, une alarme est obligatoirement associée à celui-ci. L'opération d'acquittement est réalisée par l'opérateur sur le terminal IHM en deux étapes :

- La prise en compte effective de l'alarme sur la page d'alarme associée au défaut en appuyant sur une touche spécifique.

- L’acquittement proprement dit du défaut en sélectionnant la page d’exploitation de l’équipement associé au défaut et en validant l’acquittement (cet acquittement n’est effectif que si l’événement qui est à l’origine du défaut a disparu).

Défaut électrique :

Ce défaut câblé regroupe les dysfonctionnements liés à la chaîne de commande du moteur : thermique, limiteur de couple, etc. Il est généré quel que soit le mode de marche de l’actionneur.

En principe ce défaut est traité par les armoires électriques. Au niveau programme l'équipement concerné passe en position de sécurité (arrêt dans la plupart des cas). Une alarme est immédiatement générée.

Défaut discordance :

Ce défaut à pour objet de détecter les incohérences qui peuvent se produire entre l’ordre émis par l’automate et le retour d’état réel d’un actionneur.

Lorsque l’automate donne un ordre à un actionneur (exemple : marche ou arrêt), une temporisation est lancée, si à la fin de la temporisation le système n’a pas reçus le retour de l’information attendu (retour marche), alors celui-ci est déclaré en défaut discordance.

Ce défaut provoque immédiatement le passage en position de sécurité du moteur. Une alarme est immédiatement générée, lorsque l’état de l’automate et celui du site sont de nouveau en concordance, l’actionneur reste en position de sécurité tant que l’opérateur n’a pas effectué une procédure d'acquittement.

III.5 Conclusion

Ce chapitre décrit l’ensemble des fonctions qui sont représentées dans la station de KOUBA 107. Une telle description est nécessaire pour comprendre le fonctionnement de la station et par la suite pouvoir créer les sections du programme de gestion de la station.

Chapitre IV
Création des DFB

IV.1 Introduction

Dans notre application, on créera 5 DFB principaux :

- Le DFB_anti_bélier.
- Le DFB gestion de la vanne.
- Le DFB autorisation du démarrage de la pompe.
- Le DFB défaut : qui résume les différents défauts de la station.
- Le DFB GEP pour la gestion du groupe électropompe.

Tous les programmes sont écrits en langage structuré suite à son avantage de souplesse et de facilité de compréhension par les autres programmeurs (en particulier les utilisateurs de ces DFB) pour cela des commentaires vont être mis en place.

IV.2 Blocs fonction utilisateur (DFB)

IV.2.1 Présentation

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

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 (par exemple, l'algorithme de commande d'un moteur incluant la prise en compte des sécurités locales).

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.[8]

IV.2.2 Avantage d'utiliser un DFB

L'utilisation d'un bloc fonction DFB dans une application vous permet :

- de simplifier la conception et la saisie du programme,
- d'accroître la lisibilité du programme,

- de faciliter la mise au point de l'application (toutes les variables manipulées par le bloc fonction sont identifiées sur son interface),
- de diminuer le volume de code généré (le code correspondant au DFB est chargé une seule fois, quel que soit le nombre d'appels au DFB dans le programme, seules les données correspondantes aux instances sont générées).

IV.2.3 Comparaison avec un sous-programme

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

- de paramétrer plus facilement le traitement,
- d'utiliser des variables internes propres au DFB, donc indépendantes de l'application,
- de tester son fonctionnement indépendamment de l'application.

De plus, les langages LD et FBD permettent de visualiser de manière graphique les DFB, ce qui facilite la conception et la mise au point de votre programme.

IV.2.4 Description du type DFB

Le graphe ci-dessous illustre une représentation graphique d'un modèle DFB.

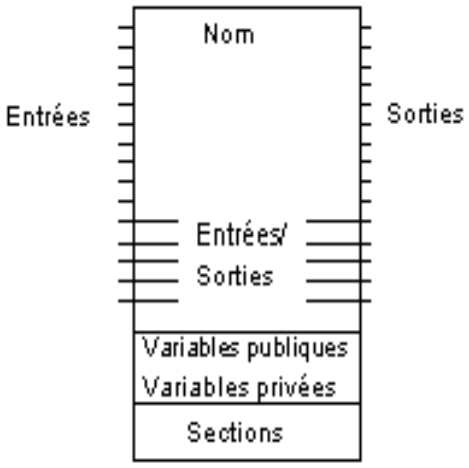


Figure IV.1: *une représentation graphique d'un modèle DFB* [1]

Le bloc fonction comprend les éléments suivants :

- Nom : nom du type DFB (32 caractères max.) Ce nom doit être unique dans les bibliothèques, les caractères utilisés autorisés dépendent du choix fait dans la zone Identificateurs de l'onglet Extensions de langage des options du projet :
- les paramètres :

- paramètres d'entrée (hors paramètres d'E/S)
- paramètres de sortie (hors paramètres d'E/S)
- paramètres d'E/S.

➢ les variables

- Variables publiques : variables internes accessibles par le programme d'application.
- Variables privées : variables internes imbriquées ou DFB, inaccessibles par le programme d'application.

➢ Sections : sections de code DFB dans LD, IL, ST ou FBD.

➢ Commentaire (1024 caractères maximum). Les caractères de formatage (retour chariot, onglet, etc.) ne sont pas autorisés.

Pour chaque type de DFB, un fichier descriptif est également accessible par une boîte de dialogue : taille du DFB, nombre de paramètres et variables, numéro de version, date de la dernière modification, niveau de protection, etc.[8]

IV.2.4.1 Paramètres DFB

Eléments à définir pour chaque paramètre :

Lors de la création du bloc fonction, les éléments suivants doivent être définis pour chaque paramètre :

- Nom : nom du type DFB (32 caractères maximum). Ce nom doit être unique dans les bibliothèques, les caractères utilisés autorisés dépendent du choix fait dans la zone Identificateurs de l'onglet Extensions de langage des options du projet :

- un type d'objet (BOOL, INT, REAL, etc.).
- Commentaire de 1 024 caractères maximum (facultatif). Les caractères de mise en forme (retour chariot, tabulation, etc...) sont interdits),
- une valeur initiale.
- Attribut Lecture/Ecriture : détermine si la variable peut être écrite lors de l'exécution (R : lecture seule - R/W : lecture/écriture). Cet attribut ne doit être défini que pour les variables publiques.
- Attribut de sauvegarde : détermine si la variable peut être enregistrée.

Types d'objets

Les types d'objets disponibles pour les paramètres DFB appartiennent aux familles suivantes :

- Famille de données élémentaires : EDT. Cette famille comprend les types d'objets suivants : Booléen (BOOL, EBOOL), Entier (INT, DINT, etc.), Réel (REAL), Chaîne de caractères (STRING), Chaîne de bits (BYTE, WORD, etc.), etc.
- Famille de données dérivées : DDT. Cette famille comprend les types d'objets tableau (ARRAY) et structure (utilisateur ou IODDT).
- Famille de données génériques : ANY_ARRAY_xxx.
- Famille de bloc fonction : FB. Cette famille comprend les types d'objets EFB et DFB.

Objets autorisés pour les différents paramètres

Les familles d'objets suivantes peuvent être utilisées pour chacun des paramètres DFB :

Famille d'objets	**Structures**	**DDT (1)**	**IODDT**	**GDT : ANY_ARRAY_x**	**FB**
Entrées	Oui	Oui	Non	Oui	Non
Entrées/sorties	Oui(2)	Oui	Oui	Oui	Non
Sorties	Oui	Oui	Non	Oui	Non
variables publiques	Oui	Oui	Non	Non	Non
variables privées	Oui	Oui	Non	Non	Oui
Légende :					
(1)	Famille de données dérivées, à l'exception des types de données dérivées d'E/S (IODDT).				
(2)	Sauf les variables statiques de type EBOOL, sur les automates Quantum.				

Tableau IV.1 : *Les familles d'objets pouvant être utilisées pour les paramètres DFB* [8]

IV.1.4.2 Variables DFB

Description des variables

Variable	**Nombre maximum**	**Rôle**
Publique	Illimité	Ces variables internes du DFB peuvent être utilisées par le DFB, par le programme application et par l'utilisateur en mode réglage.
Privée	Illimité	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 mais ces types de variables sont accessibles via le tableau d'animation. Ces variables sont généralement des variables nécessaires à la programmation du bloc, mais sans intérêt pour l'utilisateur (par exemple, le résultat d'un calcul intermédiaire,...).

Variables accessibles par le programme application

Les seules variables accessibles par le programme application sont les variables publiques. Pour cela, vous devez utiliser dans le programme la syntaxe suivante: Nom_DFB.Nom_variable

Nom_DFB représente le nom de l'instance du DFB utilisé (32 caractères au maximum),

Nom_variable représente le nom de la variable publique (8 caractères au maximum).

Exemple : Controle.Gain indique la variable publique Gain de l'instance de DFB nommée Contrôle

IV.2.5 Instances de blocs fonction utilisateur (DFB)

Instance de DFB

Une instance de DFB est une copie du modèle de DFB (type de DFB):

- elle exploite le code du type de DFB (le code n'est pas dupliqué).
- elle crée une zone de données spécifique à cette instance, qui est la recopie des paramètres et des variables du type de DFB. Cette zone est située dans l'espace données de l'application.

Vous devez repérer chaque instance de DFB que vous créez, par un nom de 32 caractères au maximum, les caractères utilisés autorisés dépendent du choix fait dans la zone Identificateurs de l'onglet Extensions de langage des options du projet.

Le premier caractère doit être une lettre. Les mots clefs et les symboles sont interdits.

Création d'une instance

A partir d'un type de DFB, vous pouvez créer autant d'instances que nécessaire; la seule limitation étant la taille mémoire de l'automate.

Règles d'utilisation des DFB dans un programme

Les instances DFB peuvent être utilisées dans tous les langages [liste d'instructions (IL), littéral structuré (ST), langage à contacts (LD) et langage en blocs fonctionnels (FBD)], ainsi que dans toutes les tâches du programme d'application (sections, sous-programmes, etc.), à l'exception des tâches d'événements et des transitions du programme SFC.

Affectation des paramètres

Le tableau suivant résume les différentes possibilités d'affectation des paramètres possibles dans les différents langages de programmation.

Paramètre	**Type**	**Affectation du paramètre (1)**	**Affectation**
Entrées	EDT (2)	Défini, valeur, objet ou expression	Facultatif (3)
	BOOL	Défini, valeur, objet ou expression	Optionnelle
	DDT	Défini, valeur ou objet	Obligatoire
	ANY_...	Défini ou objet	Obligatoire
	ANY_ARRAY	Défini ou objet	Obligatoire
Entrées/sorties	EDT	Défini ou objet	Obligatoire
	DDT	Défini ou objet	Obligatoire
	IODDT	Défini ou objet	Obligatoire
	ANY_...	Défini ou objet	Obligatoire
	ANY_ARRAY	Défini ou objet	Obligatoire
Sorties	EDT	Défini ou objet	Optionnelle
	DDT	Défini ou objet	Optionnelle
	ANY_...	Défini ou objet	Obligatoire
	ANY_ARRAY	Défini ou objet	Optionnelle

(1) Défini en langage à contacts (LD) ou langage en blocs fonctionnels (FBD). Valeur ou objet en langage littéral structuré (ST) ou liste d'instructions (IL).

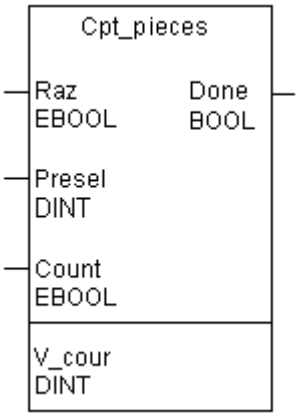
(2) Sauf paramètres de type BOOL

(3) Sauf paramètres de type STRING qui sont obligatoires

Tableau IV.2: *Bloc DFB (les différentes possibilités d'affectation des paramètres)* [8]

Exemple de programmation d'un bloc DFB

Cet exemple de programmation d'un compteur, à partir d'un DFB, est donné à titre didactique



Caractéristiques du type de DFB

Le type de DFB permettant de réaliser le compteur est le suivant

Les éléments du Type DFB Cpt_pieces sont les suivants.

- Nom du type de DFB:Cpt_pièces
- Paramètres d'entrées
- Raz: remise à zéro du compteur (type EBOOL)

- Presel : valeur de présélection du compteur (type DINT)
- Count: entrée de comptage (type EBOOL)

➢ Paramètres de sorties

- Done: sortie de valeur de présélection atteinte (type BOOL)

➢ Variable interne publique

- V_cour: valeur courante du compteur (type DINT)

Fonctionnement du compteur

Le fonctionnement du compteur doit être le suivant.

Phase	Description
1	Le DFB compte les fronts montants sur l'entrée Count.
2	Le nombre de fronts comptés est mémorisé par la variable V_cour. Cette variable est remise à zéro par un front montant sur l'entrée Raz.
3	Lorsque le nombre de fronts comptés est égal à la valeur de présélection, la sortie Done est positionnée à 1. Cette variable est remise à zéro par un front montant sur l'entrée Raz.

Programme interne du DFB

Le programme interne du type de DFB Cpt_pieces est défini en langage ST de la manière suivante.

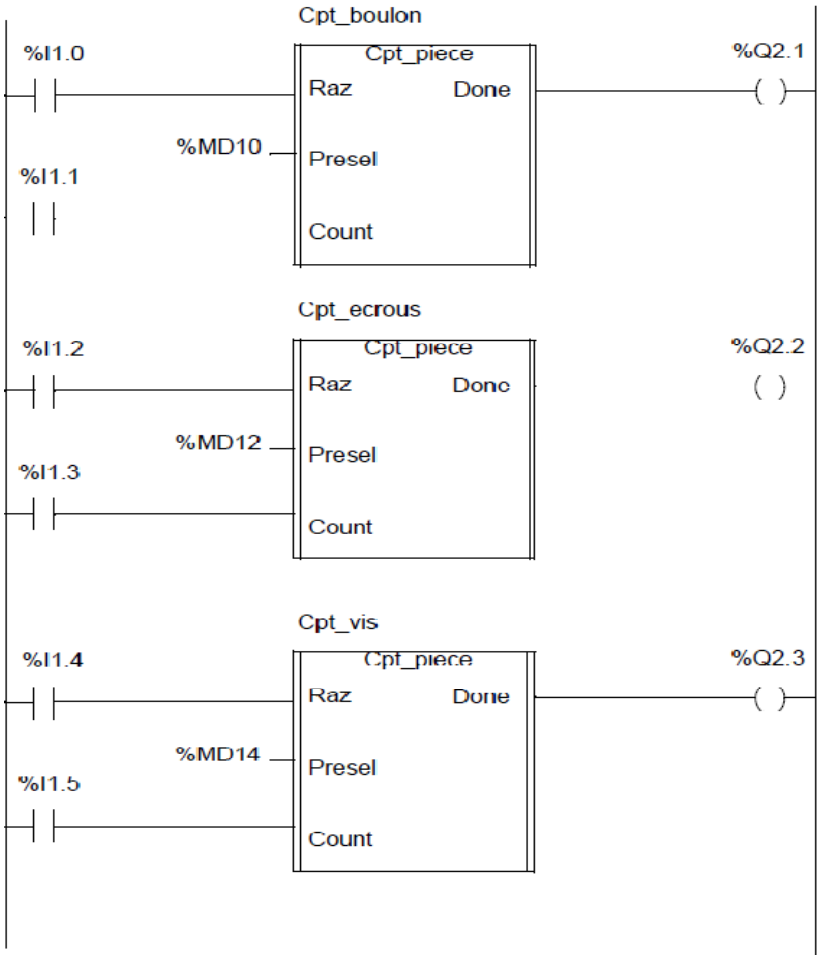
```
(*Programmation du DFB Cpt_pieces*)
IF RE (Raz) THEN
V_cour:=0;
END_IF;
IF RE (Count) THEN
V_cour:=V_cour+1;
END_IF;
IF(V_cour>=Presel) THEN
SET (Done);
ELSE
RESET (Done);
END_IF;
```

Exemple d'utilisation

Supposons que votre application nécessite de compter 3 types de pièces (par exemple, des boulons, des écrous et des vis). Vous pouvez utiliser 3 fois le type de DFB Cpt_pieces (3 instances) pour réaliser ces différents comptages.

Le nombre de pièces à approvisionner pour chaque type, est défini respectivement dans les mots %MD10, %MD12 et %MD14 Lorsque le nombre de pièces est atteint, le compteur commande une sortie (%Q1.2.1, %Q1.2.2 ou %Q1.2.3) qui pilote l'arrêt du système d'approvisionnement de pièces correspondant.

Le programme application est saisi en langage à contacts de la manière suivante. Il utilise les 3 DFB (instances) Cpt_boulons, Cpt_écrous et Cpt_vis afin de compter les différentes pièces.[8]



IV.3 DFB du système anti bélier

IV.3.1 cahier de charge du DFB anti bélier

Le ballon anti bélier comprend pour la partie automatisation : 5 capteurs de niveau, une vanne et un compresseur ,du même le bloc DFB contient les signaux des 5 capteurs en entrée ,et doit envoyer de plus des deux signaux d'alarme de très haut et très bas niveau , un signal d'activation de la vanne et du compresseur, et dans le cas normal un autre signal d'autorisation est activé .

Les 5entrées :

- HHL : niveau très haut de l'anti bélier.
- HL : niveau haut.
- NL : niveau normal ou moyen.
- LL : niveau bas.
- LLA : niveau très bas.

Les 5 sorties :

- sec_niveau_haut : sécurité niveau très haut (lancement d'une alarme lorsque le niveau est très haut)
- sec_niveau_bas : sécurité niveau très bas (lancement d'une alarme lorsque le niveau est très bas)
- Auto : autorisation du démarrage du groupe électropompe.
- Comp : démarrage du compresseur.
- cmd_ouv_vanne : commande ouverture de la vanne de l'anti bélier.

Le DFB anti bélier doit assurer :

- Activation des sorties sec_niveau_haut et sec_niveau_bas dans le cas ou les signaux HHL et LLA sont actives.
- Garder le niveau de l'eau dans l'anti bélier entre HL et LL, ainsi :

- Si le niveau est HL on démarre le compresseur, ce dernier exerce une pression sur l'air dans l'anti bélier et par la suite pouvoir baisser le niveau d'eau jusqu'à atteindre le niveau NL.
- Si le niveau est LL on démarre la vanne, ceci permettra l'évacuation de l'air dans l'anti bélier et par la suite pouvoir augmenter le niveau d'eau jusqu'à atteindre le niveau NL.

- Le signal autorisation est actif lorsque le niveau d'eau soit compris entre HL et LL.

IV.3.2 le programme du DFB anti bélier

Les signaux :

anti_bellier		<DFB>		
<entrées>				
HHL	1	BOOL		
HL	2	BOOL		
NL	3	BOOL		
LL	4	BOOL		
LLA	5	BOOL		
<sorties>				
sec_niveau_haut	1	BOOL		
sec_niveau_bas	2	BOOL		
Auto	3	BOOL		
comp	4	BOOL		
cmd_ouv_vanne	5	BOOL		

Le programme :

La section du programme est écrite en langage structuré (ST)

```
(*Ce programme est le DFB qui gère le système anti-bélier de la station pompage
Ce DFB comprend  5entrées et 5 sorties :
Les 5entrées :
HHL : niveau très haut .
HL : niveau haut .
NL : niveau normal ou moyen .
LL : niveau bas.
LLA : niveau très bas .
Les 5 sorties :
sec_niveau_haut : sécurité niveau très haut (lancement d'une alarme lorsque le niveau est très haut)
sec_niveau_bas : sécurité niveau très bas (lancement d'une alarme lorsque le niveau est très bas)
Auto : autorisation du démarrage du groupe électropompe .
Comp : démarrage du compresseur .
cmd_ouv_vanne : commande ouverture de la vanne de l'anti bélier .*)
sec_niveau_haut:=HHL;
sec_niveau_bas:=LLA;
(*On autorise le démarrage du groupe électropompe si les niveau d'eau dans l'anti bélier est bon :
ni HHL ni LLA *)
Auto:=NOT HHL AND NOT LLA;
(*démarrage du compresseur*)
IF  (HL OR HHL) THEN
comp:=1;
END_IF;
(*ouverture de la vanne *)
IF  (LL OR LLA) THEN
cmd_ouv_vanne:=1;
END_IF;
(*Si le niveau est normal : alors on arrête le compresseur et on ferme la vanne*)
if NL then
cmd_ouv_vanne:=0;
comp:=0;
end_if;
```

IV.3.3 la simulation du DFB anti bélier

Il est nécessaire d'assurer que le programme DFB fonctionne correctement avant de pouvoir être utilisé pour écrire le programme final qui va gérer la station de pompage qui n'est rien d'autre qu'une suite de DFB reliés entre eux à travers un programme FBD.

Pour cela on met en place une section écrite un programme en FBD dont on va lui affecter les entrées / sorties du programme de simulation et une autre section en langage LADDER permettant de simuler le fonctionnement de l'anti bélier.

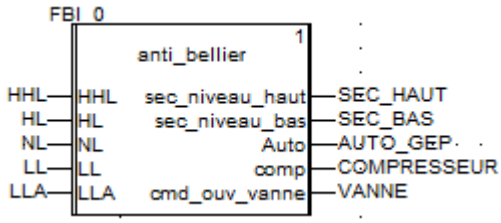


Figure IV.2: *la section de simulation FBD de l'anti bélier*

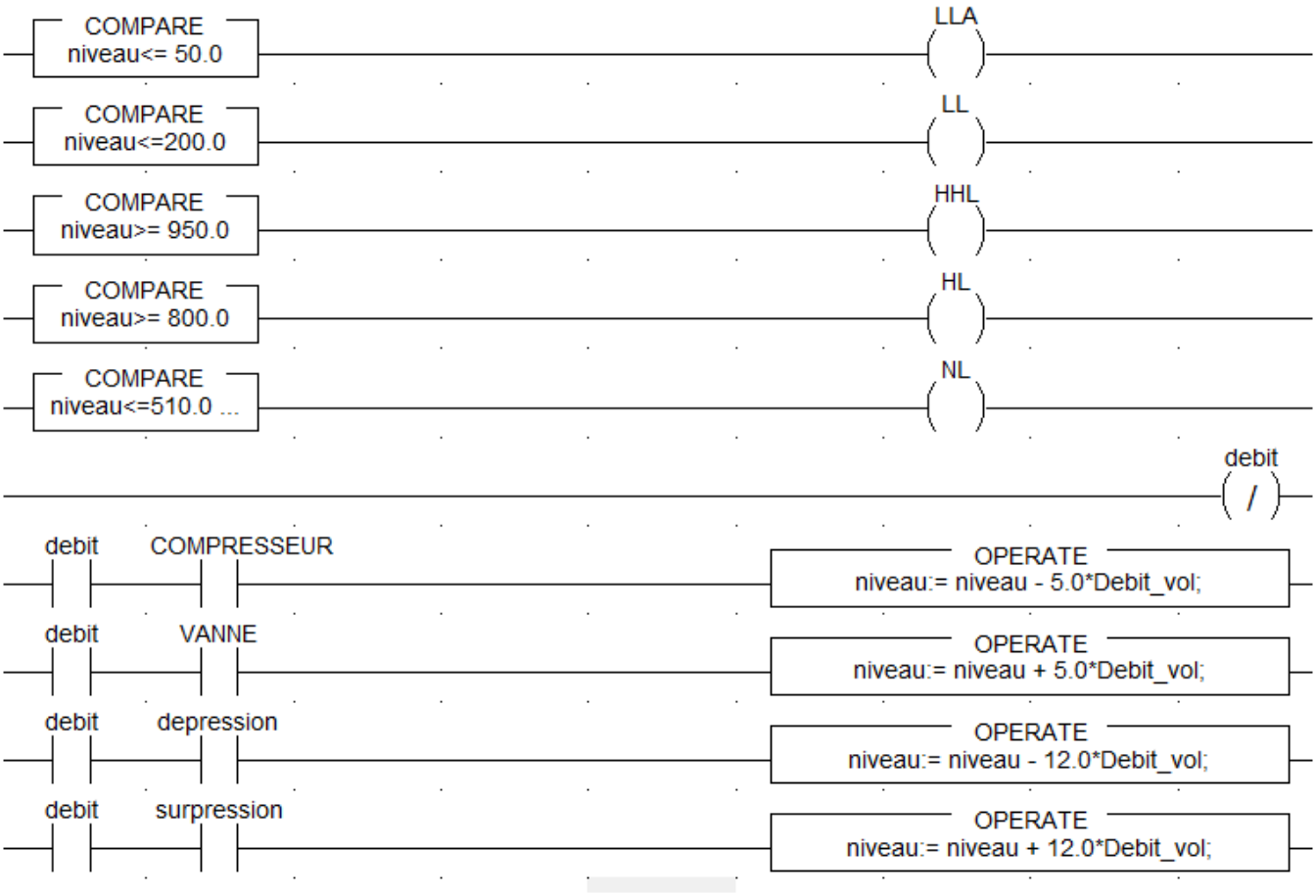


Figure IV.3: *la section de simulation LADDER de l'anti bélier*

Les variables niveau et débit sont des variables intermédiaires assurant la simulation (voir l'exemple détaillé mise en ouvre d'une application).

Pour le bien visualiser le système on utilise un écran d'exploitation.

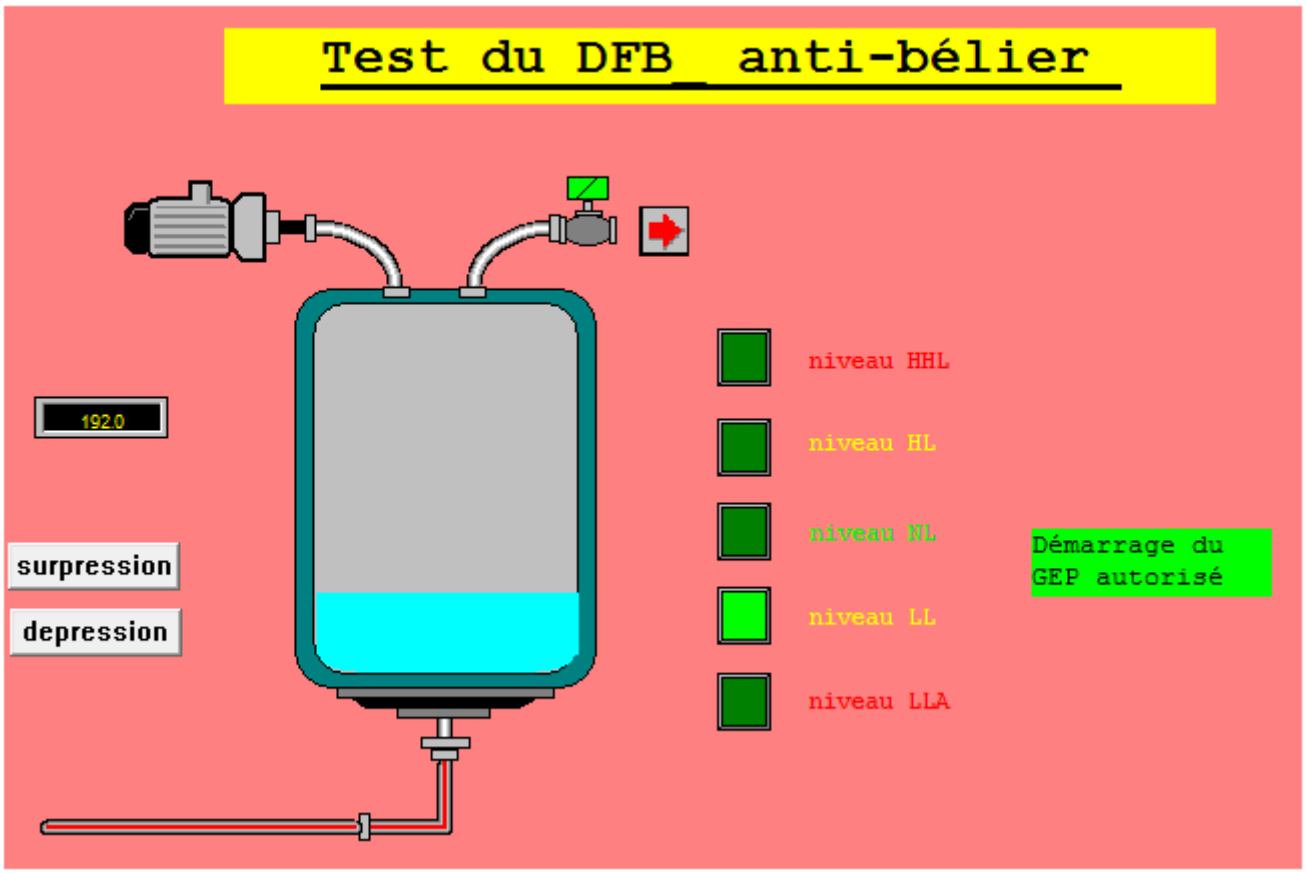


Figure IV.4: *l'écran d'exploitation de l'anti bélier.*

IV.4 Le DFB gestion de la vanne

IV.4.1 cahier de charge du DFB gestion de la vanne

La vanne motorisé reçoit les signaux d'ouverture et de fermeture et peut envoyer les deux signaux indiquant son état (complètement ouverte ou complètement fermée). Le DFB reçoit 6 signaux et doit générer 5 signaux :

Les 6 entrées :

- ouvrir : Commande ouvrir la vanne (signal envoyé par l'utilisateur ou un autre programme)
- fermer : Commande fermer la vanne (signal envoyé par l'utilisateur ou un autre programme)
- fco : Fin de course vanne ouverte (signal générer par la vanne)
- fcf : Fin de course vanne fermée (signal générer par la vanne)
- defaut : Défaut de la vanne
- aquit : Acquittement d'un défaut

Les 5 sorties :

- cmd_ou : Commande d'ouverture de la vanne (signal envoyé vers la vanne)
- cmd_fe : Commande de fermeture de la vanne (signal envoyé vers la vanne)
- dis_o : Discordance ouverture

- dis_f : Discordance fermeture
- incoh : Incohérence

Le DFB vanne doit assurer :

- La génération des signaux d'ouverture et de fermeture de la vanne en cas d'absence de défaut.
- La génération des signaux discordance d'ouverture et de fermeture :

- La discordance d'ouverture est généré lorsque on envoie la commande ouvrir la vanne et qu'on ne reçoit pas le signal « fco » indiquant que la vanne est complètement ouverte et cela après un certain temps (la vanne prend un temps pour s'ouvrir due à son inertie). pour des raisons de sécurité ce temps serai une variable locale et non pas d'entrée pour le bloc.
- De même pour la discordance fermeture.

- La génération du signal incohérence lorsque on reçoit les deux signaux fcf et fco en même temps pendant une duré réglée par le programmeur (variable locale) et cela de peur que la présence de ces deux signaux n'est qu'un parasite.
- L'acquittement des défauts : discordance d'ouverture et de fermeture et incohérence. l'acquittement de ces défauts n'est appliqué qu'à l'absence du défaut et après un appuie sur le bouton aquit :

- Pour la discordance d'ouverture, l'absence de défaut signifie que la vanne est pue être ouverte.
- Pour la discordance de fermeture, l'absence de défaut signifie que la vanne est pue être fermée.
- Pour l'incohérence, l'absence de défaut signifie que les deux signaux fcf et fco ne sont plus actifs en même temps vanne est pu être ouverte.

IV.4.2 le programme du DFB gestion de la vanne

Les signaux :

DFB_VANNE		<DFB>	
<entrées>			
ouvrir	1	BOOL	
fermer	2	BOOL	
fco	3	BOOL	
fcf	4	BOOL	
defaut	5	BOOL	
aquit	6	BOOL	
<sorties>			
cmd_ou	1	BOOL	
cmd_fe	2	BOOL	
dis_o	3	BOOL	
dis_f	4	BOOL	
incoh	5	BOOL	
<entrées/sorties>			
<public>			
temps_dis		TIME	5s
temps_inc		TIME	3s
temps		TIME	1s
<privé>			
predisc_o		BOOL	
preinc		BOOL	
predisc_f		BOOL	
TEMPORISATION_O		TON	
TEMPORISATION_F		TON	
TEMPORISATION_I		TON	

Le programme :

La section du programme est écrite en langage structuré (ST)

```
(*Ce programme est le DFB qui gère la vanne Motorisée
Ce DFB comprend 6 entrées et 5 sorties :
Les 6 entrées :
ouvrir : Commande ouvrir la vanne
fermer : Commande  fermer la vanne
fco : Fin de course vanne ouverte
fcf : Fin de course vanne fermée
defaut : Défaut de la vanne
aquit : Acquittement d'un défaut
Les 5 sorties :
cmd_ou : Commande d'ouverture de la vanne
cmd_fe : Commande de fermeture de la vanne
dis_o : Discordance ouverture
dis_f : Discordance fermeture
incoh : Incohérence  .*)
(*******************************************************)
                    (*cas d'ouverture*)
(*******************************************************)
(* si on envoi la Commande ouverir la vanne cela
Provoque l'ouverture de la vanne s'il n'y a pas de défaut  *)
IF NOT defaut and ouvrir  THEN
(* envoi un  sigale de  commande ouvrir a la vanne *)
set (cmd_ou);
reset(cmd_fe);
END_IF;
(* lancer une temporisation ''temps maximum nécessaire pour l'ouverture de la vanne'' *)
(*predisc pour dire seulement que la temporisation est terminée *)
TEMPORISATION_0 (IN:=cmd_ou, PT:=temps_dis,
    Q=> predisc_o);
(*si la temporisation est terminée et qu'on n'a pas reçu le signal fco
''fin d'ouverture de la vanne''   *)
if predisc_o and not fco
```

```
(* alors on envoi un  signal d'erreur discordance  *)
then  set(dis_o);
END_IF;
(*********************************************************)
         (*cas de fermeture c'est les Mêmes tapes*)
(*********************************************************)
IF NOT defaut and fermer  THEN
set (cmd_fe);
reset(cmd_ou);
END_IF;
TEMPORISATION_F (IN:=cmd_fe, PT:=temps_dis,
    Q=> predisc_f ) ;
if predisc_f and not fcf    then
   set(dis_f);
END_IF;
(*     Les commandes d'ouverture et de fermeture de la vanne     *)
(*s'arrêtent après recevoir des signaux fco et fcf respectivement*)
if fco then reset(cmd_ou);
end_if;
if fcf then reset(cmd_fe);
end_if;
(*********************************************************)
         (*Test de l'incohérence*)
(*********************************************************)
(*dès que les deux signaux fcf et fco arrivent en même temps
en commence à calculer le temps écoulé (preinc) avec temps=1s *)
TEMPORISATION_I (IN:=(fcf and fco), PT:=temps_inc,
    Q=> preinc );
(*si le temps ecoulé = temps_inc alors on signale l'incohérence*)
IF preinc THEN
  set(incoh);
END_IF;

(*****************************************************)
        (*Pour l'acquittement des défauts *)
(*****************************************************)
            (* pour la discordance en ouverture *)
(*ce défaut disparu en cas d'appui sur le bouton acquit après ouverture de la vanne*)
IF aquit and fco then
   reset(predisc_o);
   reset(dis_o);
END_IF;
(* pour la discordance en ouverture *)
(*ce défaut disparu en cas d'appui sur le bouton acquit après ouverture de la vanne*)
IF aquit and fcf then
   reset(predisc_f);
   reset(dis_f);
END_IF;
                       (*pour l'incohérence*)
(*       ce défaut disparu en cas d'appui sur le bouton acquit         *)
(*et que les deux signaux fcf et fco ne se présentent pas en même temps *)
IF aquit and not(fco and fcf) then
   reset(incoh);
END_IF;
```

IV.4.3 la simulation du DFB gestion de la vanne

Pour la simulation on écrit un programme en FBD dont on va lui affecter les entrées / sorties du programme de simulation.

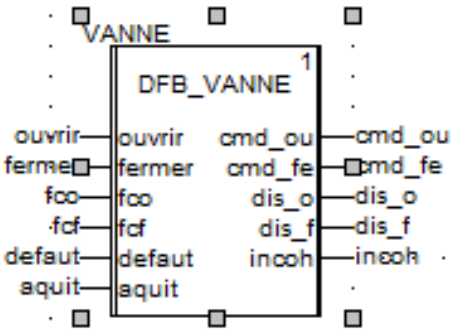


Figure IV.5: *la section de simulation FBD vanne*

Pour le bien visualiser le système on utilise un écran d'exploitation.

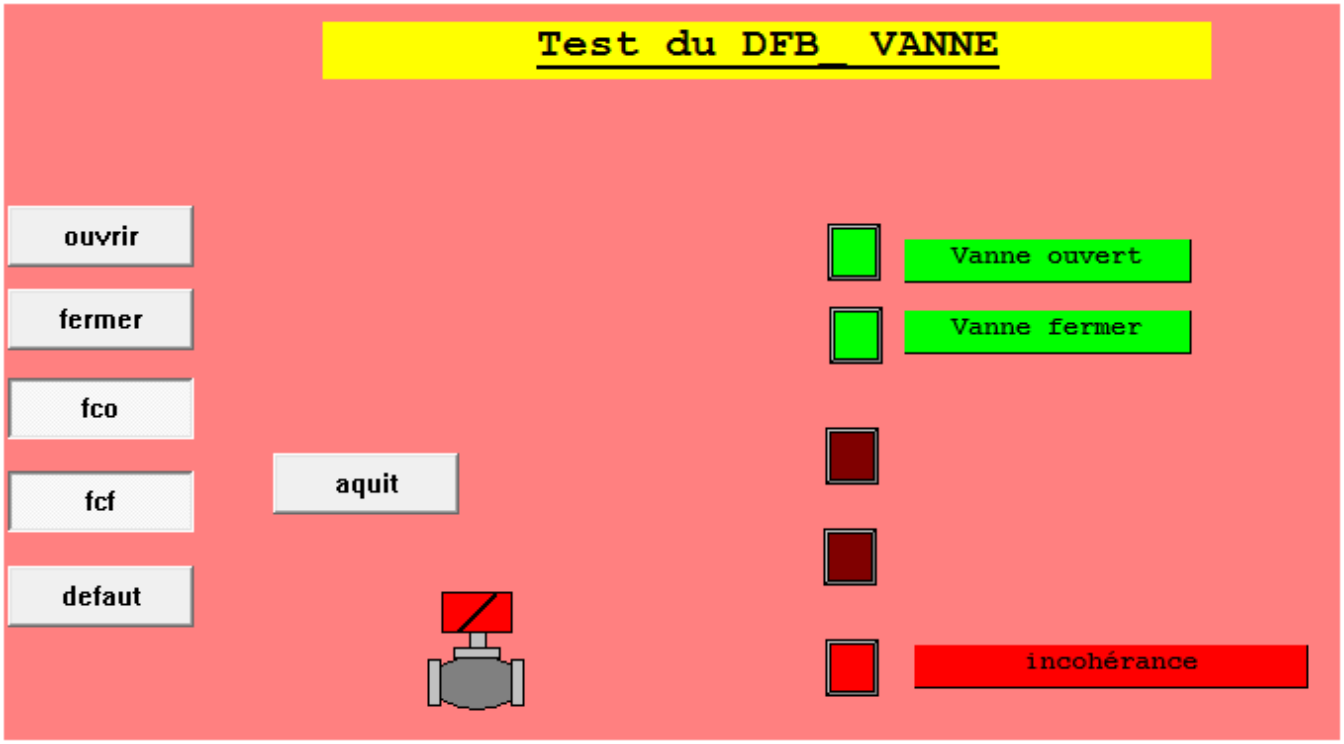


Figure IV.6: *l'écran d'exploitation de la vanne.*

IV.5 Le DFB autorisation du démarrage du GEP

Avant de démarrer le GEP il est nécessaire de vérifier plusieurs facteurs qui entrent en jeu :

- ✓ le facteur électrique (Transformateurs et groupe électrogène) permettant de savoir le nombre de GEP autorisés à se mettre en marche,
- ✓ nombre de GEP en marche,

- ✓ nombre de GEP en phase de démarrage,
- ✓ nombre de GEP en phase d'arrêt

IV.5 .1 cahier de charge du DFB autorisation

Le rôle principal de ce DFB est l'envoi d'un signal d'autorisation au GEP après la vérification de toutes les conditions nécessaires pour le démarrage.

Dans notre DFB nous avons traitées le cas de 6 GEP, 2 transformateurs ,1groupe électrogène.

Les entrées :

- tansfo_T1: le 1er Transformateur en fonctionnement.
- tansfo_T2: le 2eme Transformateur en fonctionnement.
- Groupe_EG : groupe électrogène en fonctionnement.
- GEP1_en_DEM : le GEP1 est en phase de démarrage.
- GEP2_en_DEM : le GEP2 est en phase de démarrage.
- GEP3_en_DEM : le GEP3 est en phase de démarrage.
- GEP4_en_DEM : le GEP4 est en phase de démarrage.
- GEP5_en_DEM : le GEP5 est en phase de démarrage.
- GEP6_en_DEM : le GEP6 est en phase de démarrage.
- GEP1_en_ARRET: le GEP1 est en phase d'arrêt.
- GEP2_en_ARRET: le GEP2 est en phase d'arrêt.
- GEP3_en_ARRET: le GEP3 est en phase d'arrêt.
- GEP4_en_ARRET: le GEP4 est en phase d'arrêt.
- GEP5_en_ARRET: le GEP5 est en phase d'arrêt.
- GEP6_en_ARRET: le GEP6 est en phase d'arrêt.
- GEP1_en_Marche : le GEP1 est en marche.
- GEP2_en_Marche : le GEP2 est en marche.
- GEP3_en_Marche : le GEP3 est en marche.
- GEP4_en_Marche : le GEP4 est en marche.
- GEP5_en_Marche : le GEP5 est en marche.
- GEP6_en_Marche : le GEP6 est en marche.

La sortie :

- Autoris: signale qui donne l'autorisation de démarrage au GEP.

Le DFB d'autorisation doit

- Définir le nombre de GEP autorisée à se mettre en marche selon le nombre des équipements électriques en fonctionnement :

- Si les 2 transformateurs sont en fonctionnement alors 3 GEP sont autorisés à se mettre en marche.Fd
- Si un des 2 transformateurs est en fonctionnement alors 1GEP autorisé à se mettre en marche.
- Si le groupe électrogène est en fonctionnement alors 2GEP autorisés à se mettre en marche.

- Si le nombre de GEP autorisés est supérieur au nombre de GEP en marche alors l'autorisation de marche de GEP est activé (désactivé dans le cas contraire).
- Si aucun GEP n'est en phase de démarrage ou en phase d'arrêt alors l'autorisation de démarrage de GEP activé (désactivé dans le cas contraire).

IV.5.2 le programme du DFB autorisation

Les signaux :

autorisation		<DFB>		
<entrées>				
tansfo_T1	1	BOOL		
tansfo_T2	2	BOOL		
Groupe_EG	3	BOOL		
GEP1_en_DEM	4	BOOL		
GEP2_en_DEM	5	BOOL		
GEP3_en_DEM	6	BOOL		
GEP4_en_DEM	7	BOOL		
GEP5_en_DEM	8	BOOL		
GEP6_en_DEM	9	BOOL		
GEP1_en_ARRET	10	BOOL		
GEP2_en_ARRET	11	BOOL		
GEP3_en_ARRET	12	BOOL		
GEP4_en_ARRET	13	BOOL		
GEP5_en_ARRET	14	BOOL		
GEP6_en_ARRET	15	BOOL		
GEP1_en_Marche	16	BOOL		
GEP2_en_Marche	17	BOOL		
GEP3_en_Marche	18	BOOL		
GEP4_en_Marche	19	BOOL		
GEP5_en_Marche	20	BOOL		
GEP6_en_Marche	21	BOOL		
<sorties>				
Autoris	1	BOOL		
<entrées/sorties>				
<public>				
<privé>				
Nbr_GEP_aut		UINT		
Nbr_GEP_en_march		UINT		
aut_DEM_GEP		BOOL		
aut_march		BOOL		
<sections>				

Le programme :

La section du programme est écrite en langage structuré (ST)

```
(*Ce programme est le DFB autorisation
Ce DFB comprend 21 entrées et 1 sortie :
Les 21 entrées :
tansfo_T1: le 1er Transformateur en fonctionnement.
tansfo_T2: le 2eme Transformateur en fonctionnement.
Groupe_EG : groupe électrogène en fonctionnement.
GEP1(2,3,4,5et6)_en_DEM : le GEP est en phase de démarrage.
GEP1(2,3,4,5et6)_en_ARRET: le GEP est en phase d'arrêt.
GEP1(2,3,4,5et6)_en_Marche : le GEP est en marche.
La sortie :
Autoris: signale qui donne l'autorisation de démarrage au GEP.*)

(*autorisation par rapport aux sources d'énergie*)
(*si aucun transfo n'est en fonctionnement et GEP aussi*)
(*alors aucun GEP n'est autorisé à fonctionner*)
IF (not tansfo_T1) AND (NOT tansfo_T2) AND (NOT Groupe_EG ) THEN
Nbr_GEP_aut:=0;
END_IF;
(*si un seul transfo fonctionne alors un seul GEP est autorisé*)
IF (tansfo_T1 AND NOT tansfo_T2) OR (tansfo_T2 AND NOT tansfo_T1) THEN
Nbr_GEP_aut:=1;
END_IF;
(*si les é transfos fonctionnent : 2 GEP sont autorisés*)
IF (tansfo_T1 AND tansfo_T2) THEN
Nbr_GEP_aut:=3;
END_IF;
(*s'il n'y a que le Groupe électrogène en fonctionnement : 2 GEP sont autorisés*)
IF Groupe_EG THEN
Nbr_GEP_aut:=2;
END_IF;
(*nombre total des GEP en marche = la somme arithmétique de ces derniers*)
Nbr_GEP_en_march:=BOOL_TO_UINT(GEP1_en_Marche)+BOOL_TO_UINT(GEP2_en_Marche)+
BOOL_TO_UINT(GEP3_en_Marche)+BOOL_TO_UINT(GEP4_en_Marche)+BOOL_TO_UINT(GEP5_en_Marche)+
BOOL_TO_UINT(GEP6_en_Marche);
(*on autorise le démarrage d'un autre GEP si le nombre total des GEP en
 marche est inferieur aux nombres des GEP autorisés à fonctionner*)
IF (Nbr_GEP_en_march<Nbr_GEP_aut)THEN
aut_march:=1;
  ELSE
aut_march:=0;
END_IF;
(*On autorise un autre GEP à démarrer s'il n'y a aucun autre en phase de démarrage *)
aut_DEM_GEP:=NOT(GEP1_en_DEM OR GEP2_en_DEM OR GEP3_en_DEM OR GEP4_en_DEM
OR GEP5_en_DEM OR GEP6_en_DEM OR GEP1_en_ARRET OR GEP2_en_ARRET OR GEP3_en_ARRET
OR GEP4_en_ARRET OR GEP5_en_ARRET OR GEP6_en_ARRET);
(*Et enfin l'autorisation marche du GEP est combinaison des 2 dernières autorisations *)
Autoris:=aut_DEM_GEP AND aut_march;
```

IV.6 Le DFB défaut

Pour des raisons de sécurité, le GEP ne doit démarrer que si tous les systèmes fonctionnent correctement. Dans notre station plusieurs facteurs de défaut doivent être vérifiés avant le démarrage.

IV.6.1 cahier de charge du DFB défaut

Le but de ce DFB est la signalisation de 3 défauts principaux :

- Défaut de température des bobinages du moteur de la pompe, ce dernier est équipé de trois capteurs de température.
- Défaut de température des paliers (deux capteurs de température).
- Défaut de vibration du GEP.

Ce DFB présent 16 signaux en entrée et 4 en sorties

Les 16 entrées :

- temperatureB1, 2 et 3 : Températures des bobines de la pompe.
- temperatureB_seuil : Température seuil des bobines de la pompe.
- temperatureP1 et 2 : Températures des paliers de la pompe.
- temperatureP_seuil : Température seuil des paliers de la pompe.
- vibration : vibration mécanique de la pompe.
- vibration_seuil : vibration seuil de la pompe.
- defaut_van : défaut de la vanne.
- disjoncteur_van : disjoncteur vanne.
- disjoncteur_gep : disjoncteur GEP.
- defaut_isolem : défaut isolement.
- defaut_demmareur : défaut démarreur progressive (ALTISTART).
- GEP_sous_tension : GEP sous-tension (présence d'alimentation).
- acquit : pour l'acquittement des défauts.

Les 4 sorties :

- defaut_temperatureB : défaut de température des bobinages.
- defaut_temperatureP : défaut de température des paliers.
- defaut_vibration : pour dire que la vibration a dépassé le seuil.
- defaut : qui résume tout les défauts.

Le DFB d'autorisation doit

- envoyer 4 signaux signalant l'existence d'un défaut et le préciser.

IV.6.2 le programme du DFB défaut

Les signaux :

defaut		<DFB>
<entrées>		
temperatureB1	1	INT
temperatureB2	2	INT
temperatureB3	3	INT
temperatureB_seuil	4	INT
temperatureP1	5	INT
temperatureP2	6	INT
temperatureP_seuil	7	INT
vibration	8	INT
vibration_seuil	9	INT
defaut_van	10	BOOL
disjoncteur_van	11	BOOL
disjoncteur_gep	12	BOOL
defaut_isolem	13	BOOL
defaut_demmareur	14	BOOL
GEP_sous_tension	15	BOOL
acquit	16	BOOL
<sorties>		
defaut_temperatureB	1	BOOL
defaut_temperatureP	2	BOOL
defaut_vibration	3	BOOL
defaut	4	BOOL
<entrées/sorties>		
<public>		
<privé>		
defaut_analogi		BOOL
defaut_logique		BOOL
defaut_TB		BOOL
defaut_TP		BOOL
defaut_vib		BOOL
<sections>		
prg_defaut		<ST>

Le programme :

La section du programme est écrite en langage structuré (ST)

```
(*Ce programme est le DFB des défauts
Ce DFB comprend 16 entrées et 4 sorties :
Les 16 entrées :
temperatureB1,2 et 3 : Températures des bobines de la pompe
temperatureB_seuil : Température seuil des bobines de la pompe
temperatureP1 et 2 : Températures des paliers de la pompe
temperatureP_seuil : Température seuil des paliers de la pompe
vibration : vibration mécanique de la pompe
vibration_seuil : vibration seuil de la pompe
defaut_van : défaut de la vanne
disjoncteur_van : disjoncteur vanne
disjoncteur_gep : disjoncteur GEP
defaut_isolem : défaut isolement
defaut_demmareur : défaut demmareur progressive (ALTISTART)
GEP_sous_tension : GEP soustension(présence d'alimentation )
acquit : pour l'acquitement des défauts
```

```
Les 4 sorties :
defaut_temperatureB : défaut de température des bobinages
defaut_temperatureP : défaut de température des paliers
defaut_vibration : pour dire que la vibration a dépassé le seuil
defaut : qui resume tout les défaut

(*defauts analogiques*)
(*defaut_TB s'il y a un dépassement de seuil de la température de l'un des bobines*)
IF (temperatureB1>temperatureB_seuil)OR (temperatureB2>temperatureB_seuil)OR
                (temperatureB3>temperatureB_seuil) THEN
    set(defaut_TB);
(*le defaut disparaisse après appuie sur acquit (avec absence du defaut_TB) *)
ELSIF acquit THEN
    reset(defaut_TB);
END_IF;
(*defaut_TB s'il y a un dépassement de seuil de la température de l'un des paliers*)
IF (temperatureP1>temperatureP_seuil)OR (temperatureP2>temperatureP_seuil) THEN
    set(defaut_TP);
ELSIF acquit THEN
(*le defaut disparaisse après appuie sur acquit (avec absence du defaut_TP) *)
    reset(defaut_TP);
END_IF;
(*le defaut vibration en cas de depassement de seuil de ce dernier*)
IF (vibration>vibration_seuil) THEN
    set(defaut_vib);
(*le defaut disparaisse après appuie sur acquit (avec absence du defaut_vib) *)
ELSIF acquit THEN
    reset(defaut_vib);
END_IF;
(*Le défaut analogique final est la combinaison de tout les défauts analogiques*)
defaut_analogi:=(defaut_TB or defaut_TP or defaut_vib);
(*le défaut logique est la combinaison de tout les défauts analogiques*)
IF (defaut_van OR disjoncteur_van OR disjoncteur_gep OR defaut_isolem OR defaut_demmareur
                OR GEP_sous_tension) THEN
    set(defaut_logique);
(*le defaut disparaisse après appuie sur acquit (avec absence du defaut_vib) *)
ELSIF acquit THEN
    reset(defaut_logique);
END_IF;
(* affectation des sorties*)
defaut_temperatureB:=defaut_TB;
defaut_temperatureP:=defaut_TP;
defaut_vibration:=defaut_vib;
defaut:=defaut_analogi OR defaut_logique;
```

IV.7 Le DFB GEP

le groupe électropompe est constitué de la pompe et son moteur , le GEP envoie deux signaux , un pour indiquer que son démarrage est terminé et un autre pour indiquer la fin de son arrêt .et sachant que la vanne envoie aussi 2 signaux (complètement ouverte et complètement fermée) ; le DFB GEP assurera un bon fonctionnement de la pompe (un démarrage et un arrêt sécurisé) .

IV.7.1 cahier de charge du DFB GEP

Le DFB GEP est constitué de 9 entrées et de 9 sorties.

Les 9 entrées :

- marche : commande marche du GEP (signal envoyé par l'utilisateur ou un autre programme)
- arret : commande arrêt du GEP (envoyé par l'utilisateur ou un autre programme)
- FIN_DEM : signal fin démarrage du GEP (envoyé par le GEP)
- FIN_Arret : pour dire que le GEP est complètement arrêté (envoyé par le GEP)
- van_fermee : fin de course vanne fermée
- van_ouverte : fin de course vanne ouverte
- reset_defaut : bouton reset défaut
- aut_GEP : signal autorisation démarrage du GEP (envoyé par le DFB autorisation)
- Defaut : signal défaut (envoyé par le DFB défaut)

LES 9 sorties :

- ouvrir_van : signal d'ouverture de la vanne
- fermer_van : signal de fermeture de la vanne
- marche_arret_GEP : commande d'arrêt du GEP
- reset_defaut_GEP : commande de démarrage du GEP
- voyant_marche_GEP : voyant marche du GEP
- voyant_defaut_GEP : voyant défaut du GEP
- GEP_en_DEM : pour dire que le GEP est en phase de démarrage
- GEP_en_ARRET : pour dire que le GEP est en phase d'arrêt
- GEP_en_Marche : pour dire que le GEP est en fonctionnement

Le DFB anti bélier doit assurer :

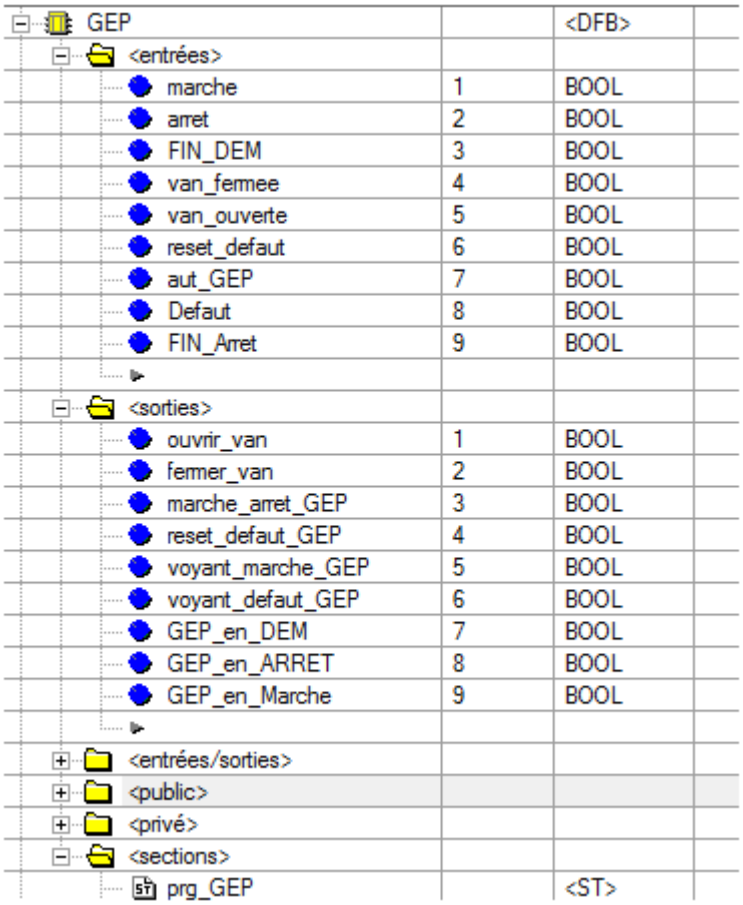
- Un démarrage à vide et un arrêt à vide de la pompe :

- Le démarrage à vide (sans charge) pour diminuer en max le couple de démarrage et assurer un démarrage rapide de la pompe, pour cela le démarrage se fait toujours avec vanne fermée. Après recevoir du signal FIN_DEM on pourra ouvrir la vanne.
- L'arrêt à vide (vanne fermée) pour éviter que la pompe absorbe le coup de bélier qui pourra la détruire mais aussi le reculement de l'eau (l'expérience a monté qu'un arrêt avec une vanne ouverte provoque le reculement de l'eau à travers la pompe avec des grandes vitesses pouvant détruire complètement le GEP).

- Envoie de 3 signaux indiquant l'état du GEP : GEP_en_DEM, GEP_en_Arret, GEP_en_Marche.

IV.7.2 le programme du DFB GEP

Les signaux :



GEP		<DFB>
<entrées>		
marche	1	BOOL
arret	2	BOOL
FIN_DEM	3	BOOL
van_fermee	4	BOOL
van_ouverte	5	BOOL
reset_defaut	6	BOOL
aut_GEP	7	BOOL
Defaut	8	BOOL
FIN_Arret	9	BOOL
<sorties>		
ouvrir_van	1	BOOL
fermer_van	2	BOOL
marche_arret_GEP	3	BOOL
reset_defaut_GEP	4	BOOL
voyant_marche_GEP	5	BOOL
voyant_defaut_GEP	6	BOOL
GEP_en_DEM	7	BOOL
GEP_en_ARRET	8	BOOL
GEP_en_Marche	9	BOOL
<entrées/sorties>		
<public>		
<privé>		
<sections>		
prg_GEP		<ST>

Le programme :

La section du programme est écrite en langage structuré (ST)

```
(*Ce programme est le DFB qui gère  le GEP
Ce DFB comprend 9 entrées et 9 sorties :
Les 9 entrées :
marche : commande marche du GEP
arret : commande arrêt du GEP
FIN_DEM : signal fin démarrage du GEP
van_fermee : fin de course vanne fermée
van_ouverte : fin de course vanne ouverte
reset_defaut : bouton reset défaut
aut_GEP : signal autorisation démarrage du GEP
Defaut : signal défaut
FIN_Arret : pour dire que le GEP est complètement arrêté
LES 9 sorties :
ouvrir_van : signal d'ouverture de la vanne
fermer_van : signal de fermeture de la vanne
marche_arret_GEP : commande d'arrêt du GEP
reset_defaut_GEP : commande de démmrage du GEP
voyant_marche_GEP : voyant marche du GEP
voyant_defaut_GEP : voyant défaut du GEP
GEP_en_DEM : pour dire que le GEP est en phase de démmarage
GEP_en_ARRET : pour dire que le GEP est en phase d'arrêt
GEP_en_Marche : pour dire que le GEP est en fonctionnement*)
```

```
(*démmarage de GEP*)
(*fermeture  de la vanne avant le démmarage de GEP*)
IF marche AND (NOT arret)AND (NOT defaut)AND (NOT FIN dem) AND aut GEP AND van ouverte THEN
Set(fermer van);
reset(ouvrir van);
END_IF;
(*si la vanne est fermée on démmare le GEP *)
IF marche AND (NOT arret)AND(NOT defaut)AND (NOT FIN dem) AND aut GEP AND van fermee THEN
set(marche arret GEP);
set(GEP en DEM);
END_IF;
(*ouverture de la vanne après le demmarage de GEP*)
IF FIN DEM  AND (NOT arret)THEN
reset(GEP en DEM);
set(GEP en Marche);
set (ouvrir van);
END_IF;
(*l'envoie du signal ouvrir_van est arrêté si la vanne est complètement ouverte *)
IF van ouverte  THEN
reset (ouvrir van);
END_IF;
(*fermeture  de la vanne avant l'arrêt du GEP*)
IF arret AND(NOT defaut)AND aut GEP AND van ouverte THEN
reset(ouvrir van);
Set(fermer van);
set(GEP en Arret);
END_IF;
(*si la vanne est fermée on arrête le GEP *)
IF arret AND (NOT defaut)AND aut GEP AND van fermee THEN
reset(marche arret GEP);
set(GEP en arret);
END_IF;
(*et on enlève la commande d'arrêt*)
IF FIN Arret THEN
reset(GEP en arret);
reset(GEP en Marche);
END_IF;
(*l'envoie du signal fermer_van est arrêté si la vanne est complètement fermée *)
IF van fermee THEN
reset (fermer van);
END_IF;
(* affectation des sorties*)
 voyant marche GEP:=FIN DEM;
 voyant defaut GEP:=defaut;
```

IV.7.3 la simulation des DFB : autorisation, défaut et GEP

Pour assurer le bon fonctionnement des 3 blocs DFB : autorisation, défaut et GEP ; un programme FBD est mis en place dont on va lui affecter les entrées / sorties du programme de simulation, une autre section en langage LADDER permettant de simuler le fonctionnement du GEP.

Les autres systèmes vont être simulés à travers l'écran d'exploitation.

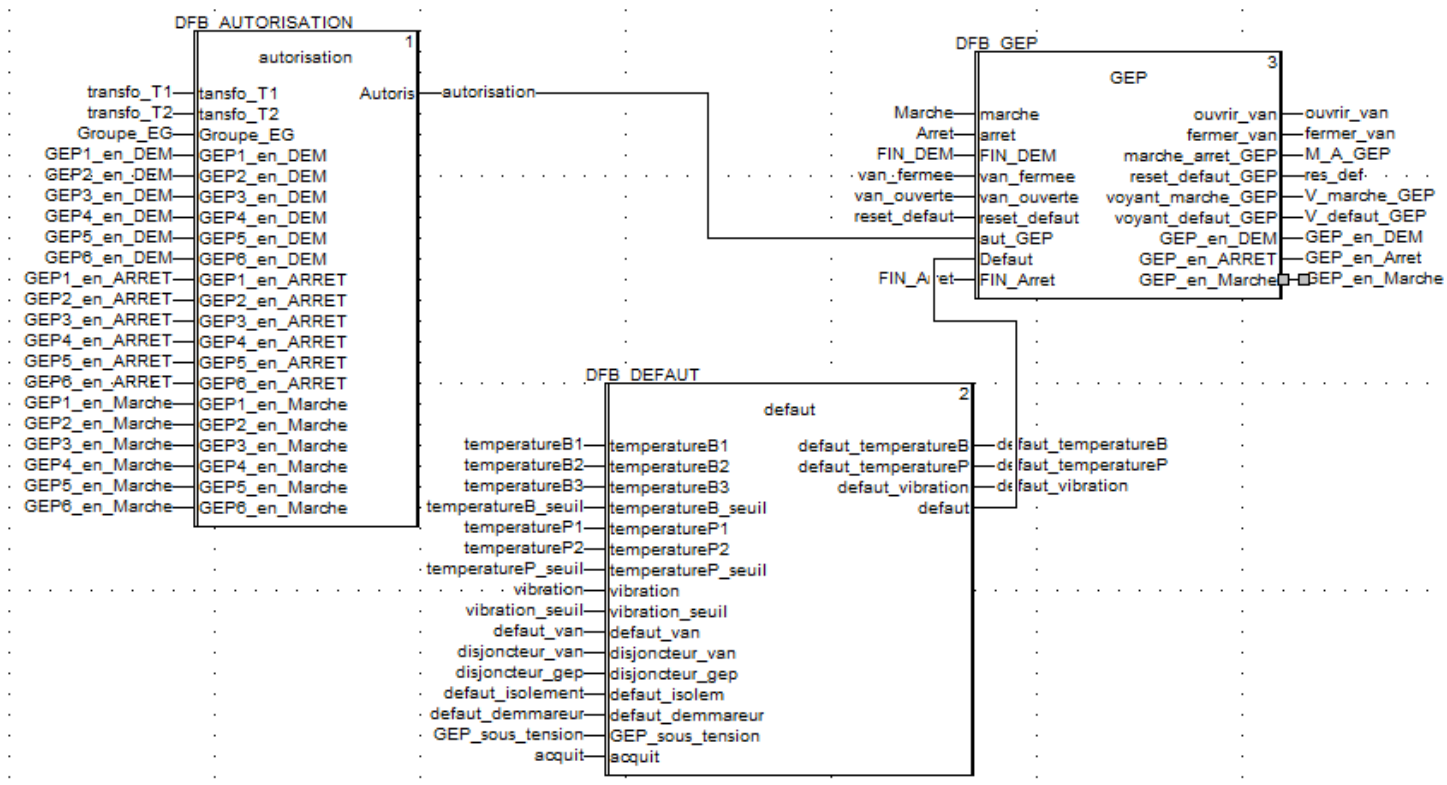


Figure IV.7: *la section de simulation FBD des trois DFB (GEP , défaut et autorisation)*

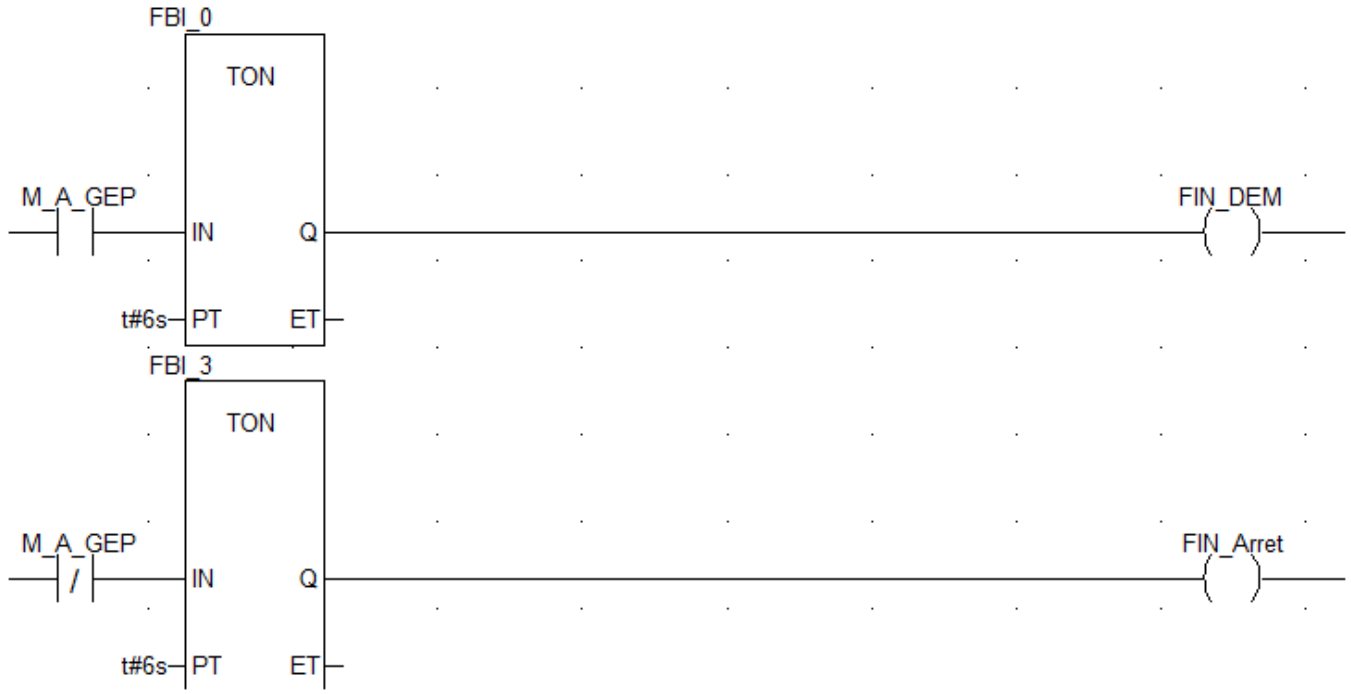


Figure IV.8: *la section de simulation LADDER des trois DFB (GEP , défaut et autorisation)*

Pour le bien visualiser le système on utilise un écran d'exploitation.

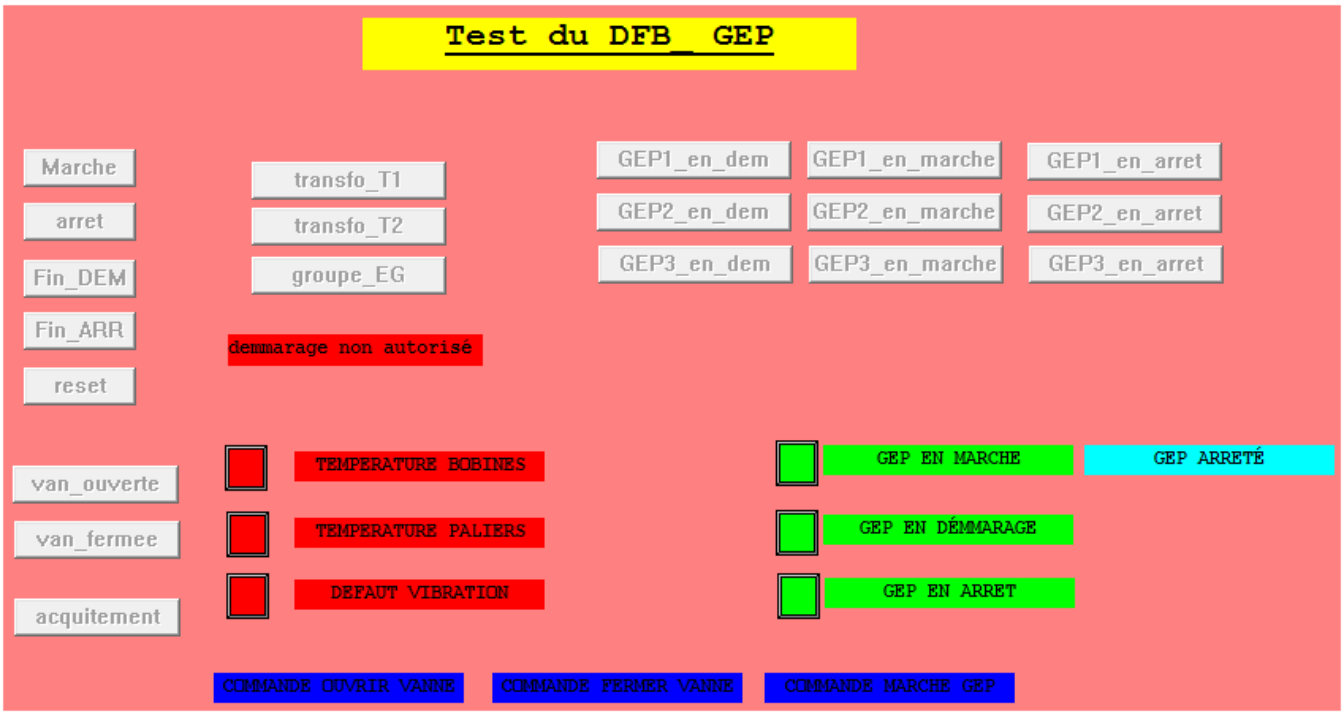


Figure IV.9: *l'écran d'exploitation des trois DFB.*

IV.8 Conclusion

La programmation avec les blocs DFB permette d'écrire des programme volumineux juste en insérant les instances des DFB, ces DFB doivent être vérifiés avant les mettre en point, Unity Pro nous offre cette possibilité avec ces outilles de simulation : table et écran d'exploitation.

Chapitre V

Le logiciel de construction des HMI Vijeo Designer

V.1 Introduction

Dans un projet industriel , il est toujours nécessaire de faciliter le contrôle des différentes entités de l'usine (ou station) et ceci en introduisant des interfaces de communication et de supervision par le moyen des Pc ou des écrans tactiles programmés par des logiciel d'interface homme machine. Dans notre cas la supervision de la station de pompage se fait avec le logiciel Vijeo Designer.

V.2 Présentation de logiciel Vijeo designer

Vijeo Designer est un logiciel de pointe permettant de réaliser des écrans opérateur et de configurer les paramètres opérationnels des périphériques d'Interface Homme Machine (IHM). Il fournit tous les outils nécessaires à la conception d'un projet IHM, de l'acquisition des données jusqu'à la création et à la visualisation de synoptiques animés. [11]

Les applications logicielles

Vijeo-Designer est constitué de deux applications logicielles :

- Vijeo-Designer, le logiciel de développement d'écrans.
- Vijeo-Designer Runtime, le logiciel d'exécution de projets.

Vijeo-Designer

L'Editeur Vijeo-Designer est l'environnement dans lequel vous développez l'application utilisateur IHM, avant de la transférer vers la machine cible.

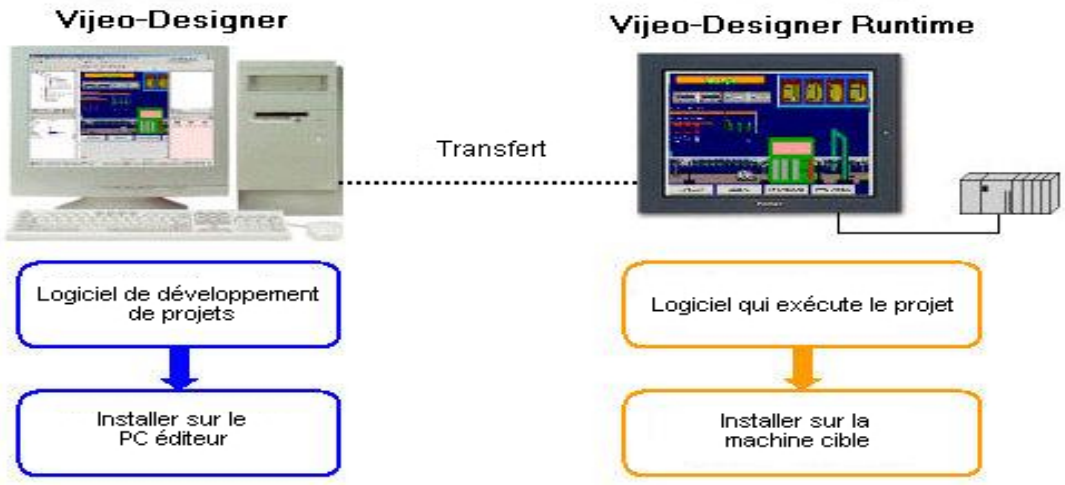


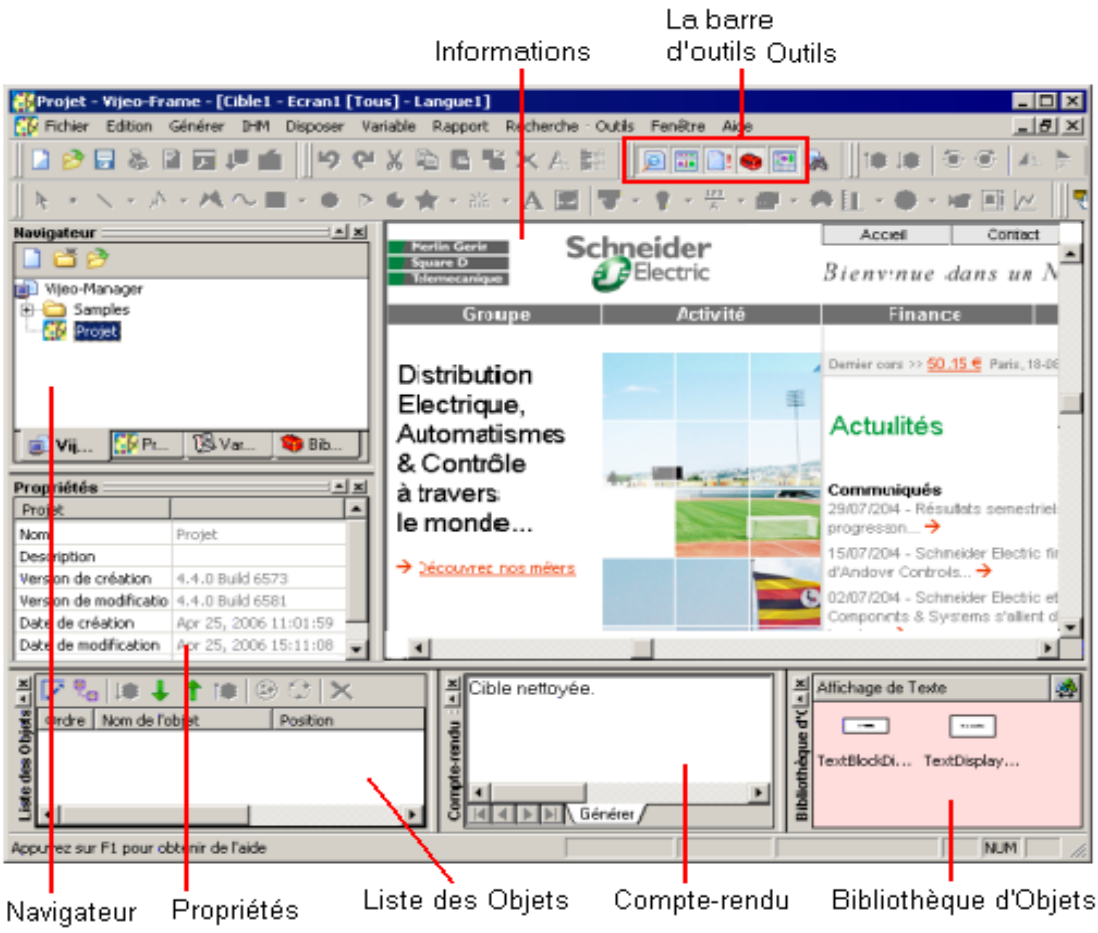
Figure V.1 : *compatibilité entre Vijeo-Designer et Vijeo-Designer Runtime* [11]

Vijeo-Designer Runtime

- Une fois l'application utilisateur IHM créée dans l'Editeur Vijeo-Designer, vous pouvez la transférer vers la machine cible, c'est-à-dire l'ordinateur sur lequel vous allez exécuter et afficher vos applications d'écrans avec Vijeo-Designer Runtime.
- Pour que l'application utilisateur fonctionne correctement, Vijeo-Designer Runtime doit être installé sur le matériel qui peut être utilisé comme écran de contrôle, moniteur ou écran tactile IHM.

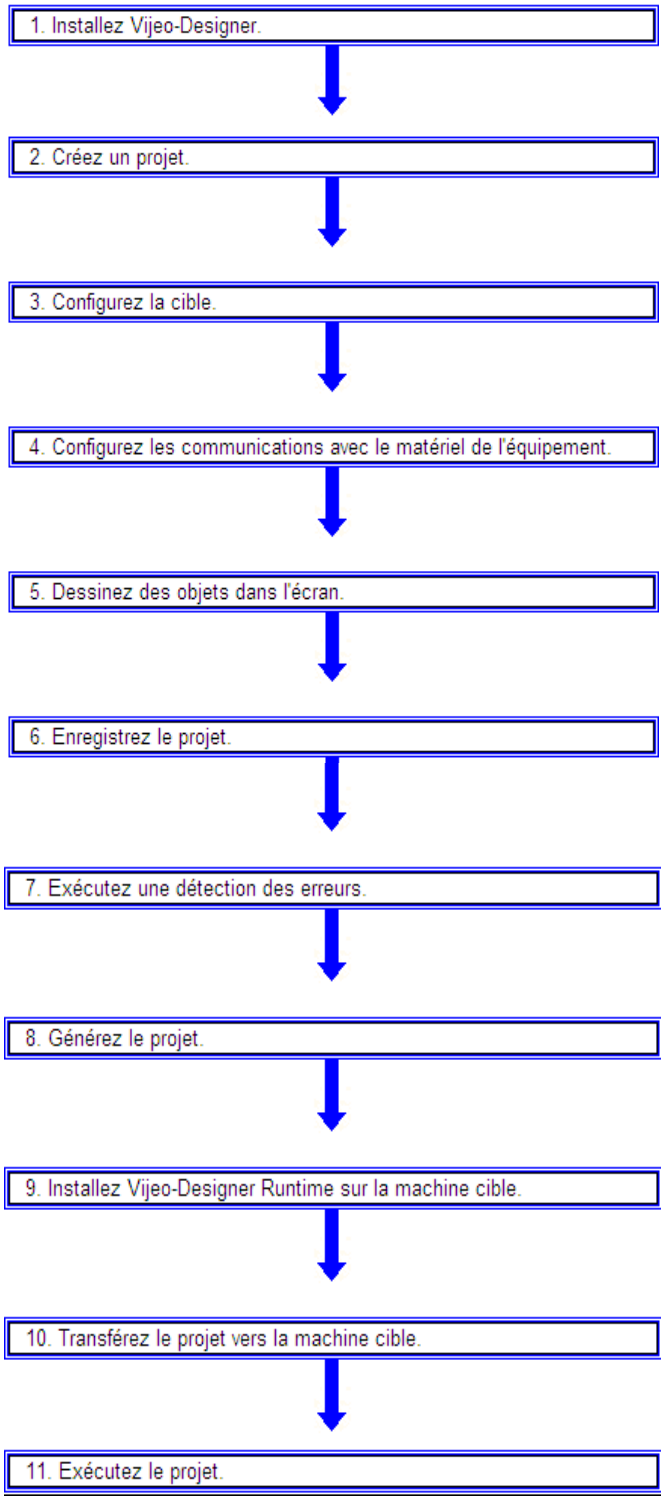
V.3 Principaux outils de Vijeo Designer

Vijeo-Designer utilise plusieurs fenêtres appelées outils afin de créer des projets de manière performante. Ces outils affichent des informations concernant le projet et les objets sur lesquels vous travaillez. Vous pouvez les redimensionner, les déplacer, les afficher ou encore les masquer.



Comment développer un projet

Cette section aborde la procédure d'utilisation de Vijeo-Designer, depuis l'installation jusqu'à l'exécution du projet dans Runtime



V.4 Création de projets

V.4.1 A propos des projets

Un projet est un fichier que vous créez dans Vijeo-Designer. comporte les informations nécessaires à la création d'un environnement pour votre application utilisateur (dessins, alarmes et informations sur le matériel). l'exécution de projets se fait a la machine cible(Magelis) avec Vijeo-Designer Runtime. [11]

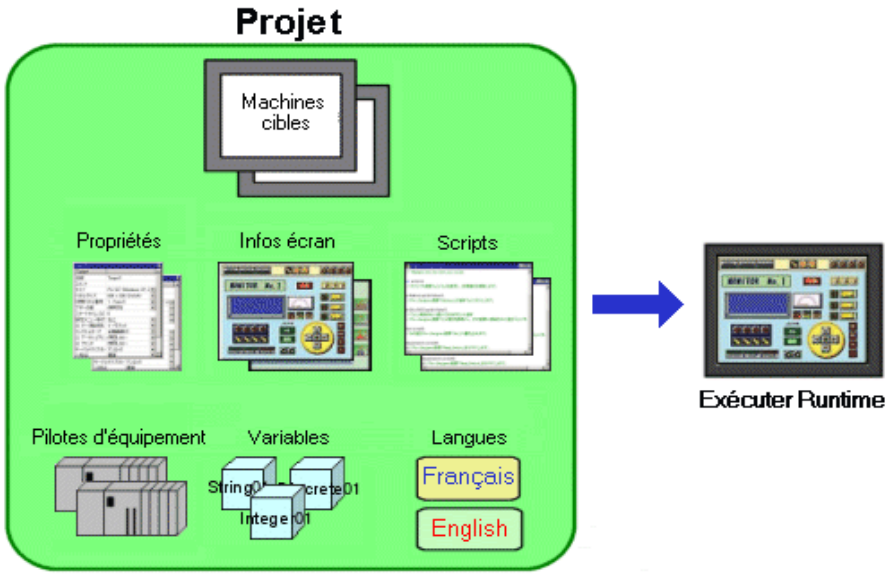


Figure V.2: *les différents éléments constituant un projet* [11]

V. 4.2 les étapes de Création d'un projet

Cette section décrit l'assistant qui s'affiche lorsque vous lancez Vijeo-Designer pour la première fois. L'assistant vous guide créer un projet dans Vijeo-Designer.

Pour démarrer Vijeo-Designer, dans le menu Démarrage de Windows, pointez la souris sur Programmes, Schneider Electric, Vijeo-Designer, puis sélectionnez Vijeo-Designer, ou bien cliquer deux fois sur l'icône Vijeo-Designer du bureau

Les étapes de création d'un projet par l'assistant Vijeo-Designer sont décrites dans l'annexe C.

V.5 Configuration de la cible

Les cibles représentent des interfaces homme-machine (IHM) qui exécutent des applications utilisateur transférées depuis Vijeo-Designer.

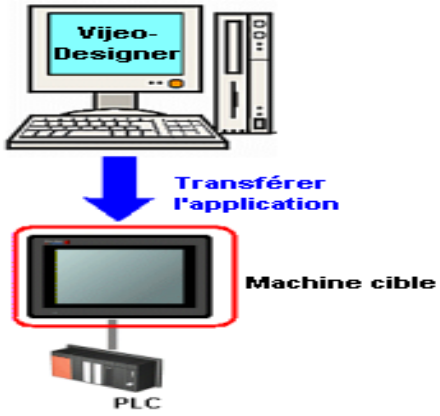
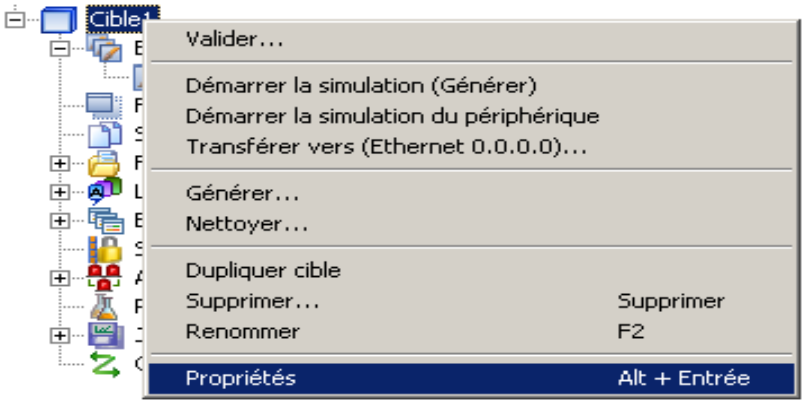


Figure V.3: *le rôle de la machine cible* [11]

Procédure de Configuration d'une cible

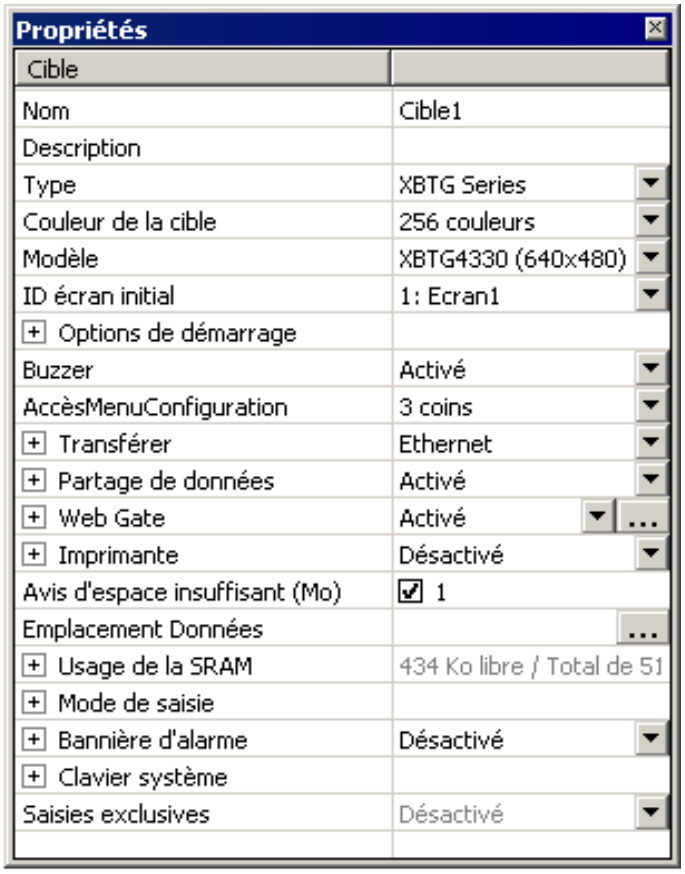
La procédure de configuration des nœuds de la cible dans l'Inspecteur de propriétés est le suivant :

1. Dans l'onglet Projet de la fenêtre du Navigateur, cliquez avec le bouton droit de la souris sur le nœud de la cible, puis sélectionnez Propriétés



.

1. Dans l'Inspecteur de propriétés, définissez les propriétés de la cible.



V.6 Communications

Pour communiquer avec des équipements, des variateurs et d'autres équipements, connectez-les au port série (RS-232C/RS-422), au port Ethernet ou au module/à la carte de communication de la machine cible, puis ajoutez un pilote. Vijeo-Designer utilise des pilotes pour activer les communications avec l'équipement. Cela évite de créer des programmes de communication complexes.

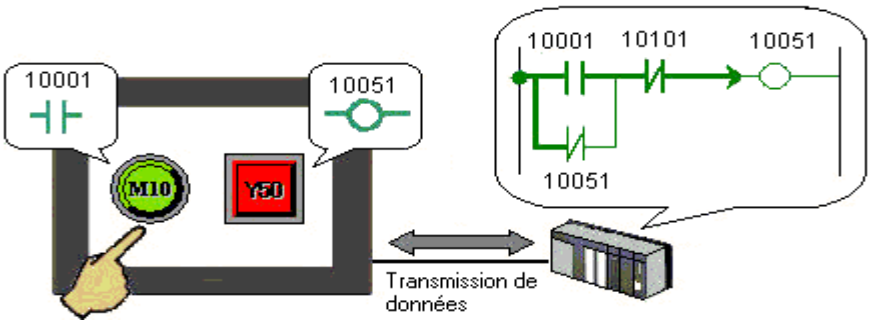
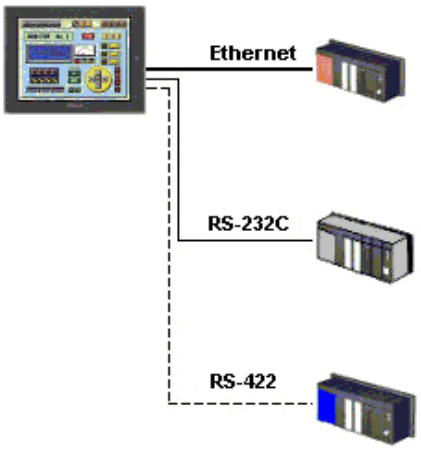


Figure V.5: *communication de la machines cibles avec des équipements* [5]



Grâce au système de communication hautement évolutif de Vijeo-Designer, vous pouvez connecter plusieurs équipements à une même machine cible. Le modèle, le fabricant et les types de connexions de cet équipement peuvent varier. Il suffit d'ajouter les pilotes appropriés au projet**.** [11]

Configuration de l'équipement

Pour configurer les communications entre la machine cible et l'équipement, vous devez : ajouter et configurer le pilote dans Vijeo-Designer, faire correspondre les paramètres de communication de Vijeo-Designer avec ceux de l'équipement, et transférer l'application utilisateur vers la machine cible**.**

Le tableau suivant décrit la procédure de configuration de l'équipement.. [11]

1. Dans l'onglet Projet de la fenêtre du Navigateur, ajoutez un nouveau pilote.
2. Dans la boîte de dialogue Nouveau pilote, sélectionnez le fabricant, le pilote et l'équipement.
3. Configurez les paramètres de communication dans la boîte de dialogue Configuration du pilote. Définissez les paramètres comme la vitesse de transmission et la longueur des données.
4. Identifiez l'équipement dans la boîte de dialogue Configuration de l'équipement.
5. Créez une variable, attribuez-la au groupe de scrutation associé avec le pilote créé à l'étape 2 et définissez les adresses de périphérique pour la variable.
6. Utilisez la variable créée à l'étape 5 dans l'animation, les objets et les autres fonctions.
7. Connectez un câble entre la machine cible et l'équipement. De plus, vérifiez que les paramètres définis dans Vijeo-Designer correspondent à ceux définis sur l'équipement.
8. Transférez l'application utilisateur sur la machine cible, puis exécutez-la.

V.7 Objets graphiques

L'Editeur Vijeo-Designer utilise deux types d'objets graphiques : des outils de dessin et des objets. La Bibliothèque d'objets Vijeo-Designer est une bibliothèque d'images prêtes à l'emploi représentant des équipements industriels et d'autres objets matériels.

les objets graphiques que vous pouvez placer dans un écran et utiliser individuellement ou en combinaison avec d'autres objets pour créer et modifier une représentation visuelle de votre conception sont les suivants :

➢ Outils de dessin d'objets graphiques

La barre d'outils Objets graphiques comprend deux types d'outils de dessin :

- **Outils et formes basiques d'objets graphiques** qu'ils sont les suivants :

  - ✓ Outil de **sélection** (principal outil de modification de Vijeo-Designer) permettant de sélectionner des objets graphiques.
  - ✓ Formes basiques (les outils **Point**, **Ligne**, **Rectangle**, **Ellipse**, **Arc**, **Secteur**, **Polyligne**, **Polygone**, **Polygone symétrique**, **Courbe Bézier**, et **Echelle**) pour concevoir, dessiner et manipuler votre architecture de projet. Chaque forme est décrite dans cette section dans la partie Dessin d'objets graphiques.
  - ✓ Outil **Objet texte** : pour dessiner des étiquettes et des messages textuels.
  - ✓ Outil **Objet image** : pour importer et coller des images provenant de sources externes dans l'écran graphique.

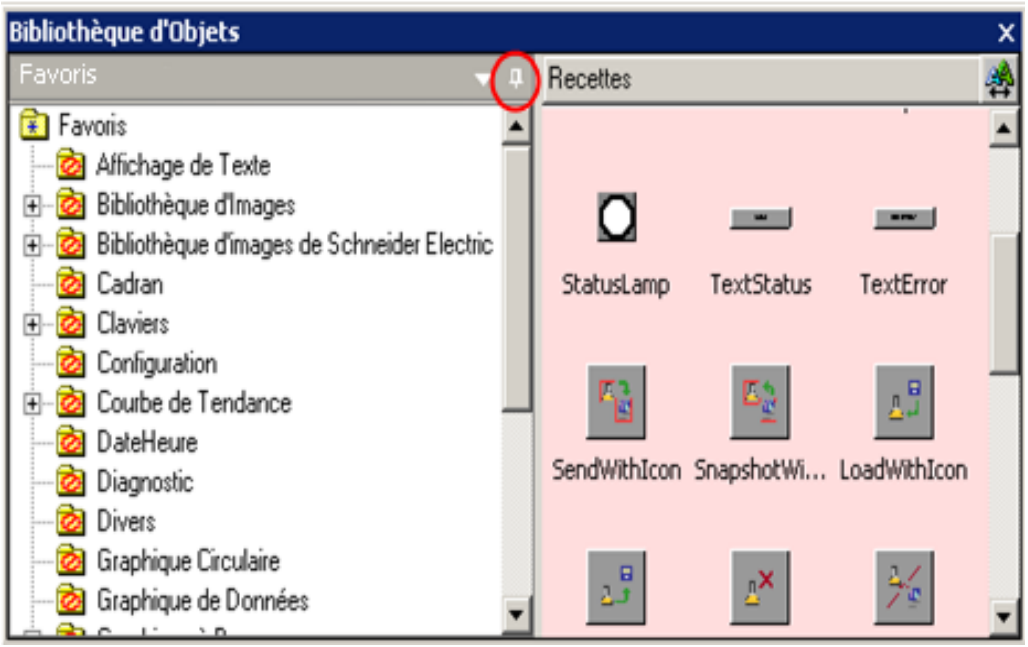
- **Objets qui transmettent des informations :** les objets Commutateur, Voyant, Affichage des données, Courbe de tendance et Résumé d'alarme.

➢ La Bibliothèque d'objets

La Bibliothèque d'objets est une hiérarchie de dossiers servant à stocker dessins, animations, écrans, groupes d'alarmes, objets graphiques, objets de bibliothèques d'images et objets de bibliothèque d'objets.

Lorsque vous stockez un objet graphique, la bibliothèque d'objets stocke également les variables associées à cet objet. [11]

La bibliothèque d'objets contient les dossiers suivant :



V.8 Variables

Les variables sont des espaces nommés de la mémoire qui stockent des données. Une variable peut être considérée comme un contenant de valeurs de données.

La variable de Vijeo-Designer, qui est affectée à une adresse de périphérique, est mise à jour lors de chaque modification des données au niveau de l'équipement. Si vous associez les adresses de périphérique à des variables, Vijeo-Designer peut lire et afficher les données provenant de l'équipement.

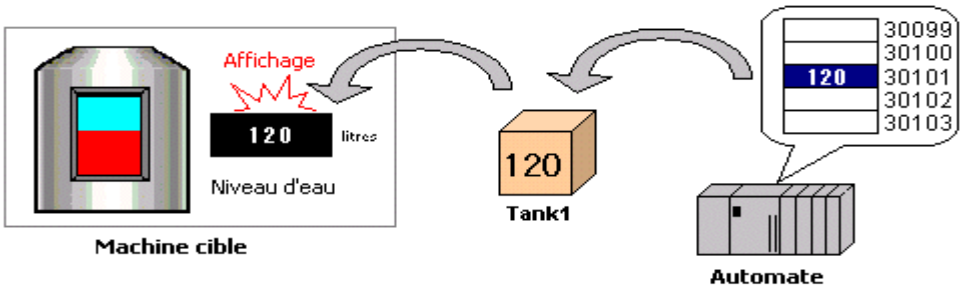


Figure V.4: *affectation d'une variable à une adresse périphérique* [5]

V.8. 1 variable interne et externe

Les variables internes ne sont pas associées à un équipement, et sont utilisées uniquement pour les opérations internes de Vijeo-Designer.

Les variables externes sont associées à une adresse de périphérique.

V.8. 2 Types de variable

Il existe six types de variable Vijeo-Designer : TOR, entier, flottant, chaîne, entier de bloc, et flottant de bloc. Vijeo-Designer propose également un autre type de variable, appelée Structure, c'est-à-dire un dossier qui contient plusieurs variables. [11]

La Création et la validation de variables est décrite sur l'Annexe C

V.9 Les Alarmes

V.9.1 présentation

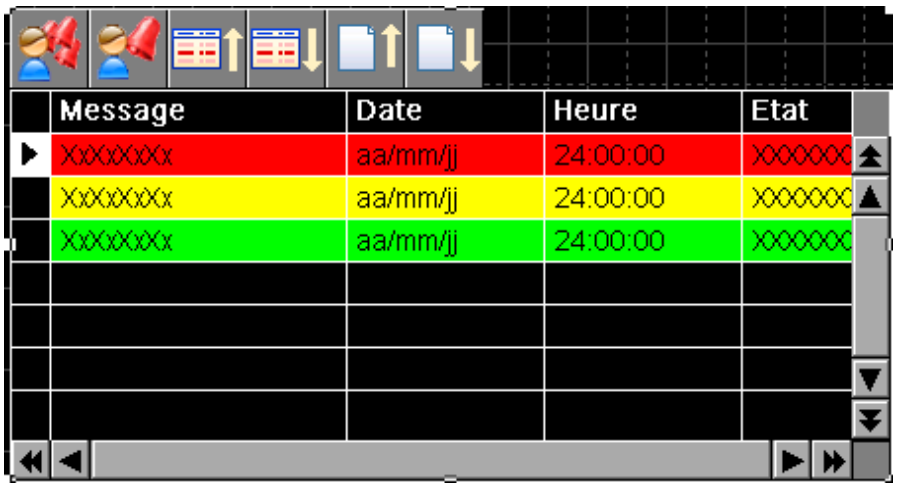
Une alarme est un signal accordé à un message avertissant d'un danger. A ce titre, l'alarme est une information émise afin de provoquer une réaction.

L'alarme nécessite une connaissance préalable du danger. En effet, il n'est pas d'alarme tant que le danger n'est pas connu.

V.9.2 Configuration d'un résumé d'alarme

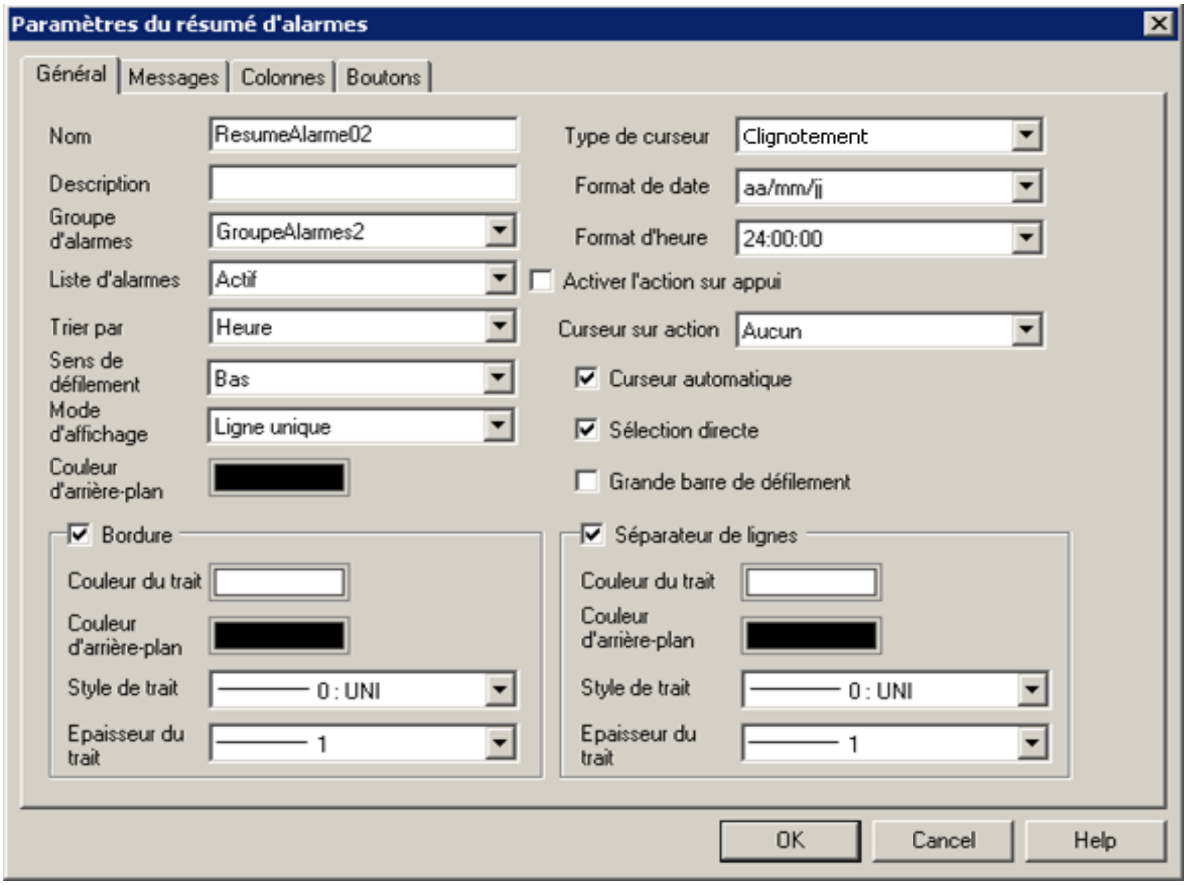
1) Cliquez sur l'icône "**Résumé d'alarme**" dans la barre d'outils et tracez une zone correspondant à l'emplacement de l'objet sur l'écran.

1)Dessinez le résumé d'alarme sur l'écran.



2)Cliquez deux fois sur le résumé d'alarme pour afficher la boîte de dialogue des paramètres du résumé d'alarme.

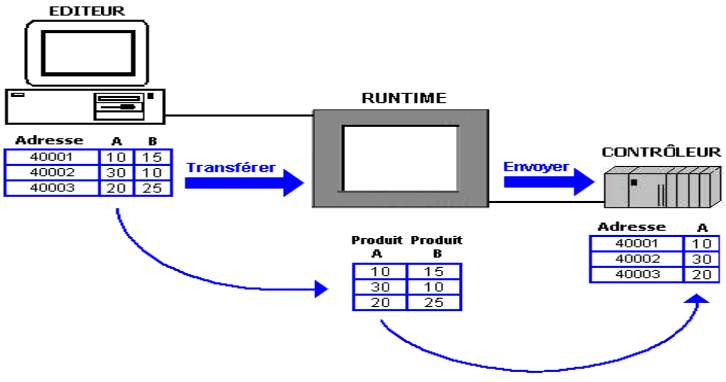
3)Utilisez les onglets du Résumé d'alarmes pour définir les paramètres de résumé d'alarme tels que requis par votre projet. **Onglet Général ,Onglet Messages, Onglet Colonnes et Onglet Boutons**



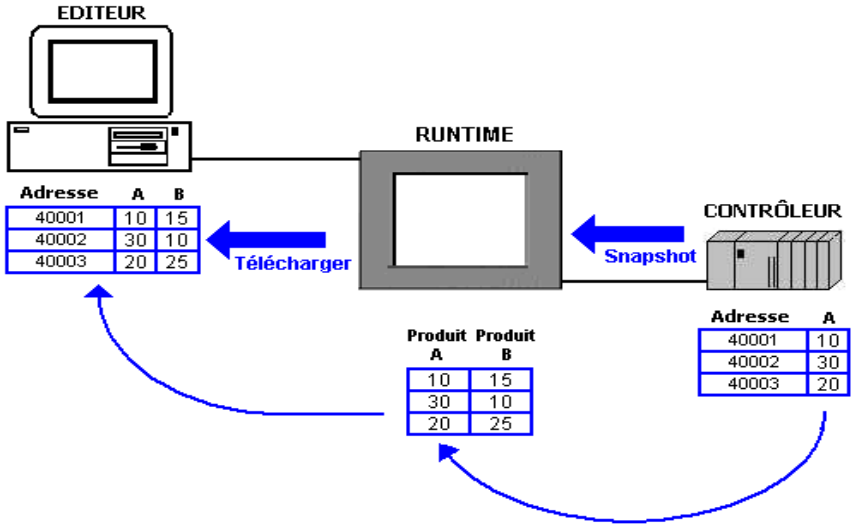
V.10 les Recettes

La fonction Recettes vous permet de travailler avec des valeurs de recette précisées d'adresses de périphériques multiples en même temps. En créant une interface utilisateur simple, vous pouvez maintenir un processus de production consistant en définissant tout simplement les paramètres de production. Désormais, lorsque le flux de travail change ou doit changer, l'opérateur n'a pas besoin d'effectuer un processus complexe. Avec la fonction Recettes, vous pouvez :

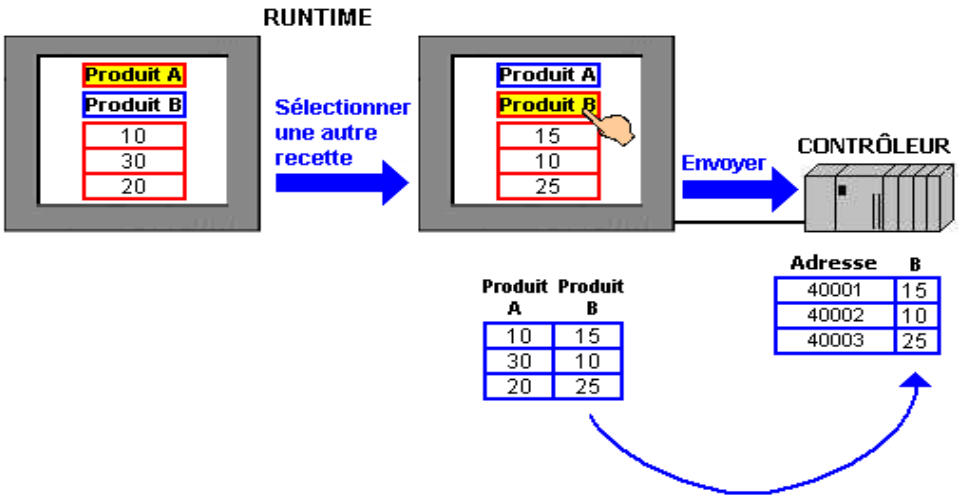
- Ecrivez les valeurs d'une recette (avec l'opération Send) depuis la machine cible vers votre équipement.



- Lisez les valeurs d'une recette (avec l'opération Snapshot) depuis votre équipement vers la machine cible.



- Modifiez l'affichage des diverses recettes lors du runtime, puis sélectionnez une recette et utilisez l'opération Send pour envoyer et écraser les valeurs de recette actuellement dans l'équipement.



Quelques définitions

- **Ingrédient** : il s'agit des éléments particuliers inclus dans une recette. Un ingrédient comporte une étiquette de langue spécifique, la variable associée, et une valeur minimale/maximale. Généralement, chaque recette utilise plusieurs ingrédients. Vous pouvez ajouter jusqu'à un maximum de 1024 ingrédients à chaque recette.

- **Recette** : il s'agit d'un ensemble de variables et de valeurs. Par exemple, Great Pizza Works a un groupe de recettes Pâtes avec cette liste d'ingrédients : farine, eau, levure, huile d'olive, et sel. La recette précisera des valeurs pour chaque ingrédient. Ensuite, à l'aide de ces ingrédients, vous pouvez faire des recettes pour les pâtes à croûte fine, les pâtes à croûte épaisse, ou les pâtes à croûte à bord haut. Vous pouvez créer un maximum de 256 recettes dans chaque groupe de recettes.

- **Groupe de recettes** : il s'agit d'un ensemble de recettes. Chaque groupe de recettes est identifié de façon unique avec un numéro ID (compris entre 1 et 65535) et se voit attribué un nom qui décrit le groupe de recettes. Les utilisateurs sont affectés d'un niveau d'accès qui précise s'ils peuvent visualiser et modifier le groupe de recettes ou non. Vous pouvez créer un maximum de 32 groupes de recette par cible.

- **Commandes de recette** : il s'agit d'un ensemble de variables de commande utilisées avec les groupes de recette. Une commande de recette comprend les variables Numéro de groupe de recettes, Numéro de recette, Etiquette de recette, Déclenchement d'opération, Verrouillage d'opérations, Statut, Erreur, et Droits d'accès.

V.11 Les Scripts

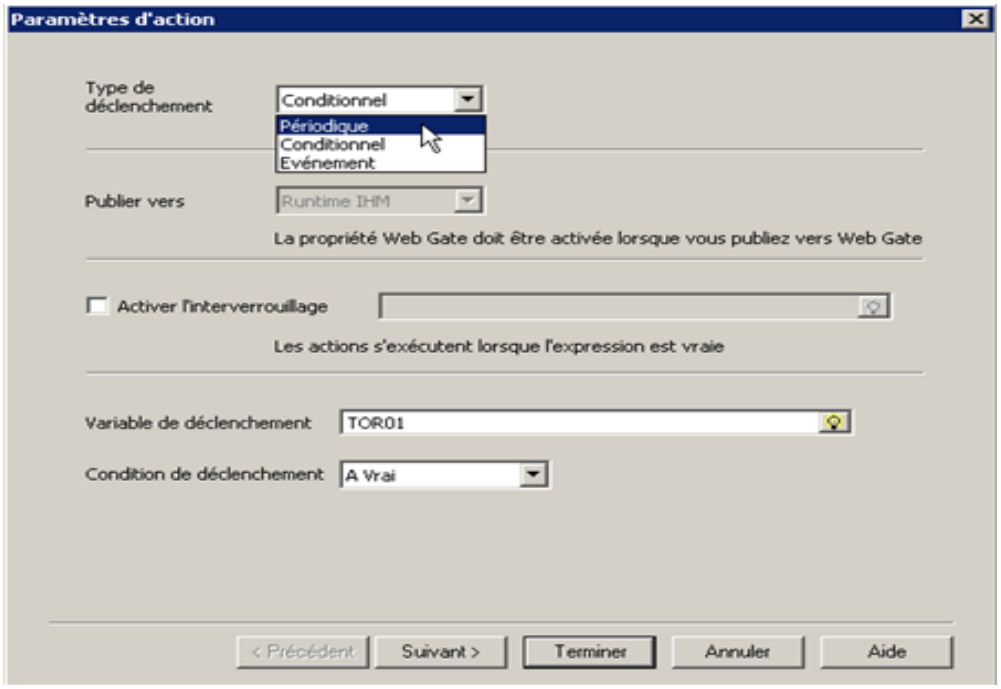
Dans Vijeo Designer, un script peut permettre d'automatiser une partie des tâches d'un programme de communication. Vous pouvez utiliser un script pour définir une procédure qui s'exécute lorsqu'une condition est remplie.

V.11. 1 Structure du script

Les scripts sont composés de deux parties : le déclenchement et les instructions codées. [11]

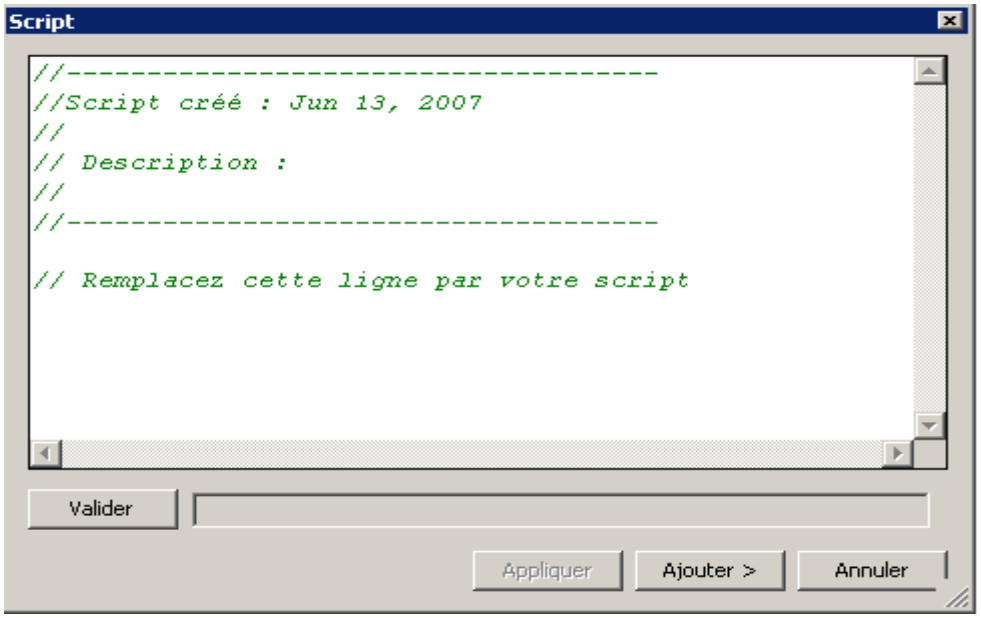
- **Déclenchement**

Le déclenchement définit le moment où le script est exécuté. Les déclenchements varient en fonction du type de script.il peut être périodique, conditionnel ou évènement.



- **Instructions**

Les instructions font référence au script réel, tel qu'il s'affiche dans l'Editeur de scripts.



V.11. 2 Programmation avec des scripts

Les scripts Vijeo-Designer sont basés sur le langage de programmation Java.

Un script contient des instructions écrites par des utilisateurs expérimentés afin de programmer le type de réaction de la machine cible face aux événements en temps réel, tels que : une pression sur un appui, une modification d'écran ou un changement de valeur. La principale raison de l'utilisation de scripts

Vijeo-Designer s'explique par le fait que la programmation n'est pas possible à l'aide des objets ou d'une animation. Une autre raison justifiant le recours aux scripts est que vous pouvez utiliser des variables de source externe dans les opérations de script afin d'ajouter une dimension de programmation susceptible de manquer dans le programme de l'équipement.

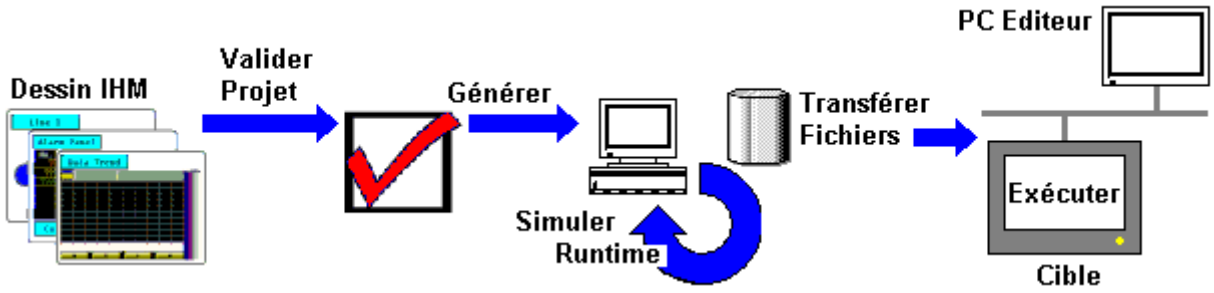
Les instructions de contrôle

Mot clé	Description
break	Quitte la boucle actuelle ou l'instruction de cas.
do-while	Répète un ensemble d'instructions tant que l'expression conditionnelle est true. Les instructions sont au moins exécutées une fois.
for	Répète un ensemble d'instructions tant que l'expression conditionnelle est true. Sténo pour une boucle, pour laquelle les variables sont initialisées avant que la boucle et les variables soient mises à jour à chaque itération de la boucle.
if-else	Contrôle le flux d'une exécution à l'aide d'une condition.
return	Quitte le script.
switch-case-default	Exécute un ensemble d'instructions de cas qui correspondent à la condition d'interruption définie. Exécute des instructions par défaut lorsque la condition d'interruption ne correspond pas à l'une des instructions de cas.
while	Répète un ensemble d'instructions tant que l'expression conditionnelle est true

V.12 Transfert et vérification de projets lors du runtime

Après avoir développé l'IHM, vous pouvez transférer le projet vers la machine cible et exécuter l'application utilisateur. Cependant, vous devez vous assurer que le projet ne contient pas d'erreurs avant de commencer son transfert.

le schéma suivant illustre la procédure type à suivre une fois l'IHM créé.



Une fois le projet IHM terminé, recherchez les erreurs en validant le projet. Lorsque toutes les erreurs évidentes ont été corrigées, l'étape suivante consiste à générer le projet et à simuler les opérations dans Runtime. L'étape de génération permet de créer les fichiers utilisés par la machine cible. Une fois que la génération réussie du projet est effectuée, vous pouvez transférer les fichiers vers la machine cible et exécuter le projet.

Vous devrez probablement exécuter plusieurs fois les processus de validation, génération, simulation et transfert avant de régler toutes les erreurs pouvant résulter du processus de développement du projet. Optimisez votre temps en corrigeant le projet dès le début du développement afin d'anticiper et de résoudre rapidement les erreurs. [11]

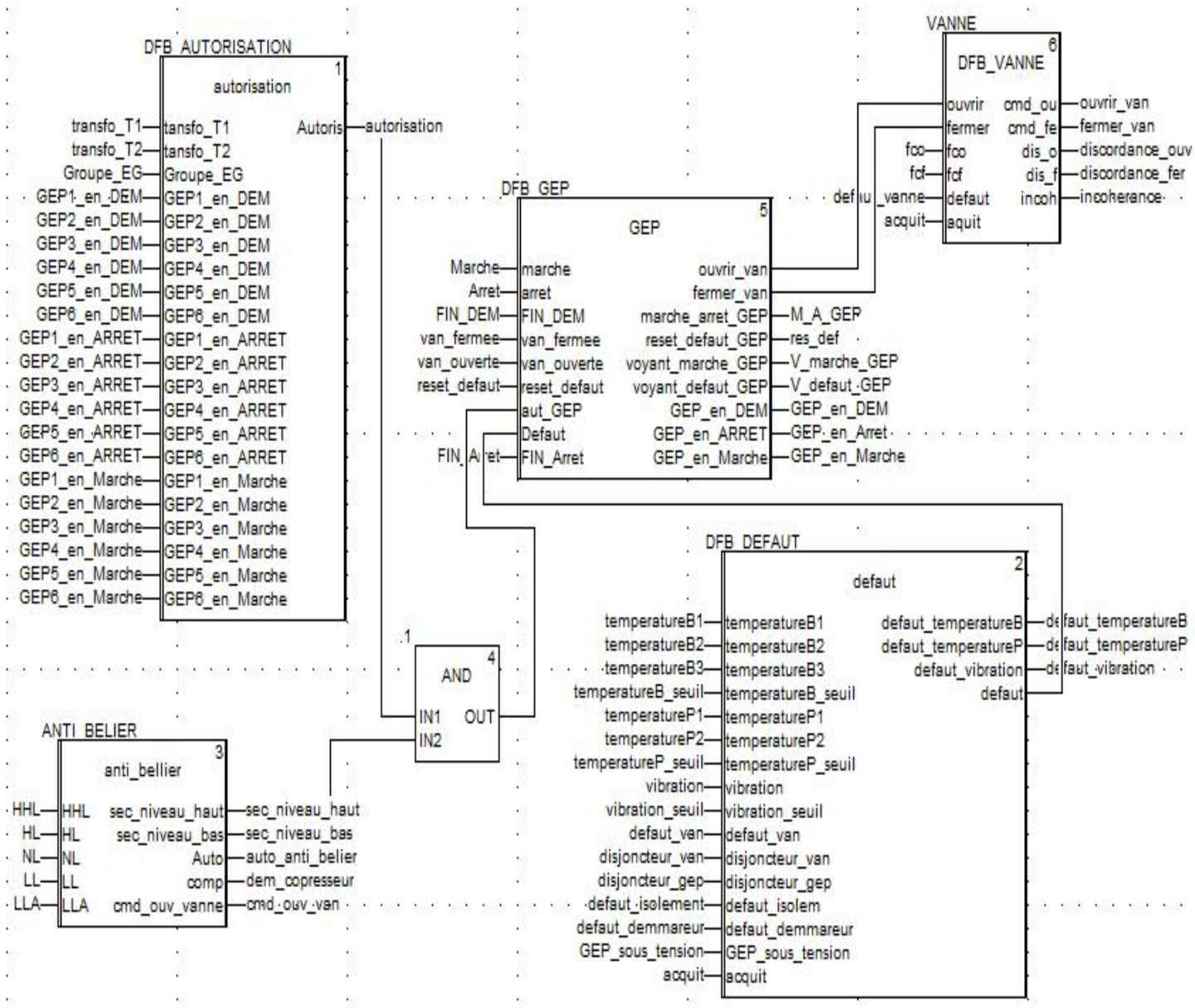
Note : Recherche des erreurs avant un transfert et Génération et Transfert du projet est décrit dans l'annexe C .

V.13 Programme de supervision de la station

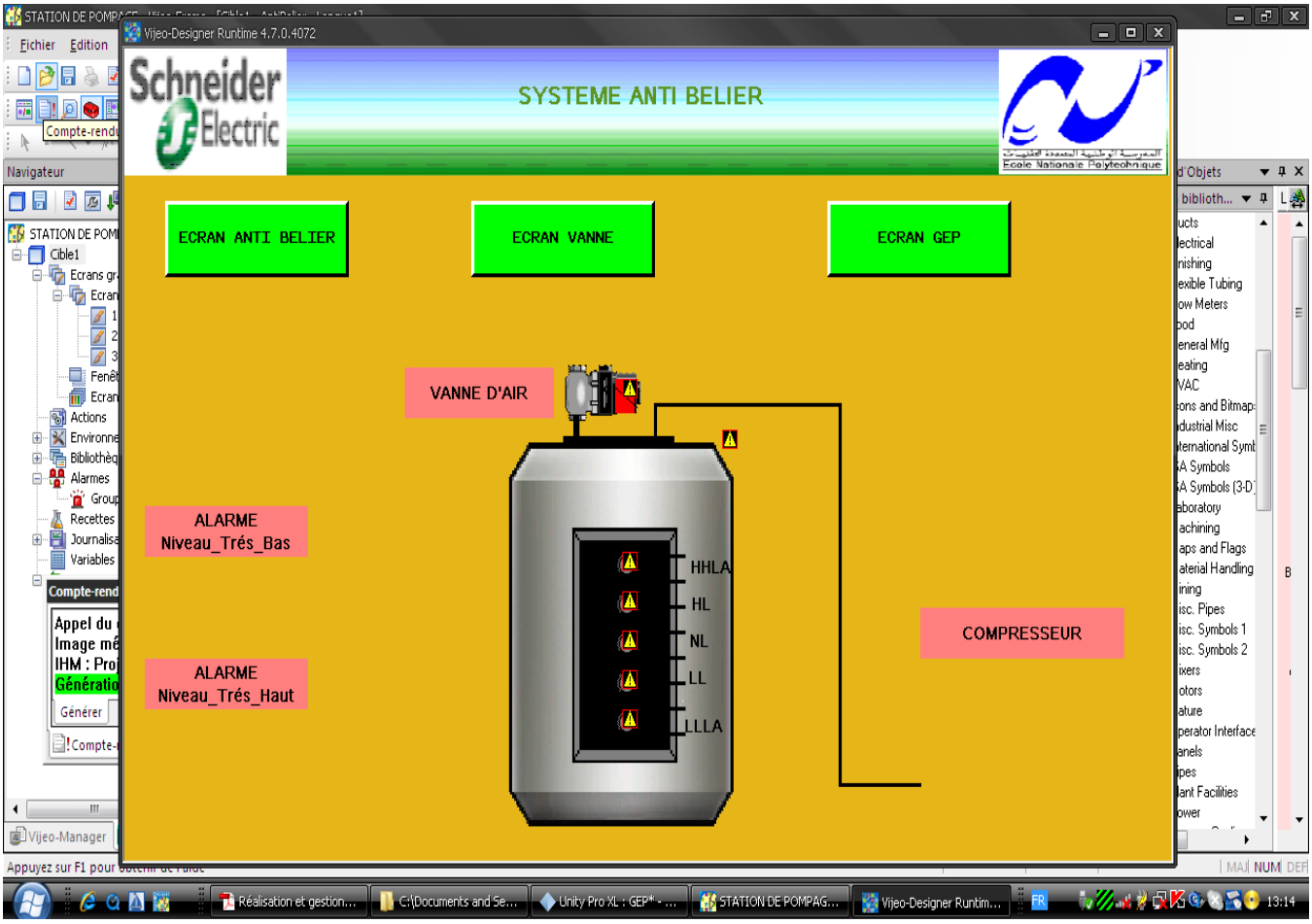
Le Programme Unity pro final

Avant d'écrire le programme de supervision, il fallait écrire le programme final de la station , selui qui prend en charge tous le DFB mentionnés dans le chapitre IV, le programme final comme il est déjà mentionné auparavant (chapitre IV) est programmé à l'aide du langage FBD , les variable de ce système vont être relier avec le programme de supervision Vijeo Designer , et vont être exécuté en même pour pouvoir visualiser le résultat de simulation .

La figure suivante représente le programme final de la station. .

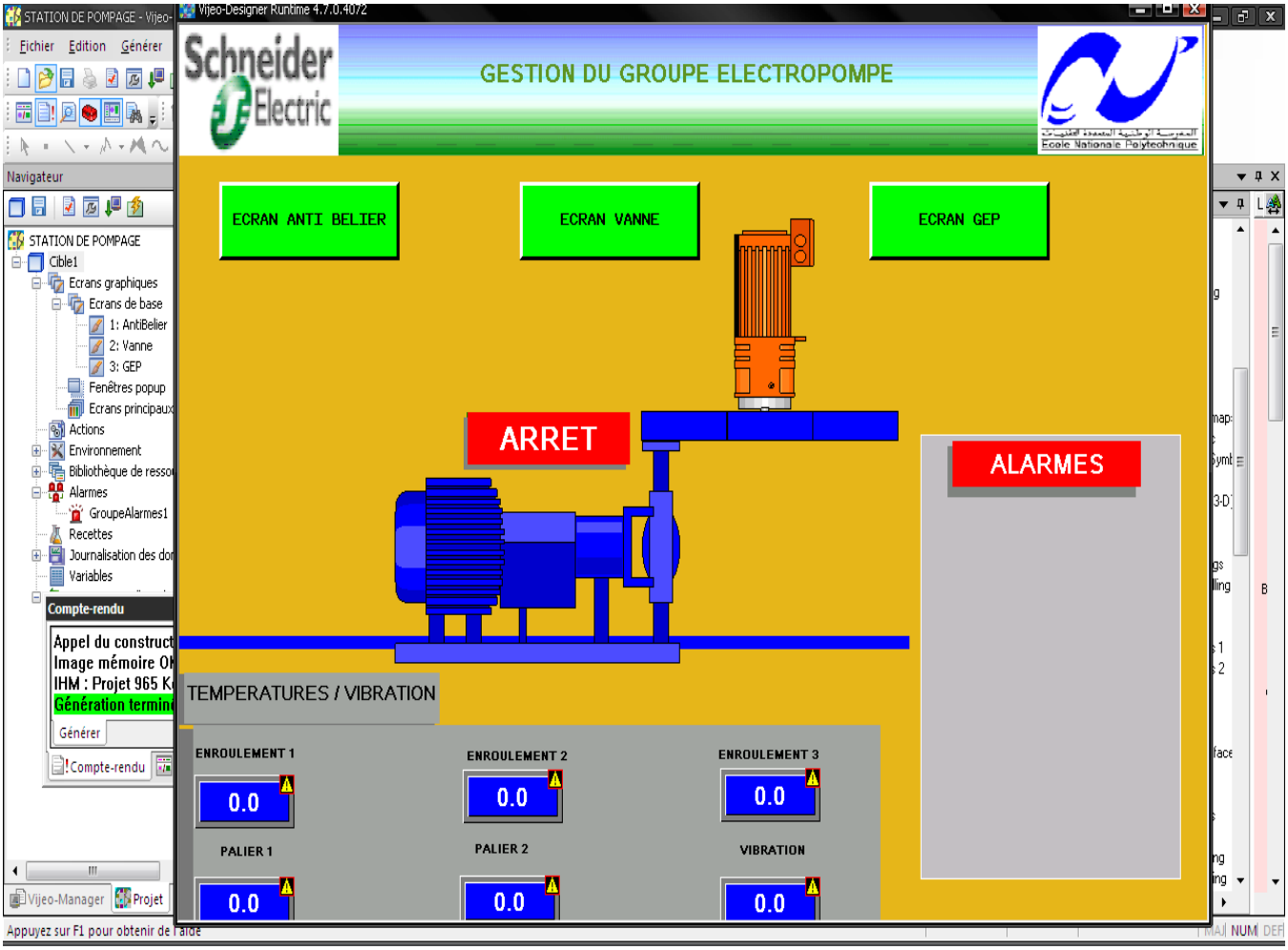


Les prochaines figures représentent respectivement les écrans de base de : l'anti bélier , la vanne et le GEP respectivement .


Vijeo-Designer Runtime 4.7.0.4072
SYSTEME ANTI BELIER
ECRAN ANTI BELIER
ECRAN VANNE
ECRAN GEP
VANNE D'AIR
ALARME Niveau_Trés_Bas
ALARME Niveau_Trés_Haut
HHLA
HL
NL
LL
LLLA
COMPRESSEUR

==========


GESTION DE LA VANNE
ECRAN ANTI BELIER
ECRAN VANNE
ECRAN GEP
ALARME INCOHERENCE
ALARME DISCORDANCE OUVERTURE
ALARME DISCORDANCE FERMETURE
FIN D'OUVERTURE
OUVRIR
FIN DE FERMETURE
FERMER



V.14 Conclusion

Schneider Electric propose un grand choix en terme de pupitre de supervision et de contrôle. La description précédente des de l'outil Vijeo Designer aide l'utilisateur (programmeur de L'IHM ou pupitre) à se familiariser, à savoir les performances mais aussi à programmer des interfaces IHM facilitant la communication entre de l'homme avec sa machine.

Conclusion Générale

La fonction dûun ingénieur doit être lôoptimisation de technique permettant dôaboutir à lôobjectif avec simplicité et performance. Notre contribution sôest portée sur la réalisation des BFB permettent de structurer et dôoptimiser le programme de gestion de la station de pompage ces DFB sont présenté dûune façon simple et compréhensible .Les résultats et tests obtenus répondent bien aux cahiers des charges.

Lôutilisation des interfaces hommes dans notre programme pour la visualisation de lôétat du process pendant lôexécution de la simulation et par la suite les intégrer réellement dans la pratique nous a monter la nécessité dôavoir un aspect artistique pouvant faire la différence dans lôétape de commercialisation du projet .

Dans la vie dôingénieur la prise en conte du coté sécurité est indispensable, en effet : avant de penser au performance des DFB du programme de gestion de la station de pompage il fallait prendre en considération la sécurité des équipement et aussi des programmes.

exemples :

- démarrage à vide et fermeture des vannes avant lôarrêt progressive des pompes, utilisation des anti béliers (sécurité des équipement).
- L ce temps dôouverture de la vanne serai une variable locale et non pas dôentrée pour le bloc DFB vanne (sécurité par rapport à la programmation).

Ce projet nous a poussé à faire appel à plusieurs connaissances et aptitudes dôélèves ingénieurs que se soit dans la mécanique des fluides , lôélectrotechnique ou lôinformatique industrielle et nous a permis dôapprécier vraiment la nécessité dôavoir un aspect de communication avec dôautres spécialistes dans le domaine de notre application ,une telle communication permettra à lôautomaticien de bien comprendre son projet et par la suite pouvoir amélioré ces performances en fessant appel à ces propres connaissances.

Perspectives

A la lumière des résultats obtenus pour ce projet de fin dôétude , de nombreuses perspectives nous sôouvrent :

- Mettre en î uvre des commandes au niveau de lôautomate pour la régulation du débit par un PID dont on déterminera les paramètres par des courbes de tendances .
- Automatisation des taches de démarrage des groupes électropompes et cela par rapport au niveau dôeau dans le château qui alimente la ville en eau ,et cela en insérant des capteurs de niveau pouvant communiquer directement avec lôautomate à travers des liaisons radios (Schneider Electric possède des modules spéciaux pour ça).
- Réécrire les DFB avec le langage de programmation C afin de réduire leurs volumes et accélérer leurs exécution ,il sôagit de créer des blocs EFB et les intégrer carrément dans les nouvelles versions de lôUnity Pro.

ملخص

العمل المنجز في هذه المذكرة يتمحور أساسا حول استخدام برمجة **شنيدر** لإدارة محطة لضخ المياه ودلك باستخدام برنامج "**اينيتي برو**" .تصميم شاشة إشراف من خلالها نستطيع مراقبة جميع أجهزة المحطة باستخدام برنامج **فيجيو ديزاينر**.
محطة الضخ المدروسة تقع في بلدية القبة بالجزائر.نوع آلي البرمجة المستخدم في المحطة هو M340.

الكلمات الرئيسية

مسيرصناعي مبرمج شنيدر, محطة لضخ المياه,برنامج اينيتي برو, برنامج **فيجيو ديزاينر**, آلي البرمجة M340

RESUME :

Le travail présent dans ce mémoire est base essentiellement sur l'utilisation des automates programmables SHNEIDER. Notre travail est La programmation de la gestion dûne station de pompage hydraulique à lôaide de logiciel Unity Pro par lôutilisation des DFB et la supervision à lôaide de logiciel Vijeo Designer. La station de pompage étudiées est situées à KOUBA _Alger . Sa gestion est assures par un automate M340.

Mots clés : automates programmables SHNEIDER , la gestion dûne station de pompage , logiciel Unity Pro , DFB , logiciel Vijeo Designer , automate M340 .

ABSTRACT:

The work presented in this memory is based primarily on the use of the programmable automats SCHNEIDER. Our job is programming for the management of à water pumping station with Unity Pro software through the use of DFB and supervision with Vijeo software designer.
The pumping station is located studied KOUBA _Alger. Its management is provided by a PLC M340

Key word : Programmable logic controller SCHNEIDER, water pumping station ,Unity Pro software ,DFB , Vijeo software designer, PLC M340

Bibliographie

[1] www.automelect.inf

[2] La définition est donnée par la norme NFC 63-850

[3] R.BOUROUBI, «Réalisation et gestion dôun prototype de station de pompage à base dôautomates programmables industriels SIEMENS », projet de fin dôétude, Ecole Nationale Polytechnique 2007.

[4] G. MICHEL, « Les A.P.I Architecture et application des automates programmables industriels », Edition DUNOD, 1987

[5] http://www.constructionlinks.ca/images/featured_products

[6] télé catalogue Schneider Electric.

[7] Modicon M340 sur Unity Pro (help on ligne).

[8] Help du logiciel Unity Pro .

[9] photo prise du logiciel Unity Pro.

[10] formation Unity Pro (document Schneider Electric).

[11] help Vijeo designer

ANNEXE A

Description des principaux outils du logiciel de programmation Unity Pro

---

A.8 Description des données

A.8.1 Présentation générale des données

A.8.2 Types de données

A.8.3 Les instances de données

A.8.4 Références de données

A.9 Editeur de données

A.10 Convertisseur PL7

A.10.1 Définition

A.10.2 Principe de conversion

A.10.2.a Conversion automatique

A.10.2.b Conversion semi-automatique

A.10.3 Différences entre PL7 et Unity Pro

A.10.3.1 Différences entre les différentes structures d'application

A.10.3.1.a les éléments structurels

A.10.3.1.b les modules fonctionnels

A.10.3.2 Différences entre PL7 et Unity Pro : types et tables

A.10.3.2.a Types

A.10.3.2.b Tableaux

Annexe A .

A.1 Les propriétés principales des versions d'Unity Pro

Le tableau suivant présente les propriétés principales des 5 versions d'Unity Pro : [8]

	Unity Pro S	**Unity Pro M**	**Unity Pro L**	**Unity Pro XL**
Langages de programmation				
Langage à blocs fonction (FBD)	+	+	+	+
Langage à contacts (LD)	+	+	+	+
Liste d'instructions IL	+	+	+	+
Littéral structuré ST	+	+	+	+
Diagramme fonctionnel en séquence SFC	+	+	+	+
Bibliothèques				
Bibliothèque standard	+	+	+	+
Bibliothèque de régulation	+	+	+	+
Bibliothèque de communication	+ (1)	+	+	+
Bibliothèque de diagnostics	+ (1)	+	+	+
Bibliothèque de gestion des E/S	+ (1)	+	+	+
Bibliothèque système	+ (1)	+	+	+
Bibliothèque de commande d'entraînement	-	+	+	+
Bibliothèque TCP Open	-	En option	En option	En option
Bibliothèque obsolète	+ (1)	+	+	+
Bibliothèque MFB	+	+	+	+
Bibliothèque de gestion des fichiers de carte mémoire	+	+	+	+
Informations générales				
Création et utilisation des structures de données (DDT)	+	+	+	+
Création et utilisation des blocs fonction dérivés	+	+	+	+
Navigateur de projet avec vue structurelle et/ou fonctionnelle	+	+	+	+
Gestion des droits d'accès	+	+	+	+

	Unity Pro S	**Unity Pro M**	**Unity Pro L**	**Unity Pro XL**
Ecrans d'exploitation	+	+	+	+
Visualisation du diagnostic	+	+	+	+
Diagnostic système	+	+	+	+
Diagnostic du projet	+	+	+	+
Convertisseur d'applications	-	Convertisseur PL7	Convertisseur PL7 Convertisseur Concept	Convertisseur PL7 Convertisseur Concept
Gestion de plusieurs stations	-	-	-	-
Plates-formes prises en charge				

Modicon M340	BMX P34 1000 BMX P34 20ÅÅ	BMX P34 1000 BMX P34 20ÅÅ	BMX P34 1000 BMX P34 20ÅÅ	BMX P34 1000 BMX P34 20ÅÅ
Premium	-	P57 0244M P57 CA 0244M P57 CD 0244M P57 104M P57 154M P57 1634M P57	Toutes les UC **sauf** : P57 554M P57 5634M P57 6634M	Toutes les UC
Quantum	-	-	140 CPU 311 10 140 CPU 434 12 U/A* 140 CPU 534 14 U/A*	Toutes les UC
Atrium	-	PCI 57 204	Toutes les UC	Toutes les UC
Simulateur	+	+	+	+
Transparence				
Liens hypertexte	+	+	+	+
Serveur Unity Pro (pour OFS, UDE, UAG)	-	-	-	+

Légende : + = disponible ; + (1) = disponible en partie : EF et fonctions des plates-formes Premium/Atrium et Quantum non inclus. - = non disponible

Tableau A.1 : *propriétés principales des différents progiciels* [8]

A.2 Navigateur de projet

A.2.1 Définition

Le navigateur de projet affiche tous les paramètres du projet. L'affichage peut se présenter sous forme structurelle (topologique) et/ou fonctionnelle. [8]

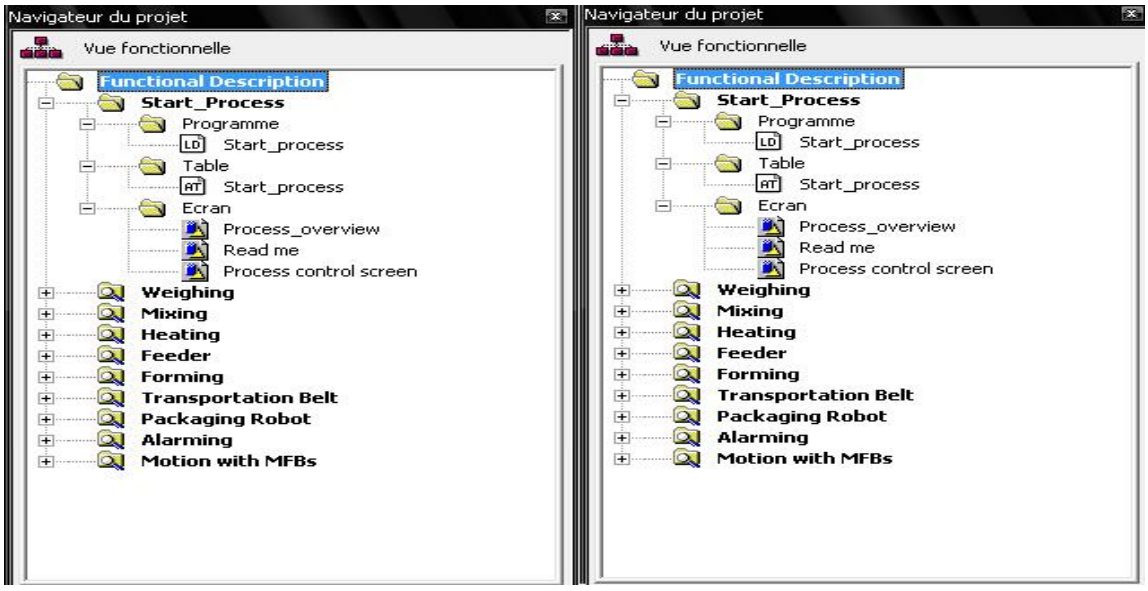


Figure A.1 : *vue structurelle et vue fonctionnelle* [9]

Vue structurelle

Dans l'affichage structurel, le navigateur de projet propose entre autres les fonctions suivantes:

- création et suppression d'éléments
- Le symbole de section affiche le langage de programmation de section et sa programmation éventuelle (dans le cas d'une section vide, le symbole est grisé)
- affichage des propriétés des éléments
- création de répertoires utilisateur
- démarrage des différents éditeurs
- démarrage de la fonction import/export

Vue fonctionnelle

Dans l'affichage fonctionnel, le navigateur de projet propose entre autres les fonctions suivantes :

- création de modules fonctionnels
- insertion de sections, tables d'animation, etc. par glisser-lâcher à partir de l'affichage structurel
- création de sections
- affichage des propriétés des éléments
- démarrage des différents éditeurs
- symbole de section indiquant le langage de programmation et d'autres attributs

A.2.2 les différents répertoires du navigateur du projet

A.2.2.a Le répertoire Projet

La vue structurelle vous permet d'accéder à la structure du projet et aux services associés.

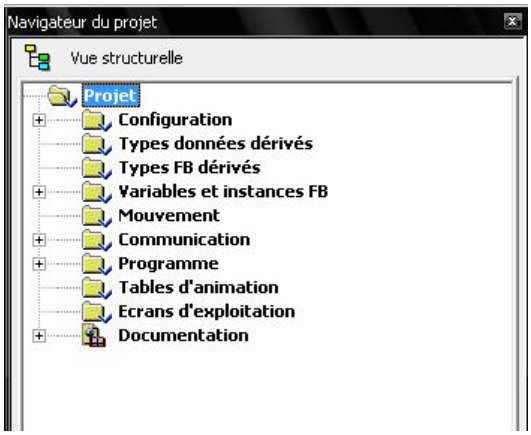


Figure A.2: *le répertoire projet* [9]

Services associés :

Le répertoire Projet vous donne accès aux services suivants, accessibles par le menu contextuel: [8]

Répertoire	Services
Projet	**Exporter le projet**: donne accès à lœexport du projet global **Propriétés**: donne accès aux propriétés du projet global.
Configuration	donne accès à la configuration matérielle et au paramétrage des modules.
Types données dérivés	donne accès aux types de DDT.
Types FB dérivés	donne accès aux types de DFB
Variables et instances FB	Permet d'accéder aux variables et aux instances de bloc fonction.
Mouvement	Permet d'accéder à la déclaration et la configuration des variateurs.
Communication	Permet d'accéder à la configuration des réseaux.
Programme	Permet d'accéder au programme du projet.
Tables dœanimation	Permet d'accéder aux tables d'animation.
Ecrans dœexploitation	Permet d'accéder aux écrans d'exploitation.
Documentation	Permet d'accéder à la documentation.

Tableau A.2 : *répertoire projet* [8]

A.2.2.b Le répertoire Configuration

Le répertoire Configuration de la vue structurelle du projet vous permet d'accéder à la configuration matérielle et au paramétrage des modules suivants : bus, rack, module.

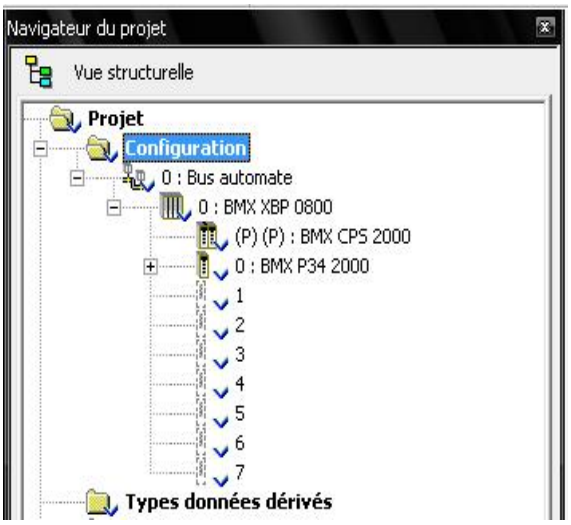


Figure A.3 : *exemple d'arborescence du répertoire Configuration* [9]

A partir du répertoire Configuration vous pouvez : [8]

- Configurer le rack.
- Configurer les équipements du bus de terrain.
- Accéder à la configuration des éléments du rack :
- les processeurs
- les modules E /S.

A.2.2.c Répertoire Types données dérivés (DDT)

Le répertoire Types données dérivés de la vue structurelle du projet vous permet d'accéder aux types de DDT.

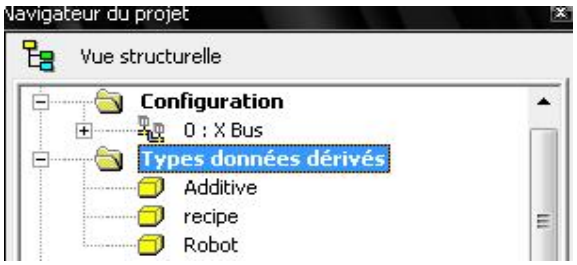


Figure A.4 : *l'arborescence du répertoire Types données dérivés* [9]

Services associés

Le répertoire Types données dérivés vous donne accès aux services suivants, accessibles par le menu contextuel:

Ouvrir: donne accès à lôonglet DDT types de lœéditeur de données à partir du quel vous pouvez :

- créer un type de données DDT,
- gérer un type de données DDT.

Obtenir de la bibliothèque: donne accès à la lecture dôun ou de plusieurs DDT depuis une bibliothèque.

Placer dans la bibliothèque: permet archiver tous les DDT dans une bibliothèque.

Exporter: donne accès à lœexport de tous les DDT.

Importer: donne accès à lôimport dôun ou de plusieurs DTT.

A.2.2.d Répertoire Types FB dérivés (DFB)

Le répertoire Types FB dérivés de la vue structurelle du projet vous permet d'accéder aux types de DFB. [8]

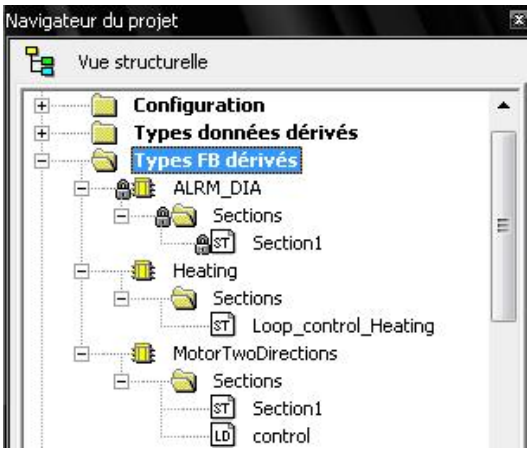


Figure A.5: *exemple dôarborescence du répertoire Types FB dérivés* [9]

Le répertoire Types FB dérivés vous donne accès aux services suivants, accessibles par le menu contextuel :

Open: donne accès à lôonglet DFB de lœéditeur de données.

Obtenir de la bibliothèque: donne accès à la lecture dôun ou de plusieurs types de DFB depuis une bibliothèque.

Placer dans la bibliothèque: donne accès à lœécriture de tous les types de DFB dans une bibliothèque. [8]

Exporter: donne accès à lœexport de tous les types de DFB du projet.

Importer: donne accès à lœimport dœun ou de plusieurs types de DFB.

A.2.2.e Répertoire Variables

Le répertoire Variables et instances FB de la vue structurelle du projet vous permet d'accéder aux variables (EDT, DDT, IODDT) et aux instances de blocs fonction (EFB, DFB).

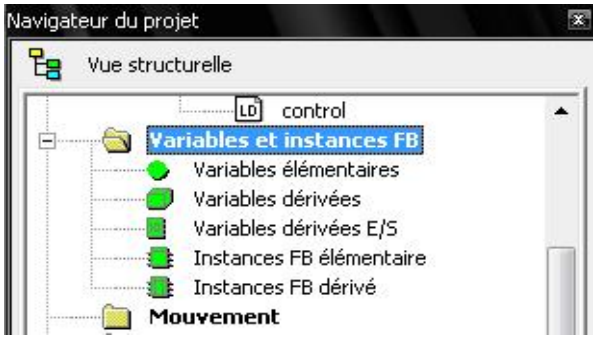


Figure A.6 : *lœarborescence du répertoire Variables et instances FB* [9]

Le répertoire Variables et instances FB vous donne accès aux services suivants, accessibles par le menu contextuel:

Variables et instances FB

Ouvrir: donne accès à lœéditeur de variables,

Exporter: permet d'accéder à l'exportation de toutes les variables du projet,

Importer: permet d'accéder à l'importation de toutes les variables du projet.

variables élémentaires, variables dérivées, Variables dérivées dœE/S, Instances FB élémentaire, Instances FB dérivé

Ouvrir: donne accès à lœonglet correspondant de lœéditeur de variables,

Exporter: permet d'accéder à l'exportation de toutes les variables de la famille sélectionnée (EDT, DFB, etc.). [8]

A.2.2.f Répertoire Mouvement

Le répertoire Mouvement de la vue structurelle du projet vous permet dôaccéder à la déclaration et la configuration des variateurs. Ce service est disponible sur les plates-formes Premium et Modicon M340. Lors de la déclaration d'un variateur, plusieurs informations sont demandées telles que:

- le nom donné au variateur,
- le type de variateur,
- l'adresse CANopen du variateur,
- la référence du variateur,
- la version du variateur,
- le nom des variables associées à l'axe. [8]

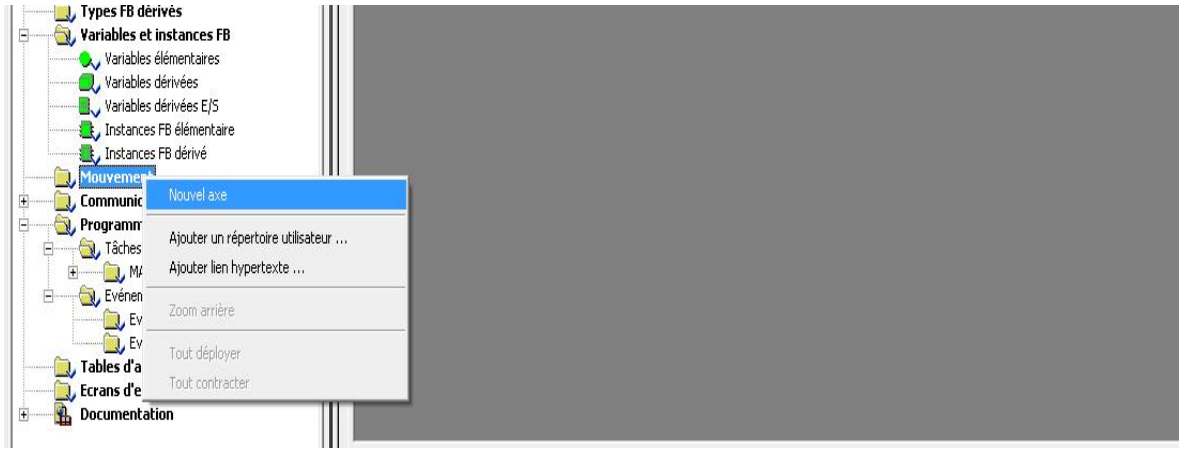


Figure A.7 *Le répertoire Mouvement* [9]

La figure suivante représente un exemple dôarborescence du répertoire Mouvement :

Dans cette figure, le nom donné au variateur est óAxe_Zô

Une recette est associée par défaut à chaque création dôaxe. Il est possible de créer plusieurs recettes.

Figure A.8: *répertoire Mouvement(vue structurelle)* [9]

Services accessibles :

Le répertoire Mouvement vous donne accès aux services suivants, accessibles par le menu contextuel :

Mouvement : **Nouvel axe :** permet de créer un axe.

Axe : **Nouvelle recette :** permet de créer une recette.

Propriétés : permet d'accéder aux propriétés de l'axe.

Recette : **Propriétés** : permet d'accéder aux propriétés de la recette.

A.2.2.g Répertoire Communication

Le répertoire Communication de la vue structurelle du projet vous permet d'accéder à la configuration des réseaux. [8]

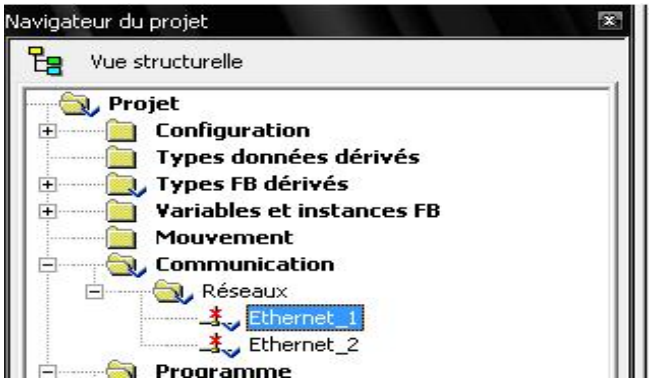
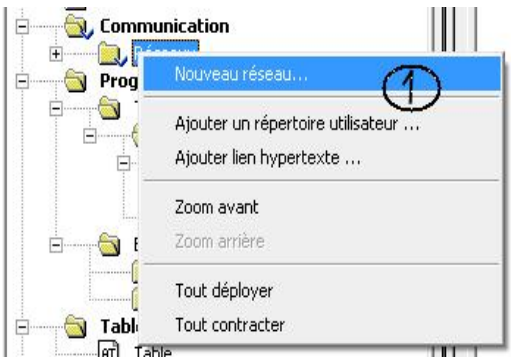


Figure A.9 : *d'arborescence du répertoire Communication* [9]

Création et configuration d'un réseau logique

(1) 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.



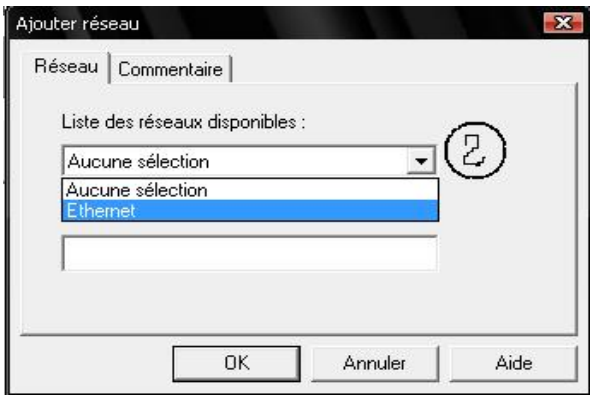


Figure A.10: *Création et configuration d'un réseau logique* [9]

(2) Sélectionnez le réseau à créer dans la liste des réseaux disponibles et donnez-lui un nom significatif.

Cliquez sur OK, un nouveau réseau logique est créé.

Résultat : Nous venons de créer le réseau Ethernet_1 qui apparaît dans le navigateur de projet

- Par un double clique sur le sous répertoire Ethernet_1Activer le réseau logique à configurer (Ethernet_1) la fenêtre suivant s'affiche

Figure A.11 : *Configuration du réseau Ethernet* [9]

- Configurer le réseau logique : global data, I/O scanning, ... (2)

A.2.2.h Répertoire Programme

Le répertoire Programme de la vue structurelle du projet vous permet de définir la structure du programme et d'accéder aux éditeurs de langage des éléments de programme : sections, modules de programme et traitements événementiels.

Editeur de programme

Un programme peut se composer de :

- ➢ tâches exécutées de manière cyclique ou périodique.

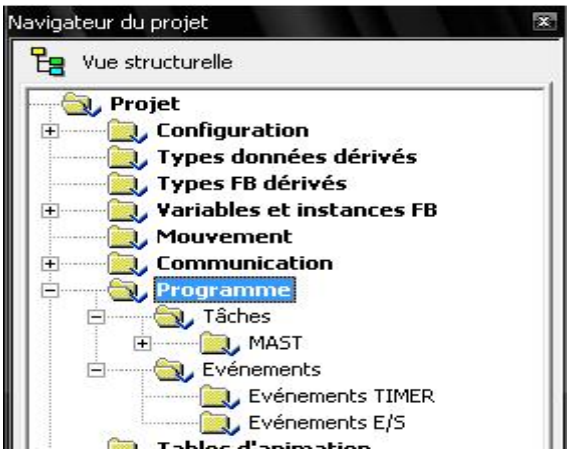


Figure A.12: *Arborescence du répertoire Programme* [9]

Les tâches sont elles-mêmes composées de :

- section
- sous-programmes
- ➢ traitements événementiels : exécutés prioritairement à toutes les autres tâches.

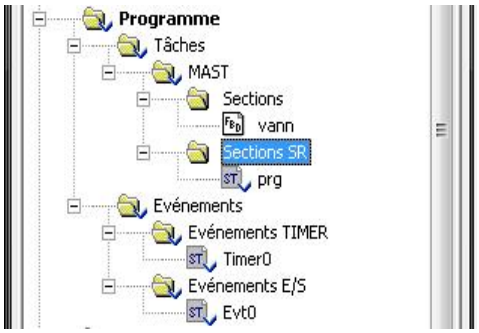


Figure A.13: *Arborescence du répertoire Programme (après création des sections)* [9]

Les traitements événementiels sont eux-mêmes composés de :

- sections de traitement d'événements à commande temporelle
- sections de traitement d'événements à commande matérielle.

Tâches :

Unity Pro gère des tâches multiples (multitâche).

Les tâches sont exécutées " en parallèle " indépendamment les unes des autres, avec les niveaux de priorité d'exécution commandés par l'API. Les tâches peuvent être adaptées à des exigences diverses et constituent ainsi un instrument puissant de structuration du projet.

Un projet multitâche peut se composer :

- ➢ d'une tâche maître (MAST)

L'exécution de la tâche maître est cyclique ou périodique.

Constituant la partie principale du programme, elle est exécutée de manière séquentielle.

- d'une tâche rapide (FAST)

L'exécution de la tâche rapide est périodique. Son niveau de priorité est plus élevé que celui de la tâche maître. La tâche rapide est destinée aux traitements de courte durée et périodiques.

- 1 à 4 tâches auxiliaires (AUX)

L'exécution des tâches auxiliaires est périodique. Elles sont destinées aux traitements plus lents, ce sont les tâches les moins prioritaires. [8]

Le projet peut aussi se composer d'une seule tâche. Dans ce cas, seule la tâche maître est active.

Sections :

Les sections sont des unités de programme autonomes dans lesquelles est créée la logique du projet. Les sections sont exécutées dans leur ordre de représentation dans le navigateur de projet (vue structurelle). Elles sont reliées à une tâche. Une même section ne peut pas appartenir à plusieurs tâches en même temps.

Sous-programmes :

Les sous-programmes sont créés en tant qu'unités distinctes dans des sections de sous-programme. Les appels aux sous-programmes s'effectuent à partir des sections ou d'un autre sous-programme. Une imbrication de huit niveaux maximum est possible. Un sous-programme ne peut pas s'appeler lui-même (non récursif). Les sous-programmes sont affectés à une tâche. Un même sous-programme ne peut pas être appelé par différentes tâches. On peut pas programmer les sous programme par le langage SFC

Traitement événementiel :

Le traitement événementiel s'effectue dans des sections dites d'événement. Ces sections d'événement sont exécutées en priorité sur les sections de toutes les autres tâches. Elles conviennent donc aux traitements demandant des délais de réaction très courts par rapport à l'arrivée de l'événement.

Les types de section disponibles pour le traitement événementiel sont les suivants :

- section de traitement d'événements à commande temporelle (section Timerx)
- section de traitement d'événements à commande matérielle (section Evtx)

A.2.2.i Répertoire Tables d'animation

Le répertoire Tables d'animation de la vue structurelle du projet permet d'accéder aux tables d'animation. [8]

Figure A.14 : *Arborescence du répertoire Tables d'animation* [9]

Le répertoire Tables d'animation permet de créer une table d'animation. A partir de cette table vous pouvez :

- ajouter des données.
- passer en mode modification.
- passer en mode forçage.

- **Ajouter des données**

Exécutez les actions suivantes :

1) Positionnez-vous sur une ligne vide

2) Soit :

- double-cliquez sur la ligne vide.
- Saisissez le nom de la variable ou sélectionnez-le dans la fenêtre Sélection d'instance à laquelle permet d'accéder le bouton ...

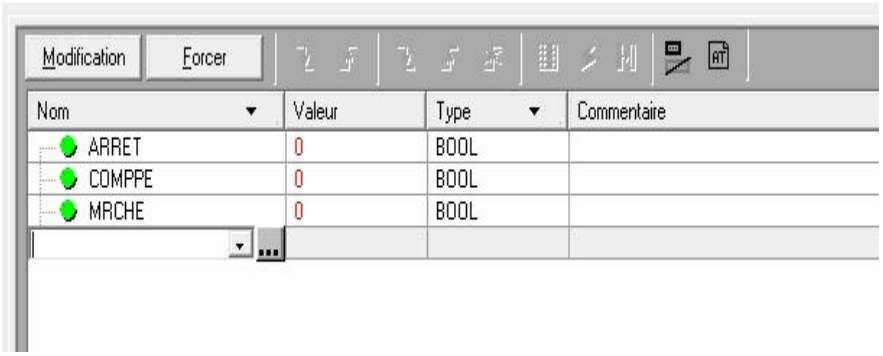


Figure A.15: *table dâanimation (ajout des données)* [9]

- **passer en mode modification**

On peut modifier la valeur du variable en Appuyant sur le bouton Modification puis on exécute les actions suivantes :

1) Cliquez deux fois dans la colonne Valeur sur la ligne correspondant à la variable à modifier.

2) Saisissez au clavier la valeur souhaitée.

3) Confirmez votre choix en appuyant sur la touche "Entrée".

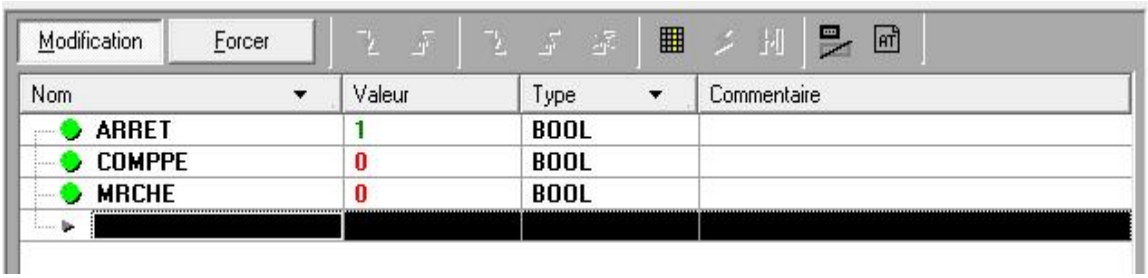


Figure A.16: *table dôanimation (mode modification)* [9]

- **passer en mode forçage**

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

- la variable doit être de type EBool.
- il doit s'agir d'une variable localisée.
- l'attribut de forçage doit être validé dans l'éditeur de variables.

Pour forcer une variable booléenne localisée, procédez comme suit :

1) A l'aide de la souris, sélectionnez la variable booléenne.

2) Cliquez sur le bouton  qui correspond à la valeur souhaitée ou, à partir du menu contextuel, sélectionnez la commande Forcer à 0 ou Forcer à 1..

A.2.2.j Répertoire écrans d'exploitation

Pour un projet donné, vous pouvez créer des écrans dôexploitation, en utilisant lôéditeur graphique.

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

Pour animer les objets dynamiques, vous devez leurs affecter une variable dont la valeur déterminera lôaffichage.

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

Les écrans peuvent être liés entre eux afin de répondre aux exigences spécifiques de lôautomatisme[8]

Le répertoire **Ecrans dôexploitation** vous permet de créer des écrans. A partir de ces écrans vous pouvez :

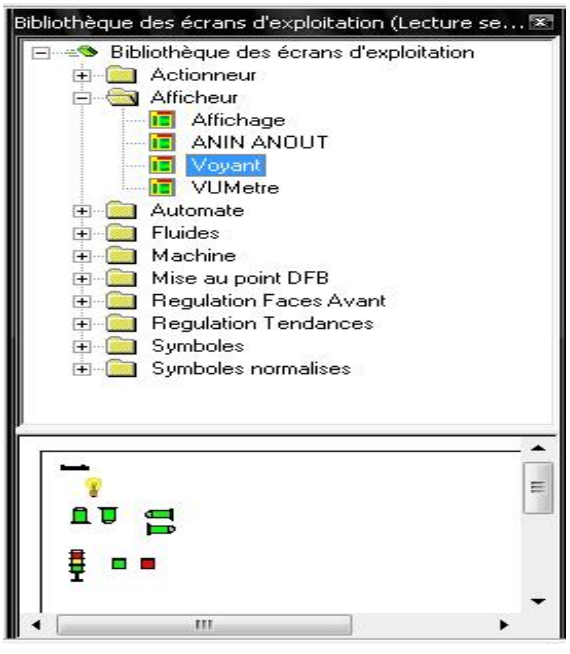
- créer des objets,
- insérer des objets à partir dôune bibliothèque,
- modifier les attributs des objets,
- manipuler les objets qui composent lôécran,
- utiliser les écrans en mode connecté

La bibliothèque dôobjets

La bibliothèque dôobjets présente les objets constructeurs et permet de les insérer dans les écrans dôexploitation. 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 sôouvre à partir de la commande Outils ® Bibliothèque des écrans dôexploitation.

La figure suivante présente la bibliothèque dôobjets.



A.2.2.k Répertoire Documentation

Unity Pro vous permet de réaliser la documentation de votre projet.

L'affichage des thèmes de la documentation se fait sous la forme d'une arborescence. La structure comprend tous les éléments utilisés du navigateur projet, c'est-à-dire que les éléments que vous n'avez pas utilisé dans le projet n'apparaissent pas non plus comme thèmes dans la documentation. De plus, la structure comprend quelques thèmes spécifiques à l'impression, comme Page de titre et Sommaire.

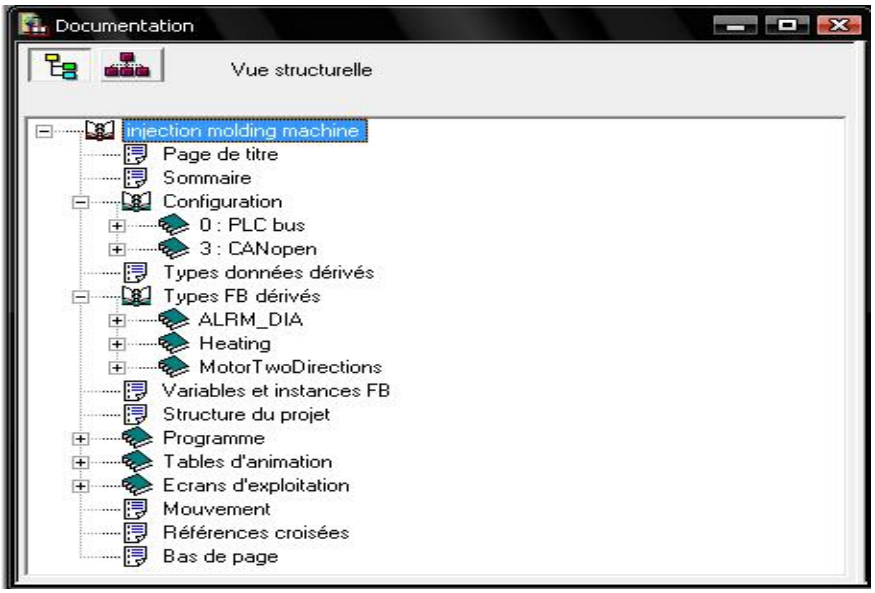
Dans l'arborescence, vous pouvez sélectionner les thèmes que vous voulez afficher (aperçu avant impression) ou imprimer et ceux que vous ne voulez ni afficher ni imprimer.

Pour certains thèmes, vous disposez de paramètres permettant de définir la manière dont ils seront imprimés.

Pour l'impression, vous pouvez sélectionner la Vue structurelle ou la Vue fonctionnelle.

Vous pourrez imprimer en partie ou entièrement ce dossier après l'avoir constituer.

La figure suivante représente la **structure de la documentation**



A.3Bibliothèques de blocs EF/EFB/DFB

A.3.1 Définition

Les blocs des nombreuses bibliothèques de blocs comprises dans l'offre de Unity Pro vont des blocs pour opérations booléennes simples aux blocs de commande de boucles de régulation complexes, en passant par des blocs pour chaînes de caractères (strings) et opérations de zones (matrice). [8]

Les blocs peuvent être utilisés dans les langages SFC, FBD, LD, IL et ST. Par souci de clarté, les différents blocs sont structurés en bibliothèques, elles-mêmes structurées en familles

Les différents types de bloc sont les suivants :

- Fonction élémentaire (EF)
- Bloc fonction élémentaire (EFB)
- Bloc fonction dérivé (DFB)
- Procédure

A.3.2 Les différentes bibliothèques Unity Pro

A.3.2.1 Bibliothèque standard

La Bibliothèque standard offre des fonctions et blocs fonction élémentaires devisées en plusieurs familles quôils sont les suivants :

- **Tableaux**
- **CLC_INT**
- **Comparaison**
- **Date et Heure**
- **Logique**
- **Mathématiques**
- **Statistique**
- **Chaînes de caractères**

- **Temporisateur et Compteur**
- **Conversion de type**

A.3.2.2 Bibliothèque de régulation

La Bibliothèque de régulation offre des fonctions et blocs fonction élémentaires devisées en plusieurs familles quôils sont les suivants :

- **Préparation des données**
- **Régulateur**
- **Mathématiques**
- **Traitement des mesures**
- **Traitement des valeurs de sortie**
- **Traitement des valeurs de consigne**

A.3.2.3 Bibliothèque de communication

Contient des fonctions et blocs fonction élémentaires sous la famille **Etendu**

A.3.2.4 Bibliothèque de gestion des E/S

La Bibliothèque de gestion des E/S offre des fonctions et blocs fonction élémentaires devisées en plusieurs familles quôils sont les suivants :

- **Configuration d'E/S analogiques**
- **Mise à l'échelle d'E/S analogiques**
- **Echange explicite**
- **E/S directes**
- **Configuration d'E/S Quantum**
- **Simulation**

A.3.2.5 Bibliothèque de mouvements

La Bibliothèque de mouvements offre des fonctions et blocs fonction élémentaires devisées en plusieurs familles quôils sont les suivants :

- **Commande d'axes**
- **Commande de cames**
- **Démarrage MMF**
- **Communication avec les variateurs Lexium**

A.3.2.6 Bibliothèque système

La Bibliothèque système offre des fonctions et blocs fonction élémentaires devisées en plusieurs familles quôils sont les suivants :

- **Gestion des fichiers**
- **Traitement d'événements**
- **Hot Stand By**
- **Gestion SFC**
- **Horloge système**
- **Particularités système**

A.3.2.7 Bibliothèque diagnostic

Contient les fonctions et blocs fonction de **diagnostic**

A.3.2.8 Bibliothèque obsolète

Les fonctions et blocs fonction de cette bibliothèque sont utilisés uniquement pour effectuer des conversions depuis les programmes utilisateurs Concept et PL7.

- **CLC**
- **CLC_PRO**
- **Extensions/compatibilité**

A.3.2.9 Bibliothèque TCP Open

TCP Open pour Premium propose un jeu de fonctions élémentaires (EF) et des blocs fonction dérivés (DFB) qui offrent des services TCP/IP via une application automatisée sur les automates Premium.

Les EF et les DBF TCP Open sont installés à partir d'un CD. Une fois installés dans Unity Pro, ils apparaissent dans la bibliothèque TCP Open, dans la famille Avancé.

Ces fonctions prédéfinies peuvent être utilisées dans les applications TCP/IP client/serveur sans que cela suppose une connaissance préalable des langages de programmation comme C ++ ou Java.

La programmation s'effectue directement à travers l'utilisation des EF et des DFB dans le langage automatisé souhaité (ST, LD, FBD ou IL).

TCP Open est disponible sur les modules suivants :

- TSX ETY 110WS
- TSX ETY 5103

A.3.2.10 Bibliothèque Blocs MFB (Motion Function Blocks)

Cette bibliothèque contient des Motion Function Block.

A.4 Simulateur automate

Le simulateur automate permet la recherche d'erreurs dans le projet sans connexion à un véritable automate. Toutes les tâches du projet (Mast, Fast, AUX et Evénements) se déroulant sur un véritable automate sont également disponibles dans le simulateur. La différence par rapport à un véritable automate réside dans l'absence de modules E/S et de réseaux de communication (p. ex. ETHWAY, Fipio et Modbus Plus) fonctionnant en temps réel non-déterministe. [10]

Naturellement, toutes les fonctions de mise au point, d'animation, les points d'arrêt, les forçages de variables, etc. sont disponibles sur le simulateur automate

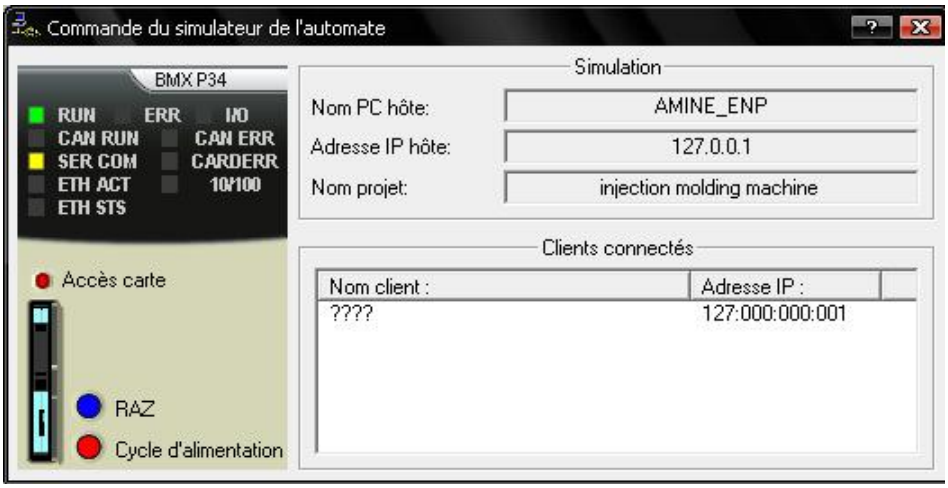


Figure A.17: *Représentation de la boîte de dialogue* [10]

Structure du simulateur :

Le panneau du simulateur propose les affichages suivants :

- type d'automates simulés ;
- état actuel des automates simulés ;
- nom du projet chargé ;
- adresse IP et nom DNS du PC hôte du simulateur et de tous les PC Clients connectés ;
- boîte de dialogue dédiée à la simulation des événements E/S ;
- bouton RAZ permettant de réinitialiser les automates simulés (simulation de démarrage à froid) ;
- bouton Mise sous/hors tension (pour simuler une reprise à chaud) ;
- menu contextuel (bouton droit de la souris) permettant de commander le simulateur.

A.5 Visualisation du diagnostic

Unity Pro dispose d'un diagnostic du système et des projets.

Dans le cas où des erreurs se produisent, celles-ci s'affichent dans une fenêtre de diagnostic. Pour corriger l'erreur, il est possible d'ouvrir la section à l'origine de l'erreur directement depuis la fenêtre de visualisation du diagnostic.

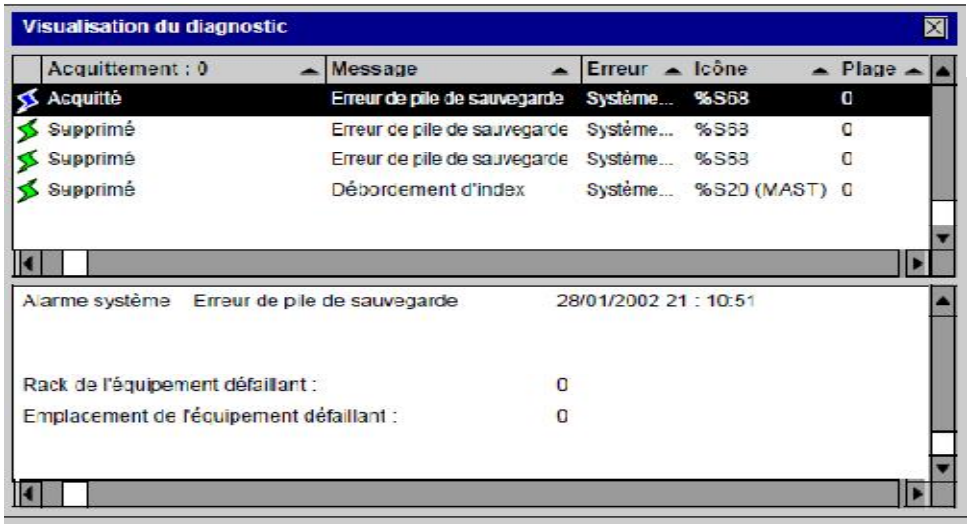


Figure A.18: *Visualisation du diagnostic* [10]

A.6 Configuration matérielle de BUS X

A.6.1 Manipulation des racks

A partir de lôéditeur de bus les fonctionnalités offertes sont les suivantes:

Si vous souhaitez	**alors**	**Et**
sélectionner un rack	sélectionnez lôadresse du rack huit poignets apparaissent autour du rack sélectionné	
faire un Couper\Coller dôun rack	sélectionnez le rack et par le menu contextuel la commande Couper	sélectionnez lôadresse destination du rack et par le menu contextuel la commande Coller.
ajouter un rack	sélectionnez une adresse vide ou le symbole extension >>	par le menu contextuel la commande Nouvel équipement,
remplacer un rack	sélectionnez le rack et par le menu contextuel la commande Remplacer le rack,	sélectionnez dans la liste proposée le rack souhaité.
supprimer un rack	sélectionnez le rack par le menu contextuel la commande	Supprimer le rack,
videz un rack	sélectionnez le rack par le menu	Effacer le rack,

	contextuel la commande	

Tableau A.3 : *Manipulation des racks* [8]

A.6.2 Manipulation des modules dôalimentation

Règles pour une station Modicon M340

Le module d'alimentation doit occuper la position la plus à gauche du rack. Cette position ne dispose pas d'adresse. Il n'y a qu'un seul module d'alimentation par rack.

Règles pour une station Premium/ Atrium

Le module d'alimentation doit occuper la position la plus à gauche du rack. Cette position ne dispose pas d'adresse. Un module dôalimentation double format occupe aussi la position dôadresse 0 (occupée habituellement par le module processeur), dans ce cas le module processeur doit être configuré à la position dôadresse 1. Il y a un seul module dôalimentation par rack.

Règles pour une station Quantum

Le module d'alimentation peut occuper n'importe quelle position du rack. Elle dispose d'une adresse.

Les modules dôalimentation sont simple format.

Plusieurs modules alimentation peuvent être configurés dans un rack.

Pour Modicon M340 et premium

Les fonctionnalités suivantes sont disponibles dans l'éditeur de bus :

Si vous souhaitez	**Alors**	**Et**
Sélectionner un module	cliquez dessus. Huit poignées apparaissent autour du module.	
Copier\Coller le module	sélectionnez le module, et par le menu contextuel la commande Copier	sélectionnez la position destination et par le menu contextuel la commande Coller.

Ajouter un module	sélectionnez la position dans le rack souhaité et par le menu contextuel la commande Nouvel équipement	sélectionnez dans la liste proposée le module souhaité.
Déplacer un module	sélectionnez le module	en maintenant lôappui sur la souris déplacez le à la position souhaitée.
Supprimer un module	sélectionnez le module	à l'aide du menu contextuel, cliquez sur Supprimer le module.

Tableau A.4 : *Manipulation des modules* [8]

Pour Quantum Le navigateur permet de placer ou d'ajouter un module d'alimentation dans la station.

Exemple de station Quantum contenant deux modules d'alimentation :

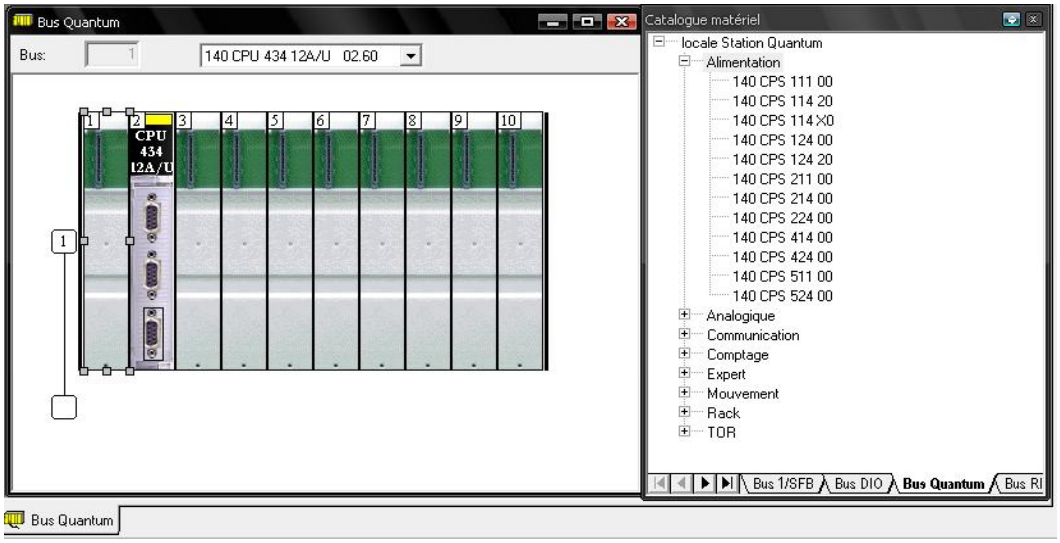


Figure A.19: *Ajouter dôun module d'alimentation pour Quantum* [9]

Exécutez les actions suivantes:

1- Sélectionnez dans le navigateur le répertoire Alimentation, et déployez le en cliquant sur +.

2- Sélectionnez le module alimentation voulu, et en maintenant lôappui sur la souris déplacez le à la position souhaitée.

A.6.3 Configuration des processeurs

Configuration du processeur Premium

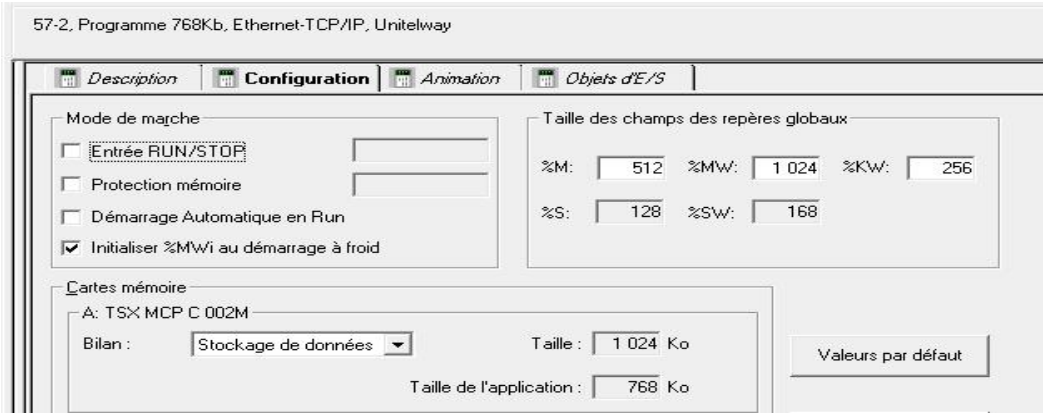


Figure A.20: *Ecran de configuration Premium* [9]

- Choisir modes de marche : entrée Run/Stop, protection mémoire, ...
- Définir les cartes mémoires.
- Définir les objets globaux de l'application : nombre de bits et de mots.

Configuration du processeur Quantum

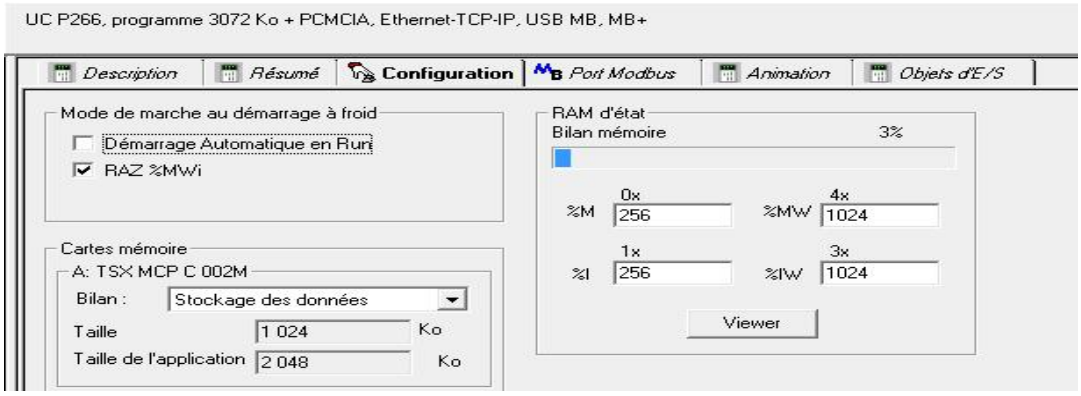


Figure A.21: *Ecran de configuration Quantum* [9]

- Choisir les modes de marche : démarrage en Run, Peer Cop, ...
- Définir les objets globaux de l'application : nombre de bits et de mots.

A.6.4 configuration des modules dôE/S

A.6.4.a module dôE/S pesage

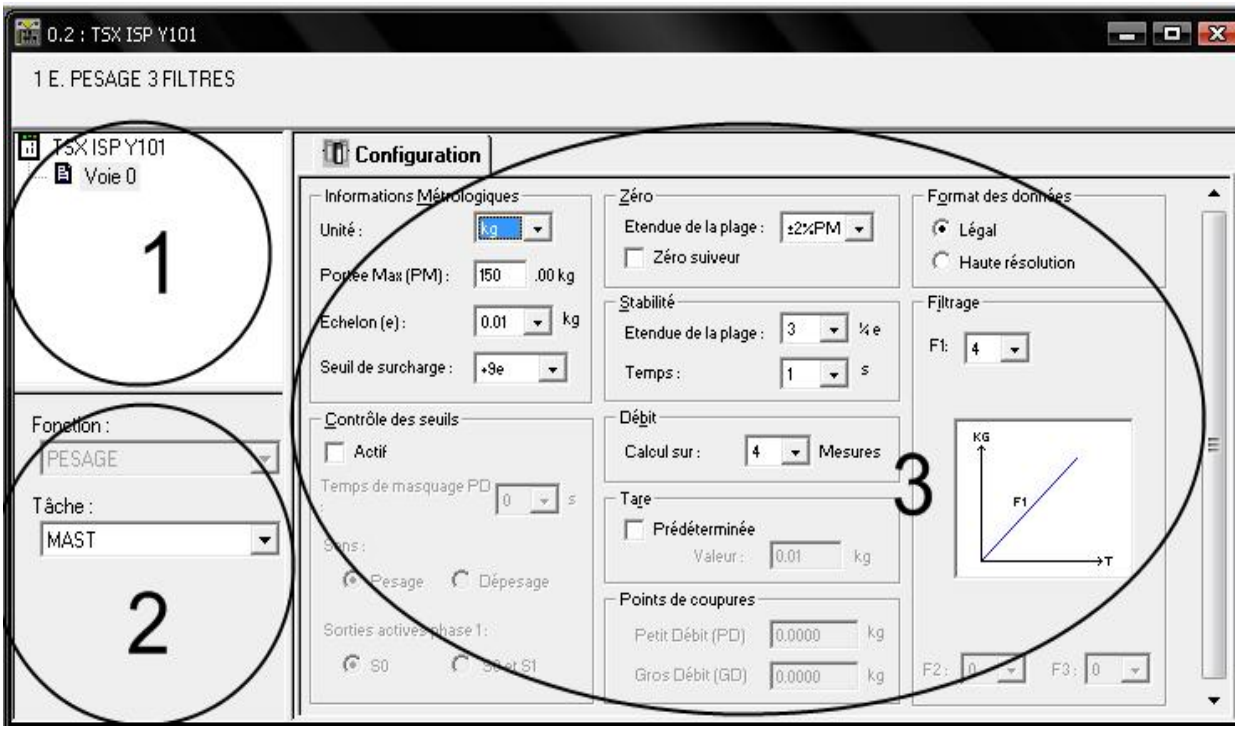


Figure A.22: *Le module pesage (l'écran de configuration.)*[9]

Adresse	Elément	Fonction
1	Champ **Voie**	Permet : ➢ de choisir la voie ; ➢ d'afficher le **Symbole**, nom de la voie défini par l'utilisateur (au travers de l'éditeur de variables).
2	Zone **Paramètres généraux**	Est composée des éléments suivants : le menu déroulant **Fonction** ; le menu déroulant **Tâche** permettant de définir la tâche (**MAST** ou **FAST**) dans laquelle seront échangés les objets à échanges implicites des voies.
3	Zone **Configuration**	Permet de configurer les paramètres des différentes voies.

Tableau A.5 : *les différents éléments de l'écran de configuration du module pesage* [8]

Fenêtre informations métrologiques

Désignation	Description
Unité	permet de choisir l'unité de mesure du poids : **g:** gramme, **kg :** kilogramme, **t:** tonne (métrique), **lb :** livre (lb = 453 g), **oz :** once (oz = 28,35 g), **sans :** toute unité.
Portée max (PM)	Poids maximum qu'il est possible de mesurer avec l'instrument et sans inclure le poids du support vide (au **format légal**).
Echelon	La valeur de l'échelon est de 1, 2 ou 5 multiplié par 10^n (n étant un entier positif ou négatif ou zéro avec une valeur absolue ≤ 3). **Exemple :** Pour un échelon de 0,002 (si l'unité choisie est le kg), la mesure augmente de 2 g à la fois.
Seuil de surcharge	Le seuil correspond à la valeur de poids au dessus de laquelle le poids n'est plus affiché (la surcharge est alors indiquée par une ligne > sur l'affichage). Il peut avoir les valeurs suivantes : • +9 échelons, • +2 % de la portée maximum, • +5 % de la portée maximum. **Exemple :** la portée maximum est à 150 kg, l'échelon est à 10 g. En fonction du choix de l'utilisateur, la limite de fonctionnement sera de : • 9 e : Portée maximum + 9 échelons, 150,09 kg par exemple, • +2 %PM : 102 % de portée max, 153 kg par exemple, • +5 %PM : 105 % de portée max, 157,5 kg par exemple, **Remarque :** Le seuil de sous-charge ne peut pas être paramétré : il définit la limite admissible de l'indication en dessous de zéro. C'est -2 % de la portée maximum (la sous-charge est alors indiquée avec une ligne < sur le panneau d'affichage).

Tableau A.6 : *les différents éléments de la fenêtre informations métrologiques* [8]

Fenêtre du zéro

Désignation	Description
Etendue de la plage	Tout écart à partir de zéro peut être corrigé à partir du moment où cette plage n'est pas dépassée. Elle est définie en tant que % de la portée maximum. Elle peut avoir les valeurs suivantes : • +/-2 % PM (+/- 2 % de la portée maximum), • +/-5 % PM (+/- 5 % de la portée maximum),
Zéro suiveur	Cette fonction facultative est utilisée pour compenser les écarts bas à partir de zéro au sein de la plage (+/-2 % de la portée maximum). Il n'est pas conseillé de choisir cette option dans les installations

	automatiques.

Tableau A.7 : *les différents éléments de la fenêtre du zéro* [8]

Fenêtre du format des données

L'écran de configuration permet de choisir le format d'affichage des mesures.

Entrez la valeur du poids :

- soit en tant qu'unité physique à virgule fixe : format Légal.
- soit en tant que centième d'une unité physique à virgule fixe : Haute résolution.

Exemple : Format Légal : La valeur 3014 signifie 301,4 kg si l'échelon est 2*10-1 kg.

Format Haute résolution : La valeur 301403 signifie 301,403 kg si l'échelon est 2*10-1 kg.

Fenêtre de la stabilité

Désignation	**Description**
Etendue de la plage	Tout écart à partir de zéro peut être corrigé à partir du moment où cette plage n'est pas dépassée. Elle est définie en tant que % de la portée maximum. Elle peut avoir les valeurs suivantes : • +/-2 % PM (+/- 2 % de la portée maximum), • +/-5 % PM (+/- 5 % de la portée maximum),
Zéro suiveur	Cette fonction facultative est utilisée pour compenser les écarts bas à partir de zéro au sein de la plage (+/-2 % de la portée maximum). Il n'est pas conseillé de choisir cette option dans les installations automatiques.

Tableau A.8 : *les différents éléments de la fenêtre de la stabilité* [8]

Fenêtre des filtres d'entrée de la mesure

Les filtres concernent l'entrée de mesure des capteurs de pesage.

Par défaut, un filtre unique est proposé. Il est défini pour la durée totale de l'action de pesage.

Pour augmenter la précision/vitesse d'exécution du pesage, 3 filtres différents sont utilisés de la manière suivante, pour la même action de pesage :

- filtre F1 associé à la phase 1 (phase par défaut).
- filtre F2 associé à la phase 2.

- filtre F3 associé à la phase 3.

La liste suivante répertorie les significations des coefficients de filtrage :

Valeur	Type de filtre	Caractéristiques
0	Aucune	non filtré
1	moyenne mobile	moyenne des 2 dernières mesures
2	moyenne mobile	moyenne des 3 dernières mesures
3	moyenne mobile	moyenne des 4 dernières mesures
4	moyenne mobile	moyenne des 5 dernières mesures
5	moyenne mobile	moyenne des 8 dernières mesures
6	moyenne mobile	moyenne des 16 dernières mesures
7	moyenne mobile	moyenne des 25 dernières mesures
8	moyenne mobile	moyenne des 32 dernières mesures
9	moyenne mobile	moyenne des 40 dernières mesures
10	moyenne mobile	moyenne des 50 dernières mesures
11	moyenne mobile	moyenne des 64 dernières mesures
12	filtre de second ordre	fréquence de coupure à 15 Hz
13	filtre de second ordre	fréquence de coupure à 10 Hz
14	filtre de second ordre	fréquence de coupure à 8 Hz
15	filtre de second ordre	fréquence de coupure à 6 Hz
16	filtre de second ordre	fréquence de coupure à 4 Hz
17	filtre de second ordre	fréquence de coupure à 2 Hz
18	filtre de second ordre	fréquence de coupure à 1 Hz
19	filtre de second ordre	fréquence de coupure à 0,8 Hz

Tableau A.9 : *les significations des coefficients de filtrage* [8]

Fenêtre du calcul du débit

Vous pouvez choisir le nombre de mesures pour le calcul du débit. (une mesure est prise toutes les 20 millisecondes). Le débit correspond à une différence dans les valeurs des poids filtrées pour un nombre de mesures configurées. Le débit est calculé grâce à la formule suivante :

Débit n = Valn - (Valn-b) où :

- b = le nombre de mesures pour le calcul du débit.
- Valn = la valeur du poids filtré à un moment n.
- Valn-b = la valeur du poids filtré à un moment n-b.

Exemple La figure ci-dessous montre un calcul sur 4 mesures.

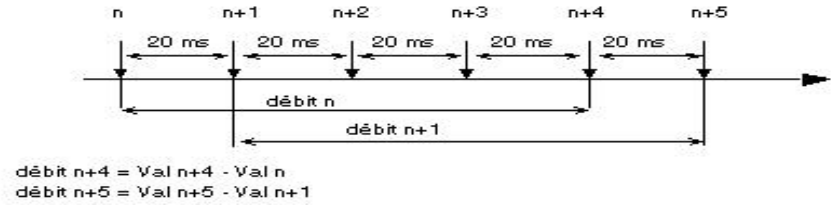


Figure A.23 : *exemple de calcule des mesures* [8]

Fenêtre de la tare

La tare cêst une charge placée sur le récepteur de charge avec le produit à peser. Par exemple : emballage ou conteneur du produit.

On peut introduire manuellement une valeur de la tare. Ainsi, cette valeur de la tare est prédéterminée ou manuelle et peut être envoyée au module. Elle est exprimée au format légal (unité physique avec virgule décimale fixe).

Cette tare doit être positive ou à zéro, et inférieure à la Portée Max

Fenêtre du contrôle des seuils

Désignation	**Description**
Active	La gestion des sorties TOR est opérationnelle si la case correspondante est cochée. Elle n'est pas cochée par défaut.
Sens	Le sens de la détection correspond à celui dans lequel les seuils ont été reconnus. Par exemple : ✓ **Pesage** (remplissage), ✓ **Dépesage** (emptying). Dans le cas du pesage, on parle de dépassement par une valeur supérieure ou, dans le cas du dépesage, de dépassement par une valeur inférieure. Pesage est sélectionné par défaut.
Sorties actives phase 1	Le choix se fait par rapport au contrôle de la sortie S0 en elle-même ou par les sorties S0 et S1 en même temps. Par défaut, le module active uniquement S0 dans la première phase.
Points de coupure	La mesure peut être associée à deux seuils pour les dosages suivants : un point de coupure gros débit et un point de coupure petit débit. En fonction de la logique définie, les sorties S0 et S1 sont sur 0 lorsque ces seuils sont atteints. Les valeurs de seuils autorisées se situent entre 0 et la portée maximum. Elles sont exprimées en haute résolution (un centième d'une unité physique avec virgule décimale fixe).
Temps de masquage	Définit le temps après le point de coupure gros débit durant lequel le

petit débit (PD)	module ne vérifie plus le poids/seuil. Cela masque le dépassement engendré lorsque le produit a une chute de tension. Les valeurs autorisées se situent entre 0 et 1,5 secondes par étape de 1/10 de seconde. Voir l'explication suivante. Par défaut, cette durée est à 0.

Tableau A.9 : *Fenêtre du contrôle des seuils* [8]

A.6.4.b module dœE/S TOR

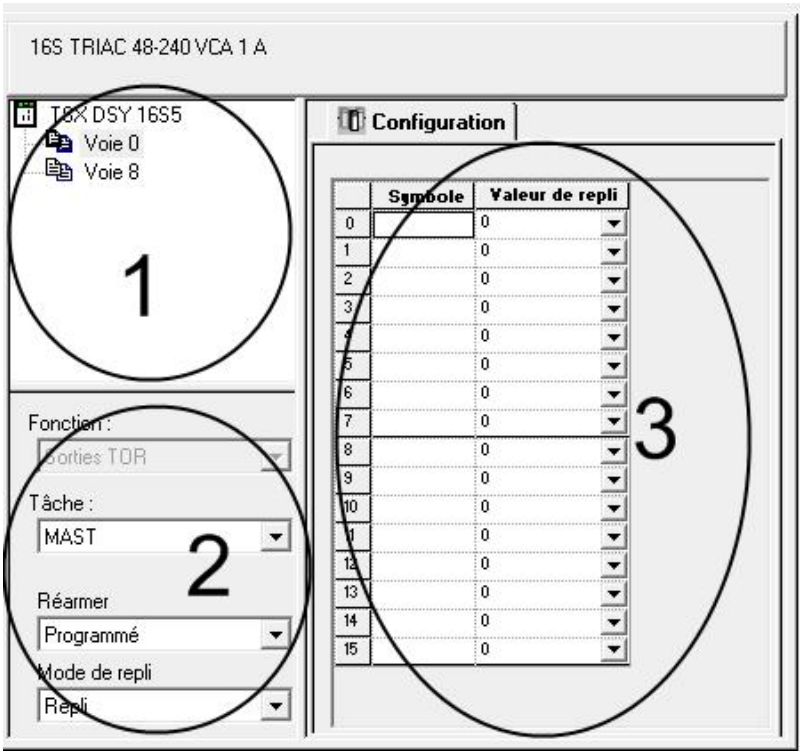


Figure A.24: *module dœE/S TOR* [9]

Adresse	Elément	Fonction
1	Zone **Voie**	Permet : ➢ de sélectionner la voie, ➢ d'afficher le Symbole, nom de la voie défini par l'utilisateur (au travers de l'éditeur de variables).
2	Zone **Paramètres Généraux**	Permet de choisir la fonction et la tâche associées par groupe de 8 voies : ➢ **Fonction :** définit la configuration/déconfiguration du groupe de voies sélectionné (autre que le groupe 0 à 7), ➢ **Tâche :** définit la tâche (**MAST**, **FAST** ou **AUX0/3**) dans laquelle seront échangés les objets à échange implicite des voies. La case à cocher **Surveillance alimentation** définit l'état activé ou désactivé de la surveillance de défaut de l'alimentation externe

		(disponible seulement pour certains modules TOR). Les menus déroulants **Réarmement** et **Mode de repli** permettent de configurer le réarmement et le mode de repli des sorties (disponible seulement pour certains modules TOR).
3	Zone **Configuration**	Permet de configurer les paramètres des différentes voies. Cette zone comprend différentes rubriques, affichées selon le choix du module TOR. La colonne **Symbole** affiche le symbole associé à la voie lorsque celui-ci a été défini par l'utilisateur (depuis l'éditeur de variables).

Tableau A.10: *Module dÕE /S TOR* [8]

Fenêtre des Tâches

Ce paramètre définit la tâche processeur dans laquelle se font l'acquisition des entrées et la mise à jour des sorties

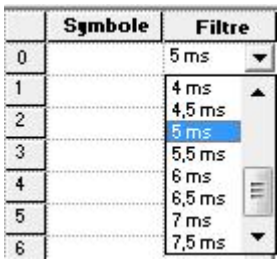
Les choix possibles sont les suivants :

- la tâche MAST.
- la tâche FAST.
- les tâches secondaires AUX0/3. [8]

Fenêtre de la surveillance d'alimentation externe d'un module TOR

Ce paramètre définit l'état (activation ou désactivation) de la surveillance des erreurs d'alimentation externe.

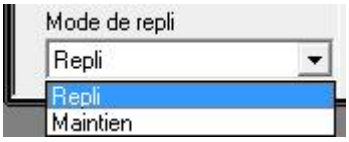
Fenêtre de la Filtrage



Ce paramètre définit le temps de filtrage de la voie sélectionnée.

Les valeurs par défaut sont les suivantes : de 0,1 à 7,5 ms par incréments de 0,5 ms.

Fenêtre du paramètre Mode de repli



Ce paramètre définit le mode de repli que prennent les sorties lorsque le contrôleur passe à Stop, après une erreur de processeur, de rack ou de câble inter-rack .Les modes possibles sont :

Repli : Les voies sont mises à l'état 0 ou 1 en fonction de la valeur de repli paramétrée pour le groupe de 8 voies correspondant.

Maintien : Les sorties conservent l'état dans lequel elles se trouvaient avant le passage en Stop.

Fenêtre du paramètre Réarmement des sorties

Ce paramètre définit le mode de réarmement des sorties déconnectées. Les modes possibles sont :

Programmé : Le réarmement est exécuté par une commande de l'application automate ou par l'intermédiaire de l'écran de mise au point approprié.

Remarque : Afin d'éviter des réarmements répétitifs rapprochés, le module assure automatiquement un délai de 10 s entre deux réarmements.

Automatique : Le réarmement est réalisé automatiquement toutes les 10 s jusqu'à la disparition de l'erreur.

A.6.4.c module dôE/S Analogique

L'écran de configuration du module analogique sélectionné à partir du rack est :

- Si en cliquant sur la référence de l'équipement les onglets suivant ils sôaffichent :

Description, qui donne les caractéristiques de l'équipement.

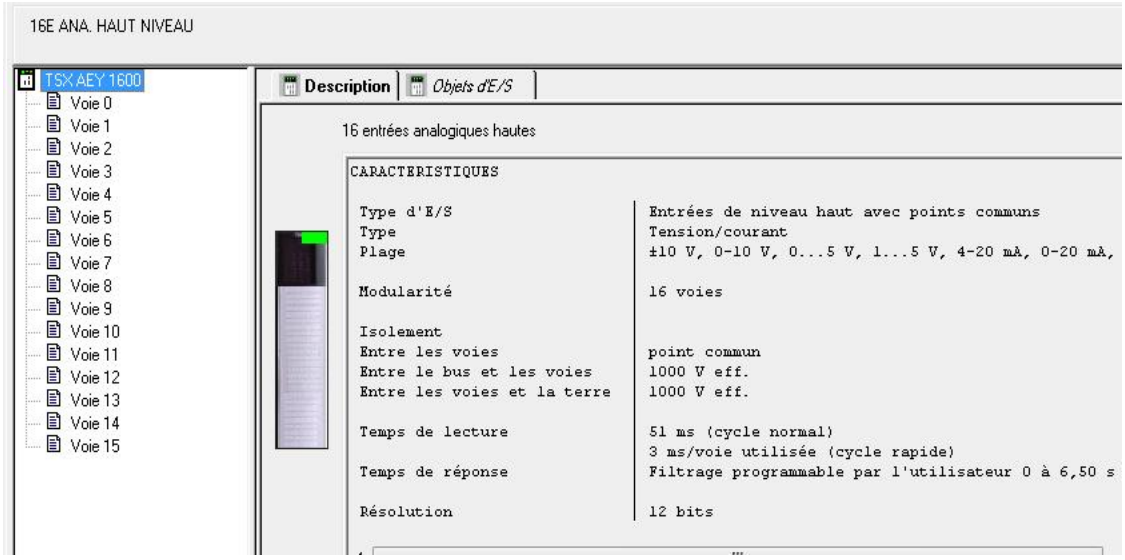


Figure A.25 : *module dôE/S Analogique (description)* [9]

Objets d'E/S, qui permet de présymboliser les objets d'entrée/de sortie.

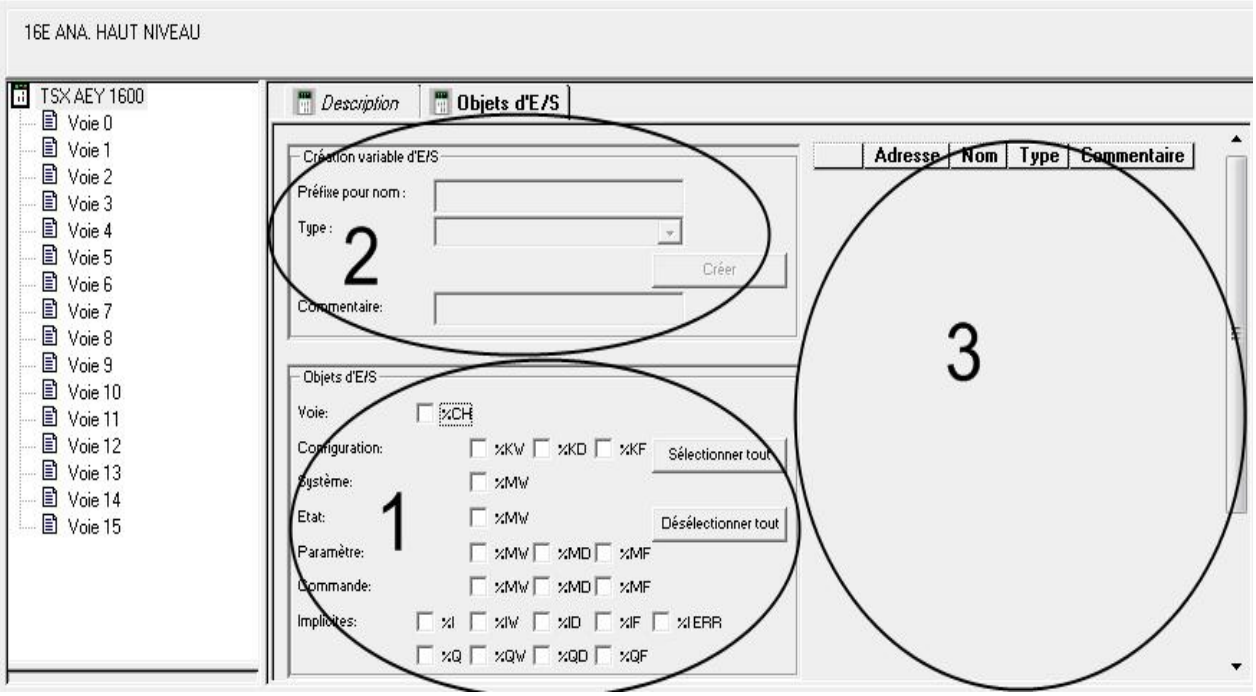


Figure A.26: *module dÕE/S Analogique(objet dÕE /S)*[9]

1. Description de la zone Objets d'E/S

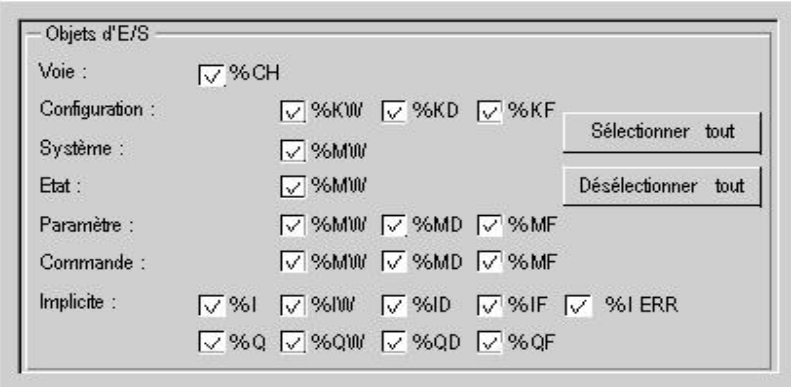


Figure A.27: *Description de la zone Objets d'E/S* [9]

La sélection de divers objets à l'aide des cases à cocher permet de les afficher dans la zone Adresse Nom Type Commentaire dès que vous avez cliqué sur le bouton Mettre à jour grille avec... dans la zone Mise à jour.

Il est possible de sélectionner les divers objets par type :

- Voie : pour les voies de module ou un équipement de bus.
- Configuration : pour les objets langage de configuration.
- Système : pour les objets langage gérant les échanges explicites.
- Etat : pour les objets langage d'état (accessibles par READ_STS).

- Paramètre : pour les objets langage de configuration (accessibles par READ_PARAM, WRITE_PARAM, SAVE_PARAM et RESTORE_PARAM).
- Commande : pour les objets langage de commande (accessibles par WRITE_CMD).
- Implicite : pour les objets langage implicite du module ou d'un équipement de bus. [8]

2. Description de la zone Créer une variable d'E/S

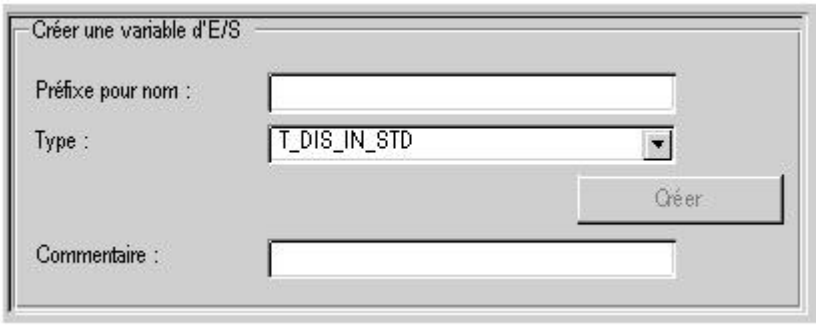


Figure A.28 : *Description de la zone Créer une variable d'E/S* [9]

Après avoir sélectionné un ou plusieurs objets dans la zone Adresse Nom Type Commentaire, vous pouvez sélectionner un type d'IODDT et créer une ou plusieurs variables de ce type en cliquant sur Créer.

Règles de fonctionnement :

- En sélectionnant une ligne de la zone Adresse Nom Type Commentaire, vous pouvez créer une variable et lui attribuer un nom. Il vous est également possible d'affecter un commentaire à cette variable.
- En sélectionnant plusieurs lignes homogènes (du même type) dans la zone Adresse Nom Type Commentaire, vous pouvez créer automatiquement plusieurs variables portant des préfixes identiques (la première variable porte le suffixe 0, la seconde le suffixe 1, la troisième le suffixe 2, etc.). Cette méthode s'applique également au commentaire formulée pour la variable (le premier commentaire porte le suffixe 0, le deuxième le suffixe 1, le troisième le suffixe 2, etc.
- Si la ou les variables sélectionnées sont de type EDT, la zone Type est grisée. La
- sélection du type est possible uniquement lorsque plusieurs types sont disponibles

3. Description de zone Adresse Nom Type Commentaire

Cette zone permet :

	Adresse	Nom	Type	Commentaire
8	%IW0.2.12		INT	commentaire 12
9	%IW0.2.13		INT	commentaire 12
10	%IW0.2.14		INT	commentaire 14
11	%IW0.2.15		INT	commentaire 15
12	%QW0.3.0		INT	commentaire 3.0
13	%QW0.3.1		INT	commentaire 3.1
14	%QW0.3.2		INT	commentaire 3.2
15	%QW0.3.3		INT	commentaire 3.3
16	%QW0.3.4		INT	commentaire 3.4
17	%QW0.3.5		INT	commentaire 3.5
18	%QW0.3.6		INT	commentaire 3.6

Figure A.29 : *Description de la zone adresse, non, type et commentaire* [9]

- d'afficher les objets sélectionnés dans les zones Objets UC et Objets d'E/S ;
- de sélectionner une ou plusieurs lignes d'objet afin de créer des variables et de les associer à ces lignes ;
- d'ouvrir la fenêtre Propriétés des données ;
- d'afficher le commentaire associé à la variable.

Une colonne supplémentaire est disponible pour les automates Quantum :

- La colonne RAM d'état permet de voir la correspondance entre l'adresse topologique et la RAM d'état.

4. Description de la zone Mise à jour

En cliquant sur le bouton Mettre à jour grille avec..., vous pouvez mettre à jour la zone Adresse Nom Type Commentaire à partir des informations sélectionnées dans les zones Objets UC et Objets d'E/S.

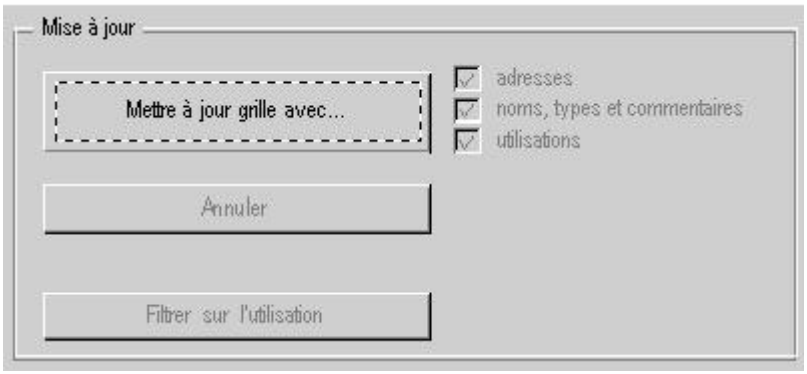


Figure A.30: *Description de la zone Mise à jour*[1]

La zone Adresse Nom Type Commentaire est aussi utilisée pour sélectionner des objets dans le but de créer des noms de variable et des commentaires (ne vaut que pour les modules, les boucles de régulation et les équipements du bus de communication).

Le bouton Annuler permet d'annuler la mise à jour des noms d'objet depuis la zone Adresse Nom Type Commentaire. Le bouton Filtrer sur l'utilisation permet d'afficher uniquement les

objets utilisés dans le projet. Les cases à cocher relatives aux adresses, noms, types et commentaires et à l'utilisation permettent d'afficher, respectivement :

- la colonne Adresse,
- les colonnes Nom , Type et Commentaire,
- les variables utilisées dans le programme (en gras).

Si en cliquant sur la voie lôonglet suivant il sôaffiche:

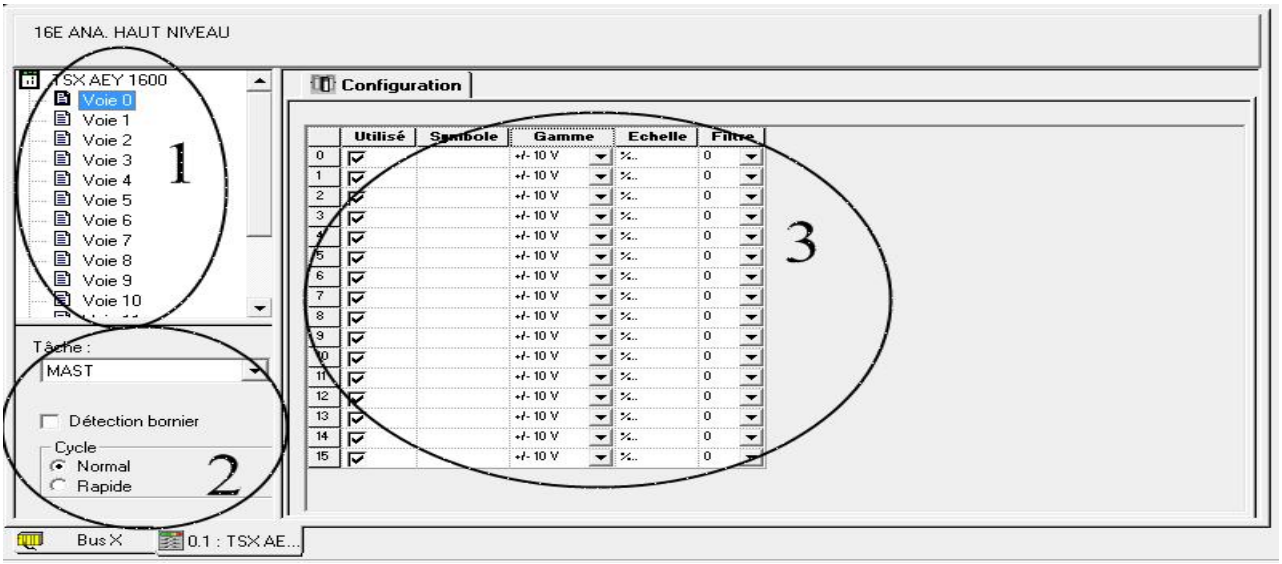


Figure A.31: *Configuration des voies* [9]

Nombre	Elément	Fonction
1	Zone Voie	Permet : ➢ de choisir la voie. ➢ d'afficher le Symbole, nom de la voie défini par lôutilisateur (au travers de l'éditeur de variables).
2	Zone Paramètres généraux	Cette zone permet de sélectionner la tâche associée à la voie : ➢ Tâche : définit la tâche MAST, FAST ou AUX0/3 dans laquelle seront échangés les objets à échanges implicites de la voie. ➢ La case à cocher Détection du bornier permet de modifier la fonction de détection du bornier. ➢ La zone Cycle permet de définir le cycle de scrutation des entrées (disponible seulement pour certains modules

		analogiques).
3	Zone Configuration	Cette zone vous permet de définir les paramètres applicables à différentes voies. Cette zone comprend différentes rubriques, affichées selon le choix du module analogique. La colonne Symbole affiche le symbole associé à la voie lorsque celui-ci a été défini par l'utilisateur (depuis l'éditeur de variables).

Tableau A.11 : *Configuration des voies* [8]

A.6.4.d module dœE/S BusX distant

Généralité :

Le bus de l'automate Premium permet de raccorder 8 racks de 12 emplacements (TSX RKY 12EX) ou 16 racks de 4, 6 ou 8 emplacements (TSX RKY 4EX/6EX/8EX) répartis sur une longueur de 100 mètres maximum.

Dans le cas d'applications nécessitant des distances plus élevées entre les racks, le module d'extension du bus X (TSX REY 200) permet d'augmenter de façon considérable cette distance tout en conservant les caractéristiques et la performance d'une station automate composée d'un seul segment de bus X sans module d'extension.

Le système se compose des éléments suivants :

- un module d'extension de bus X (TSX REY 200) appelé « maître » situé sur le rack ayant l'adresse 0 (rack hébergeant le processeur) et sur le segment principal du bus X. Ce module possède deux voies permettant d'étendre les deux segments du bus X à une distance maximum de 250 mètres,
- un ou deux modules TSX REY 200 modules appelés « esclave » et chacun situé sur un rack sur les segments de bus étendus,
- chacun des modules esclave est raccordé au module maître par un câble TSX CBRY 2500 équipé de connecteurs TSX CBRY K5.

Exemple de topologie

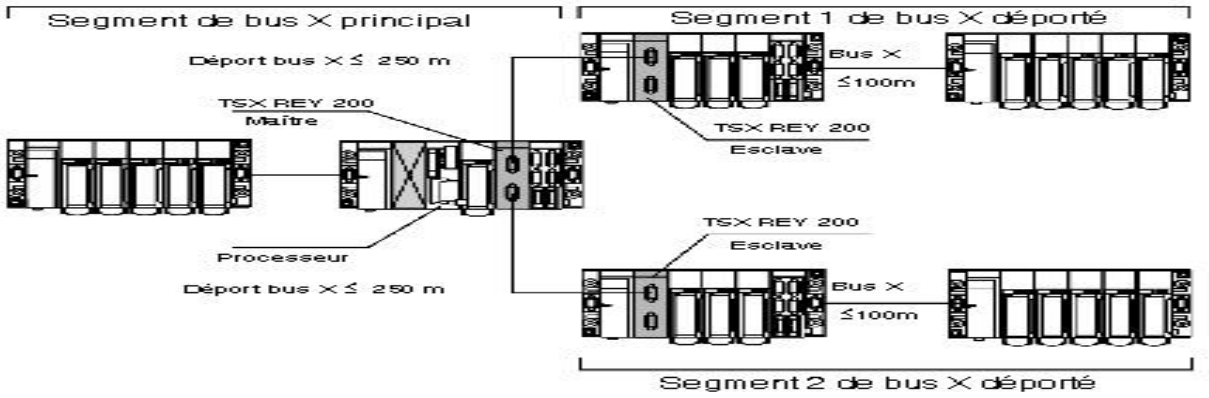
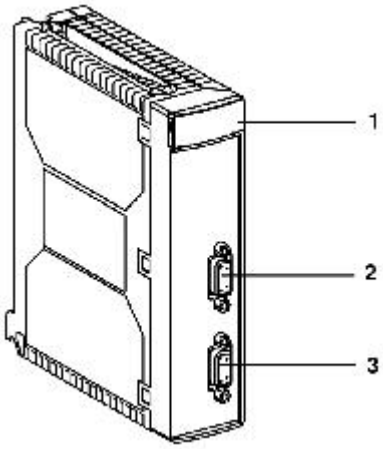


Figure A.32: *module d'E/S BusX distant (Exemple de raccordement)*[8]

Schéma descriptif



1) Bloc de visualisation composé de 6 voyants :

Voyant RUN : indique l'état de fonctionnement du module.

Voyant ERR : signale un défaut à l'intérieur du module.

Voyant I/O : signale un défaut extérieur au module.

Voyant MST : indique l'état de la fonction maître ou esclave du n

Figure A.33: *module d'E/S BusX distant (Schéma descriptif)*[8]

Voyant CH0 : indique l'état de fonctionnement de la voie 0.

Voyant CH1 : indique l'état de fonctionnement de la voie 1.

2) Connecteur pour la liaison de la voie 0 du module.

3) Connecteur pour la liaison de la voie 1 du module.

Configuration

La configuration du module comme maître ou esclave est automatique :

- si le module est installé sur le rack d'adresse 0, il sera automatiquement déclaré comme maître.
- si le module est installé sur un rack d'adresse différente de 0, il sera automatiquement déclaré comme esclave.

Note : Si 2 racks sont déclarés à l'adresse 0, le module maître doit être positionné sur le rack hébergeant les adresses de module « basses », comme indiqué dans la figure ci-dessous.

Adresses modules « basses » :

- adresses 0 à 6 sur TSX RKY 8EX
- adresses 0 à 4 sur TSX RKY 6EX
- adresses 0 à 2 sur rack TSX RKY 4EX

Exemple : 2 racks TSX RKY 8EX à l'adresse 0.

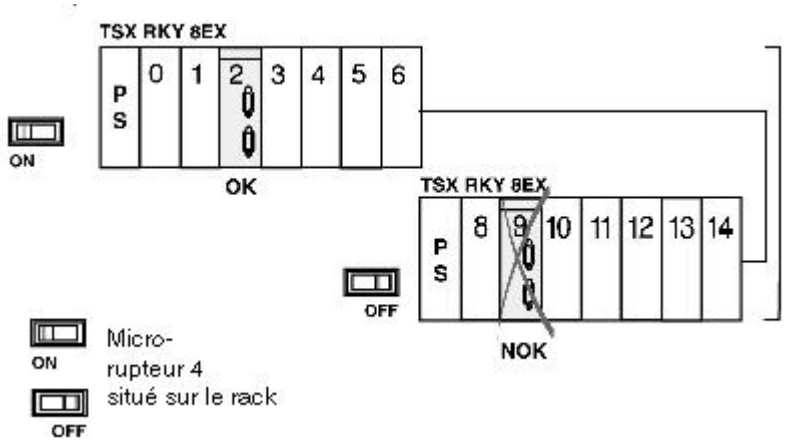


Figure A.34: *module dE/S BusX distant (exemple de configuration)*[8]

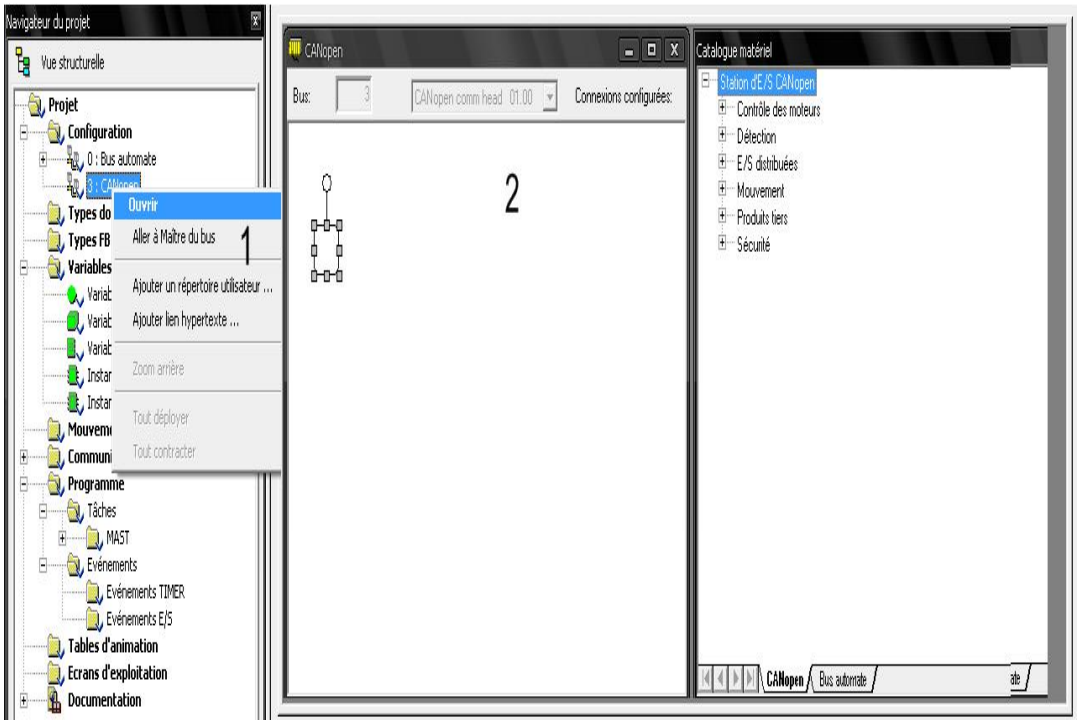
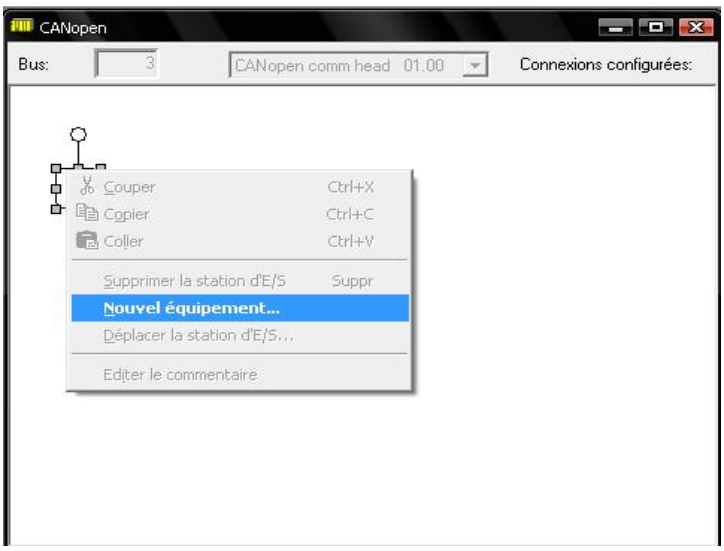
A.6.5 Configuration d'équipements sur le bus de terrain

On traite le bus de terrain CANopen

Dans le cas des automates Modicon M340, vous pouvez configurer le bus CANopen à l'aide d'esclaves (63 maximum). Unity Pro se charge de la configuration complète, l'utilisation d'un logiciel complémentaire n'est pas nécessaire

Configuration de l'esclave CANopen

Le tableau ci-dessous décrit la procédure de configuration de l'esclave CANopen.

Etape	Opération
1	Dans le Navigateur de projet de Unity Pro, développez complètement le répertoire **Configuration,** puis cliquez a droite sur CANopen puis sur **ouvrir** **Résultat** : la fenêtre CANopen et son station dŒ /S apparaîtent : 
2	En cliquant a droite sélectionner **Nouvel équipement.**  **Résultat :** La fenêtre Nouvel équipement s'affiche.

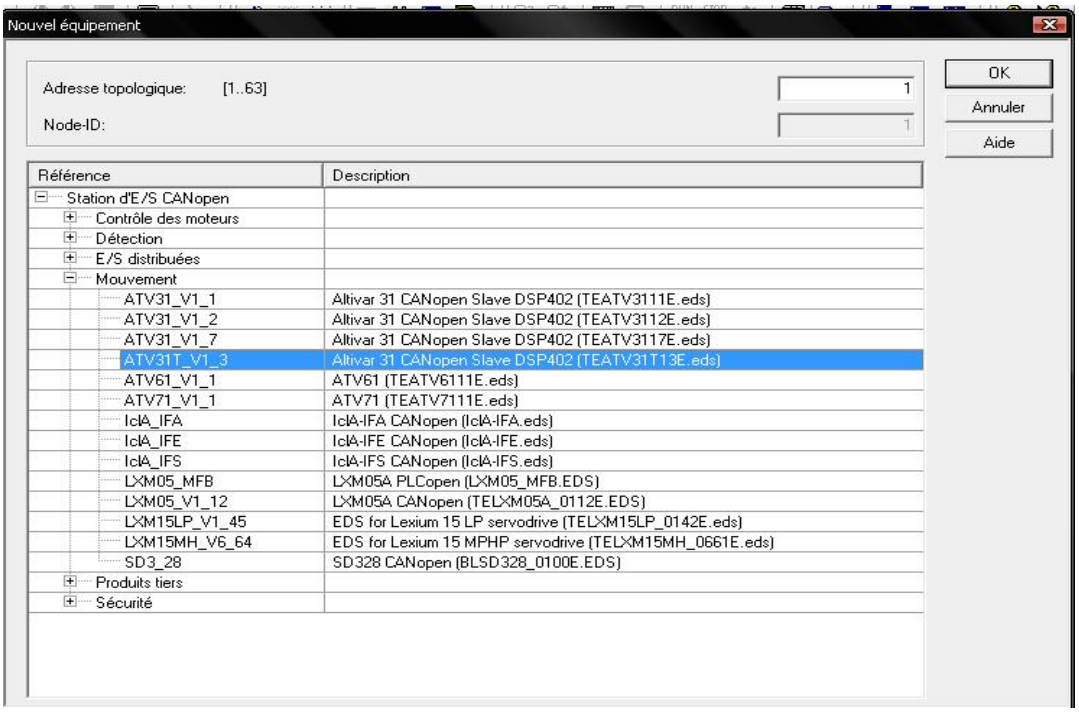
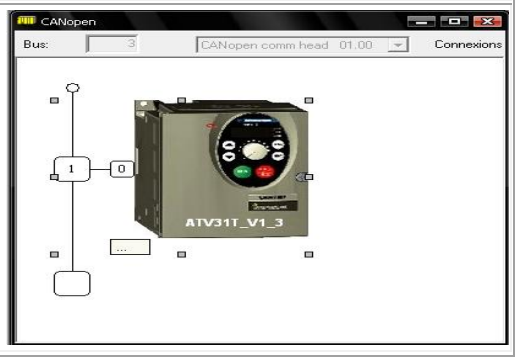
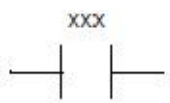
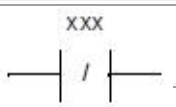
	
3	Définissez l'adresse esclave dans le champ de l'adresse topologique. Sélectionnez l'équipement esclave.
4	Cliquez sur **OK** pour confirmer votre sélection. **Résultat :**la fenêtre CANopen s'affiche avec le nouvel équipement sélectionné. 

Tableau A.12: *la procédure de configuration de l'esclave CANopen* [1]

A.7 Instruction des Langages de programmation LD

Un contact ne modifie pas la valeur du paramètre réel associé.

Les contacts disponibles sont les suivants :

Désignation	Représentation	Description
A fermeture		Dans le cas de contacts à fermeture, l'état de la liaison de gauche est transféré vers la liaison de droite si l'état du paramètre booléen réel associé (indiqué par xxx) est ON. Sinon, l'état de la liaison de droite est OFF.
A ouverture		Dans le cas de contacts à ouverture, l'état de la liaison de gauche est transféré vers la liaison de droite si

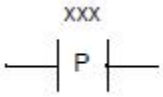
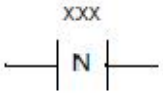
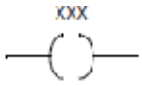
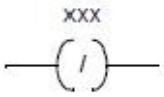
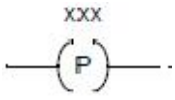
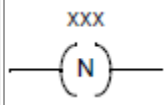
		l'état du paramètre booléen réel approprié (indiqué par xxx) est OFF. Sinon, l'état de la liaison de droite est OFF.
Contact de détection de transitions positives		Dans le cas de contacts de détection de transitions positives, la liaison de droite est ON pour un cycle de programme, si un passage de OFF à ON du paramètre réel booléen (xxx) associé a lieu et quœn même temps, l'état de la liaison de gauche est ON. Sinon, lœétat de la liaison de droite est 0.
Contact de détection de transitions négatives		Dans le cas de contacts de détection de transitions négatives, la liaison de droite est ON pour un cycle de programme, si un passage de ON à OFF du paramètre réel booléen (xxx) associé a lieu et qu'en même temps, l'état de la liaison de gauche est ON. Sinon, lœétat de la liaison de droite est 0.

Tableau A.13 : *Le langage LD (les différents contacts)* [8]

Bobines : Une bobine est un élément LD permettant de transférer l'état de la liaison horizontale sur la gauche, inchangée, vers la liaison horizontale sur la droite. L'état est stocké dans le paramètre booléen réel respectif. elles occupent une cellule.

Types de bobines :

Les bobines suivantes sont disponibles :

Désignation	**Représentation**	**Description**
Bobine		Dans le cas de bobines, l'état de la liaison de gauche est transféré vers le paramètre booléen réel associé (indiqué par xxx) et la liaison de droite.
bobine inverse		Dans le cas de bobines inverses, l'état de la liaison de gauche est copié sur la liaison de droite. L'état inversé de la liaison de gauche est copié vers le paramètre booléen réel associé (indiqué par xxx). Si la liaison de gauche est OFF, alors la liaison de droite sera également OFF et le paramètre booléen réel associé sera ON.
Bobine de détection de transitions positives		Dans le cas de bobines de détection de transitions positives, l'état de la liaison de gauche est copié sur la liaison de droite. Le paramètre réel associé du type de données EBOOL (indiqué par xxx) est 1 pour un cycle de programme, si un passage de 0 à 1 de la liaison gauche est effectué.
Bobine de détection de transitions négatives		Dans le cas de bobines de détection de transitions négatives, l'état de la liaison de gauche copié sur la liaison de droite. Le paramètre réel booléen associé (indiqué par xxx) est 1 pour un cycle de programme, si un passage de 1 à 0 de la liaison gauche est effectué.

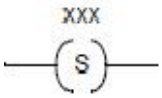
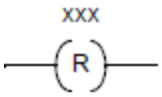
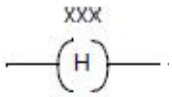
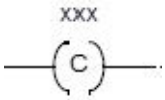
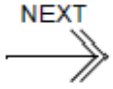
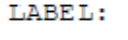
Bobine d'enclenchement		Avec une bobine d'enclenchement, l'état de la liaison de gauche est copié sur la liaison de droite. Le paramètre booléen réel associé (indiqué par xxx) est défini sur ON si l'état de la liaison de gauche est ON, sinon il reste inchangé. Le paramètre booléen réel associé peut être réinitialisé via la bobine de réinitialisation.
Bobine de réinitialisation		Avec une bobine de réinitialisation, l'état de la liaison de gauche est copié sur la liaison de droite. Le paramètre booléen réel associé (indiqué par xxx) est défini sur OFF si l'état de la liaison de gauche est ON, sinon il reste inchangé. Le paramètre booléen réel associé peut être enclenché via la bobine d'enclenchement.
Bobine d'arrêt		Avec des bobines d'arrêt, si le statut de la liaison de gauche est 1, l'exécution du programme est arrêtée immédiatement. (Avec des bobines d'arrêt, l'état de la liaison de gauche n'est pas copié sur la liaison de droite.)
Bobine d'appel		Avec des bobines d'appel, l'état de la liaison de gauche est copié vers la liaison de droite. Si l'état de la liaison de gauche est ON alors le sous-programme associé (indiqué par xxx) est appelé. Le sous-programme à appeler doit se trouver dans la même tâche que la section LD appelante. Il est possible d'appeler des sous-programmes au sein de sous-programmes. Les sous-programmes sont un complément de la norme CEI 61131-3 et doivent être activés de manière explicite. Dans les sections d'actions SFC, les bobines d'appel (appels de sous-programmes) ne sont autorisés que si le mode Multitoken a été activé.

Tableau A.14: *Le langage LD (les différentes bobines)* [8]

Contrôle :

Les contrôles suivants sont disponibles.

Désignation	**Représentation**	**Description**
Jump		Si l'état de la liaison gauche est 1, un saut est exécuté jusqu'à l'étiquette (dans la section courante). Pour générer un saut inconditionnel, l'objet saut est placé directement sur la barre d'alimentation gauche. Pour générer un saut conditionnel, l'objet saut est placé à la fin d'une rangée de contacts.
Libellé		Les repères (destinations de saut) sont représentés comme du texte avec deux-points à la fin. Le texte est limité à 32 caractères et doit être unique dans l'ensemble de la section. Le texte doit respecter les conventions de nommage générales.

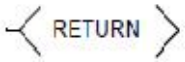
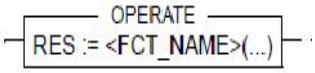
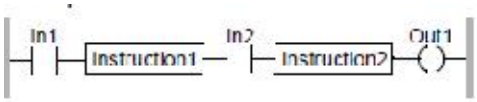
		Les étiquettes de saut ne peuvent être placées que dans la première cellule directement sur la barre d'alimentation gauche.
Return		Des objets RETURN ne peuvent pas être utilisés dans le programme principal. ➢ Dans un DFB, un objet RETURN force le retour au programme qui a appelé le DFB. ➢ Le reste de la section de DFB contenant l'objet RETURN n'est pas exécuté. ➢ Les sections suivantes du DFB ne sont pas exécutées. Le programme qui a appelé le DFB sera exécuté après retour du DFB. Si le DFB est appelé par un autre DFB, le DFB appelant sera exécuté après le retour. ➢ Dans un SR, un objet RETURN force le retour au programme qui a appelé le SR. ➢ Le reste du SR contenant l'objet RETURN n'est pas exécuté. Le programme qui a appelé le SR sera exécuté après retour du SR.

Tableau A.15: *Le langage LD (les contrôles)* [1]

Les objets suivants sont disponibles :

Désignation	**Représentation**	**Description**
Bloc opération		Si l'état de la liaison gauche est 1, l'instruction ST comprise dans le bloc est exécutée. Toutes les instructions ST sont permises sauf les instructions de commande : (RETURN,JUMP,IF,CASE,FOR etc.) Pour les blocs opération, quel que soit le résultat de l'instruction ST, l'état de la liaison gauche est transmis à la liaison droite. Un bloc peut contenir jusqu'à 4 096 caractères. Si tous les caractères ne peuvent pas être affichés, les premiers caractères seront affichés suivis de points de suspension (...). Un bloc opération occupe 1 ligne et 4 colonnes. Exemple :  Dans l'exemple, Instruction1 est exécutée si In1=1. Instruction2 est exécutée si In1=1 et In2=1 (le résultat de Instruction1 ne joue aucun rôle pour l'exécution de Instruction2). Out1 est 1, si In1=1 et

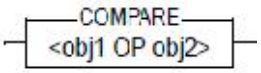
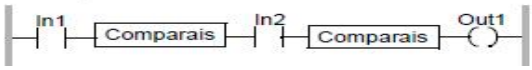
		In2=1 (les résultats de Instruction1 et Instruction2 n'influent pas sur l'état de Out1).
Bloc de comparaison horizontal		Les blocs de comparaison horizontaux servent à exécuter une expression de comparaison (<, >, <=, >=, =, <>) dans le langage de programmation ST. (Remarque : La même fonctionnalité est également disponible via les expressions ST.) Un bloc comparaison exécute un ET de sa broche d'entrée gauche et du résultat de sa condition de comparaison, puis affecte le résultat du ET de façon inconditionnelle à la broche de sortie droite. Par exemple, lorsque l'état de la liaison gauche est 1 et que le résultat de la comparaison est 1, l'état de la liaison droite est 1. Un bloc de comparaison horizontal peut contenir jusqu'à 4 096 caractères. Si tous les caractères ne peuvent pas être affichés, les premiers caractères seront affichés suivis de points de suspension (...). Un bloc de comparaison horizontal occupe une ligne et deux colonnes. Exemple :  Dans l'exemple, Comparaison1 est exécutée si In1=1. Comparaison2 est exécutée si In1=1 , In2=1, résultat de Comparaison1=1. Out1 est 1, siIn1=1, In2=1, le résultat de Comparaison1=1 et le résultat de Comparaison2=1.

Tableau A.16: *Le langage LD (les objets)* [8]

A.8 Description des données

A.8.1 Présentation générale des données

Une donnée désigne un d'objet qui peut être instancié tel que :

- Une variable,
- Un bloc fonction.

Les données sont définies en trois phases qui sont :

- la phase types de données, dans laquelle est précisée :
- sa catégorie,
- son format.
- La phase instances de données, dans laquelle est défini son emplacement mémoire et sa propriété qui est:

- Localisée ou,
- Non localisée.
- La phase références de données, dans laquelle est défini son moyen d'accès :
- par valeur immédiate,
- par nom,
- par adresse.

Les trois phases caractérisant les données :

Instancier une donnée consiste à partir de son type à lui allouer un emplacement mémoire.

Référencier une donnée consiste à lui définir une référence (nom, adresse, etc...) permettant de ***lâatteindre en mémoire.***

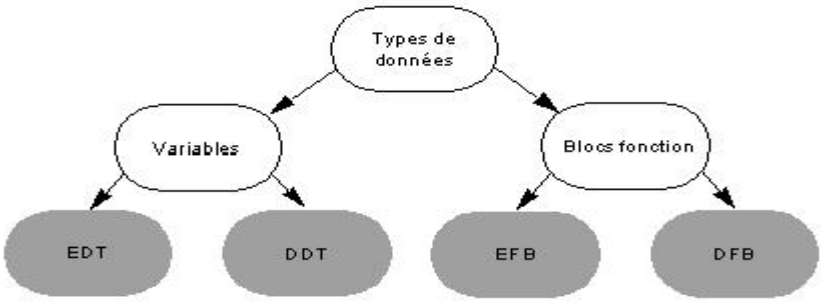
A.8.2 Types de données

Un type de donnée est une information logicielle qui spécifie pour une donnée:

- Sa structure,
- Son format,
- La liste de ses attributs,
- Son comportement.

Ces propriétés sont partagées par toutes les instances du type de donnée.

Les familles de types de données sont classées dans différentes catégories (gris foncé).



Familles de types de données et leurs définitions. [8]

Famille	Définition
Structures	Types de données élémentaires (Elementary data types) tels que: ➢ Bool ➢ Int ➢ Byte ➢ Mot ➢ Dword etc.
Tableaux	Types de données dérivés (Derived data types) tels que: ➢ tableaux, qui contiennent des éléments de même type: ➢ tableaux de Bool (tableaux d'EDT), ➢ tableaux de tableaux (tableaux de DDT), ➢ tableaux de structures (tableau de DDT), ➢ structures, qui contiennent des éléments de type différents : • structures de Bool, Word, etc. (structures EDT) • structures de tableaux, structures de structures, structures de tableaux/structures (structures de DDT), • structures de Bool, structures de tableaux, etc. (structures EDT et DDT) • structures concernant les données d'entrées/de sorties (structures d'IODDT), • Structures contenant des variables restituant les propriétés et l'état d'une action ou transition d'un diagramme fonctionnel en séquence (Séquentiel Function Chart).
EFB	Blocs fonction élémentaires écrits en langage C. Ils comprennent : ➢ des variables d'entrées, ➢ des variables internes, ➢ des variables de sorties, ➢ un algorithme de traitement.
DFB	Blocs fonctions utilisateurs (Derived function blocs) écrits en langage d'automatisme (Litteral Structuré, Liste d'instructions, etc...). Ils comprennent : ➢ des variables d'entrées, ➢ des variables internes, ➢ des variables de sorties, ➢ un algorithme de traitement.

Tableau A.17 : *Familles de types de données et leurs définitions* [8]

A.8.3 Les instances de données

Une instance de données est une entité fonctionnelle individuelle, qui possède toutes les caractéristiques du type de données duquel elle dépend.

Une ou plusieurs instances peuvent être rattachées à un type de données.

Lôinstance de données peut avoir une allocation mémoire:

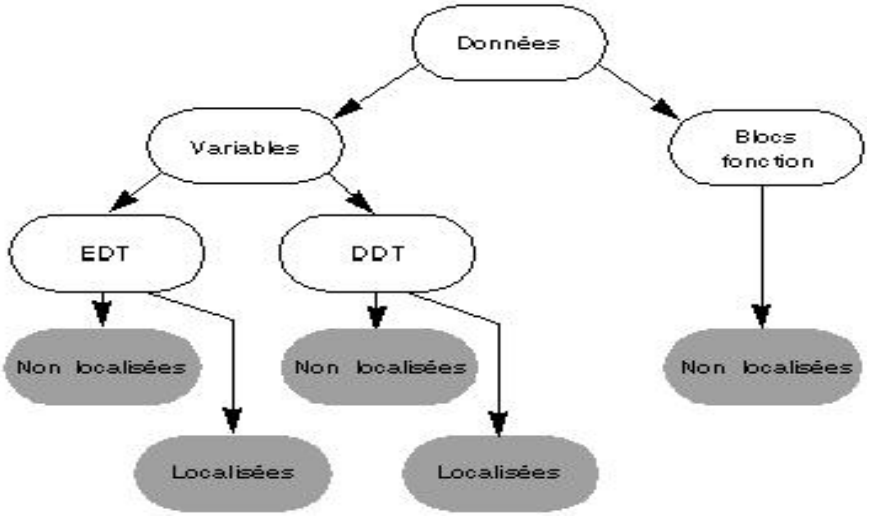
- Localisée : lôemplacement mémoire de lôinstance est fixe, il est prédéfini et ne change jamais.

Lôinstance est repérée par un nom (symbole) choisi par lôutilisateur et une adresse topologique définie par le constructeur, ou uniquement par lôadresse topologique constructeur.

- Non localisée : L'emplacement mémoire de l'instance est alloué automatiquement par le système, il peut changer à chaque génération de l'application.

Lôinstance est repérée par un nom (symbole) choisi par lôutilisateur.

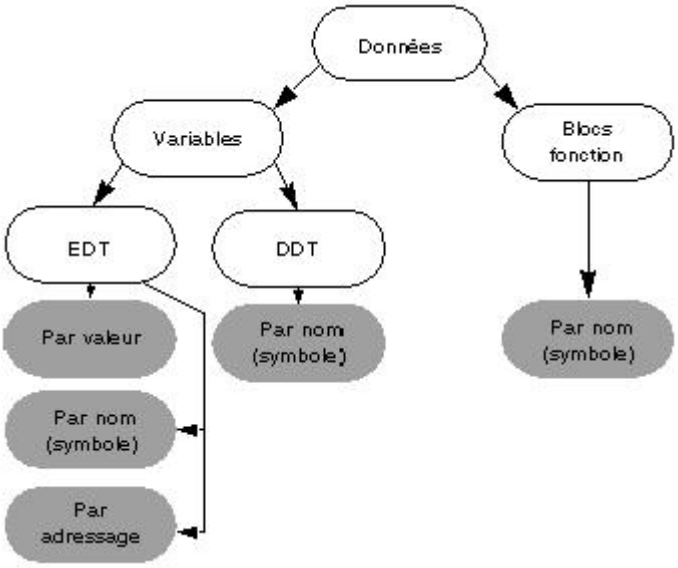
Allocation mémoire des instances appartenant aux différents types.



A.8.4 Références de données

Une référence de données permet à lôutilisateur dôaccéder à lôinstance de cette donnée soit par:

- Valeur immédiate, vrai uniquement pour les données de type EDT
- Adressage, vrai uniquement pour les données de type EDT
- Nom (symbole), vrai pour tous les types de données EDT, DDT, EFB, DFB ainsi que les objets SFC.



A.9 Editeur de données

L'éditeur de données propose les fonctions suivantes :

- déclaration d'instances de variable.
- définition de types de données dérivés (DDT).
- déclaration d'instance de blocs fonctions élémentaires et dérivés (EFB/DFB).
- définition des paramètres de blocs fonctions dérivés (DFB).

Lôonglet variable permet de :

- définition d'un symbole pour les variables
- boîte de dialogue de sélection personnalisée pour les types de données dérivés
- affectation d'une adresse
- symbolisation automatique des variables E/S
- affectation d'une valeur initiale
- saisie d'un commentaire

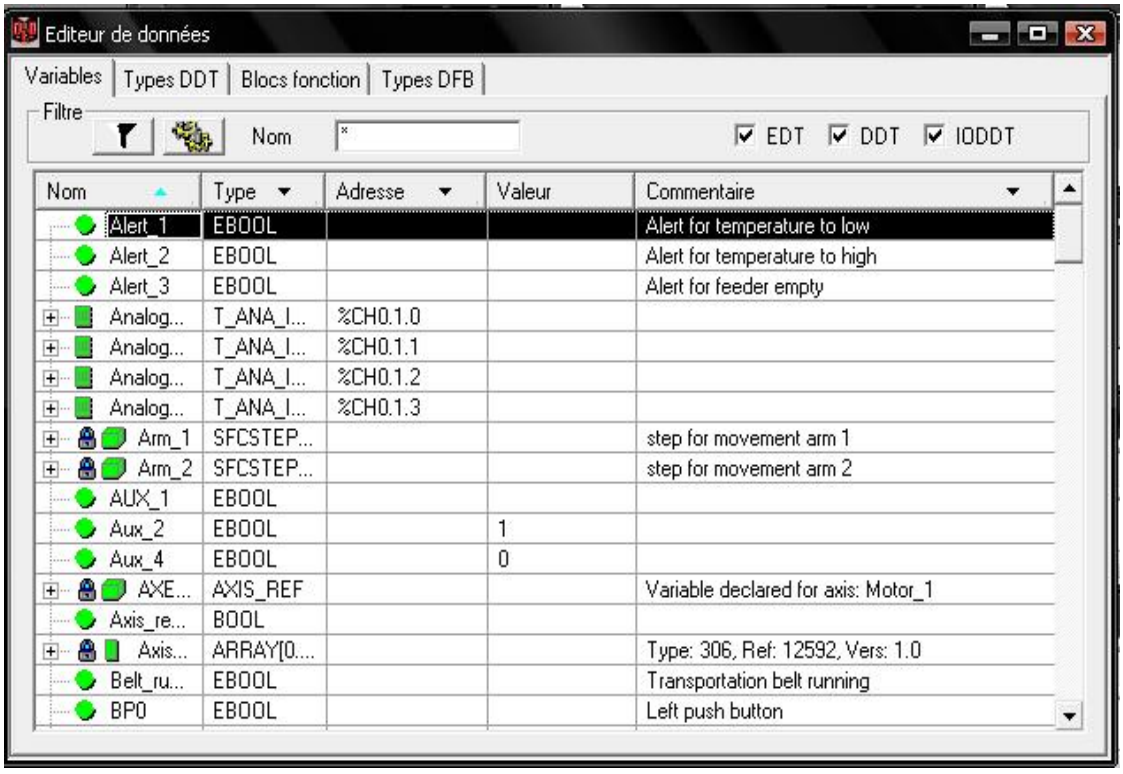


Figure A.35 : *éditeur de données* [1]

Types de données liés au matériel (IO DDT)

Les IO DDT servent à affecter la structure E/S complète d'un module à une variable unique.

Affectation d'IO DDT :

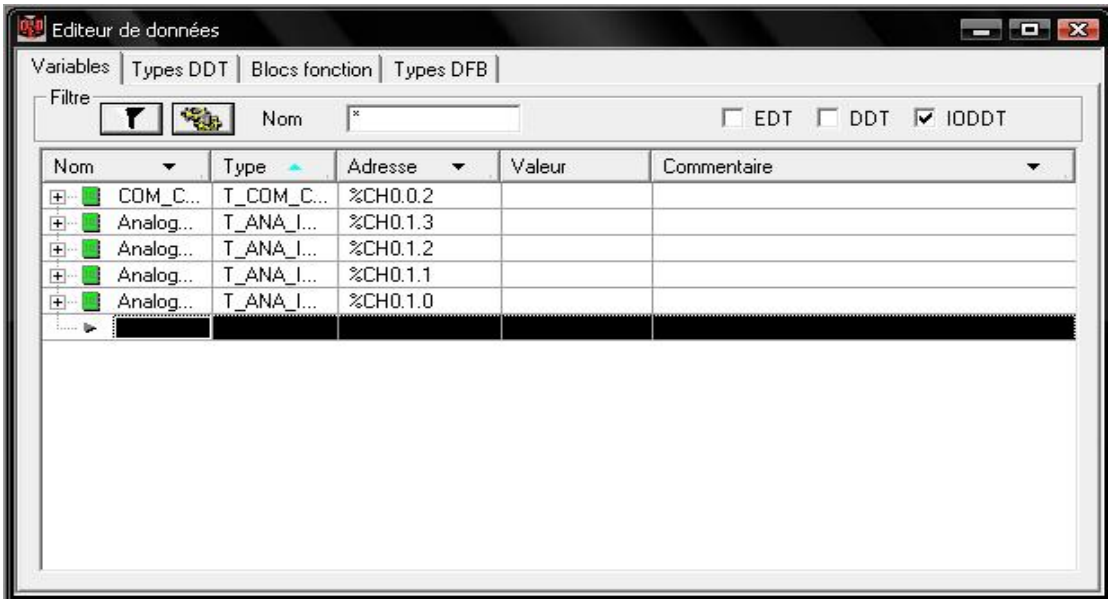


Figure A.36: *Affectation d'IO DDT* [9]

Types de données dérivés (DDT) :

L'onglet Types DDT permet de définir des types de données dérivés (DDT).

Un type de données dérivé est la définition d'une structure ou d'une zone de tout type de données déjà défini (élémentaire ou dérivé).

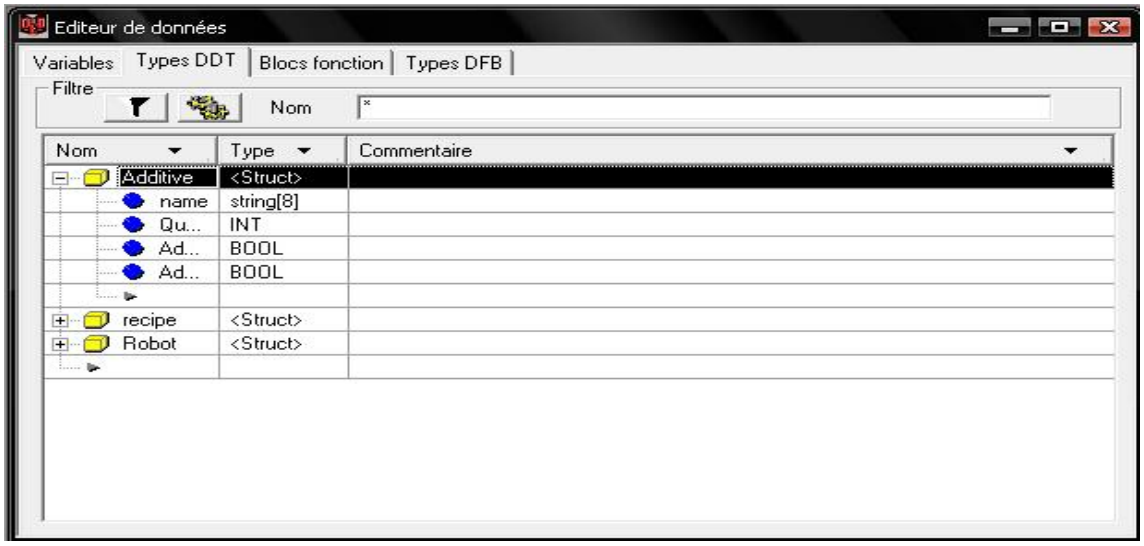


Figure A.37 : *Onglet Types DDT* [9]

Les fonctions disponibles sont les suivantes :

- définition de DDT imbriqués (max. 8 niveaux).

- définition de zones (matrices) ayant jusqu'à six dimensions.
- affectation d'une valeur initiale.
- affectation d'une adresse.
- saisie d'un commentaire.
- analyse du type de données dérivé.
- affectation du type de données dérivé à une bibliothèque.

Blocs fonction :

L'onglet Blocs fonction permet la déclaration d'instance de blocs fonction élémentaires et dérivés (EFB/DFB).

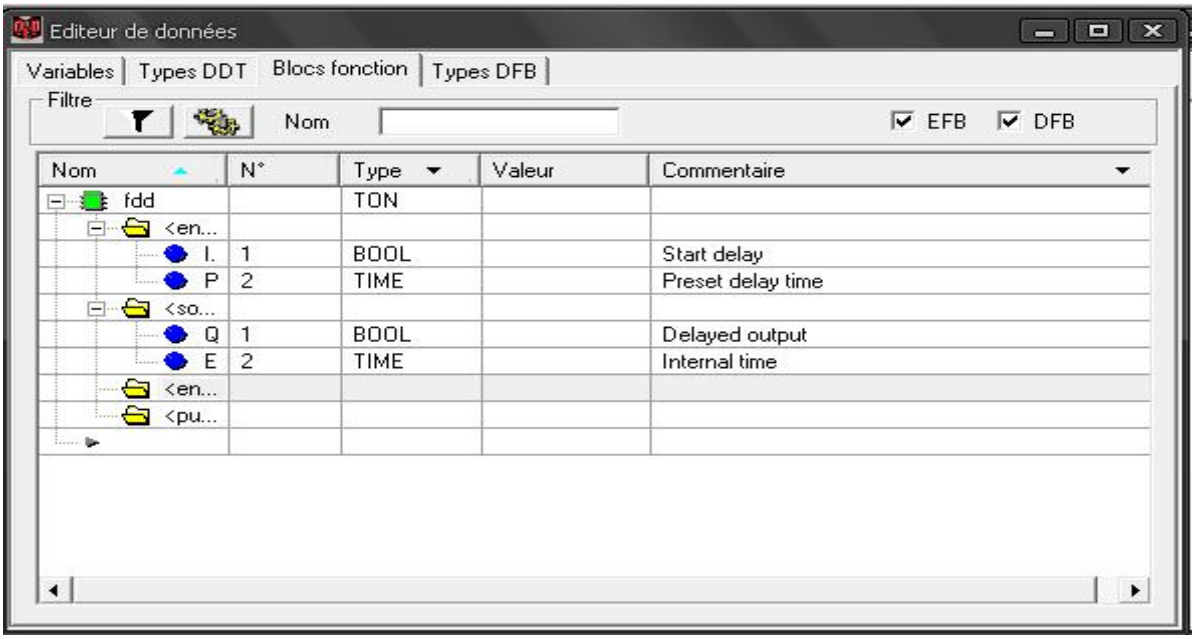


Figure A.38 *: Onglet Blocs fonction* [9]

Les fonctions disponibles sont les suivantes :

- affichage des blocs fonction utilisés dans le projet
- définition d'un symbole pour les blocs fonction utilisés dans le projet
- copie automatique des symboles définis dans le projet
- saisie d'un commentaire relatif aux blocs fonction
- affichage de tous les paramètres (entrées/sorties) des blocs fonction
- affectation d'une valeur initiale aux entrées/sorties de bloc fonction

Types DFB :

L'onglet Types DFB permet de définir les paramètres de blocs fonction dérivés (DFB).

La création de la logique DFB s'effectue directement dans une ou plusieurs sections des langages de programmation FBD, LD ou ST.

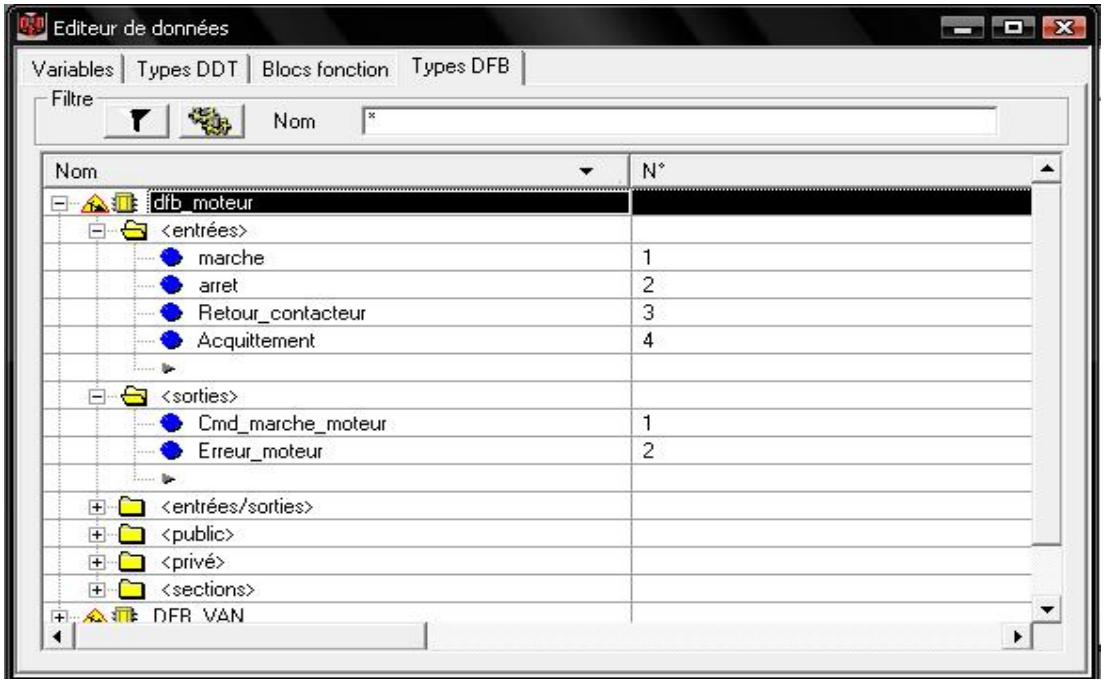


Figure A.39: *Onglet Types DFB* [9]

Les fonctions disponibles sont les suivantes :

- définition du nom DFB
- définition de tous les paramètres du DFB, tels que les :
  - entrées
  - sorties
  - VAR_IN_OUT (entrées/sorties combinées)
  - variables privées
  - variables publiques
- affectation du type de données aux paramètres DFB
- boîte de dialogue de sélection personnalisée pour les types de données dérivés
- affectation d'une valeur initiale
- imbrication de DFB
- utilisation de plusieurs sections dans un DFB
- saisie d'un commentaire relatif aux DFB et paramètres DFB

- analyse des DFB définis
- affectation des DFB définis à une bibliothèque

A.10 Convertisseur PL7

A.10.1 Définition

L'outil de conversion d'applications PL7 est intégré à Unity Pro ; il permet de convertir des applications PL7 en applications Unity Pro. Pour réaliser une conversion, vous devez d'abord effectuer les opérations suivantes :

- Mettre à jour l'application vers PL7 V4.3 ou une version ultérieure ;
- Déverrouiller, si nécessaire, la fonction de protection de l'application, ainsi que celle de l'ensemble des sections, des modules fonctionnels et des blocs fonction dérivés (DFB) ;
- Exporter et enregistrer le fichier source.

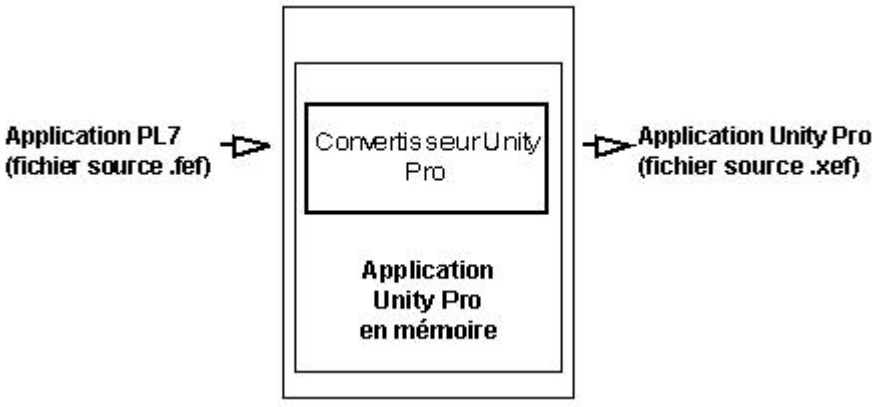
A.10.2 Principe de conversion

Le convertisseur d'applications PL7 permet de convertir :

- Une application PL7 complète (procédure automatique) .
- Un DFB PL7 (procédure semi-automatique).

A.10.2.a Conversion automatique

La procédure permettant de convertir une application PL7 en une application Unity Pro est la suivante :



Le fichier source PL7.fef est converti en un fichier source Unity Pro .xef, puis est importé et analysé automatiquement dans le projet Unity Pro. La phase d'analyse doit être lancée

manuellement afin de détecter toute erreur de conversion et les afficher à l'écran dans une fenêtre de visualisation.

En fin de procédure, l'application PL7 convertie et la fenêtre de visualisation apparaissent à l'écran dans le logiciel Unity Pro. Pour corriger toute erreur de conversion, cliquez sur la ligne d'erreur affichée dans la fenêtre de visualisation pour accéder directement à la partie du programme à modifier.

Procédure de conversion d'une application PL7 en une application Unity Pro

Le tableau ci-dessous décrit la procédure permettant de convertir une application PL7 en une application Unity Pro. [8]

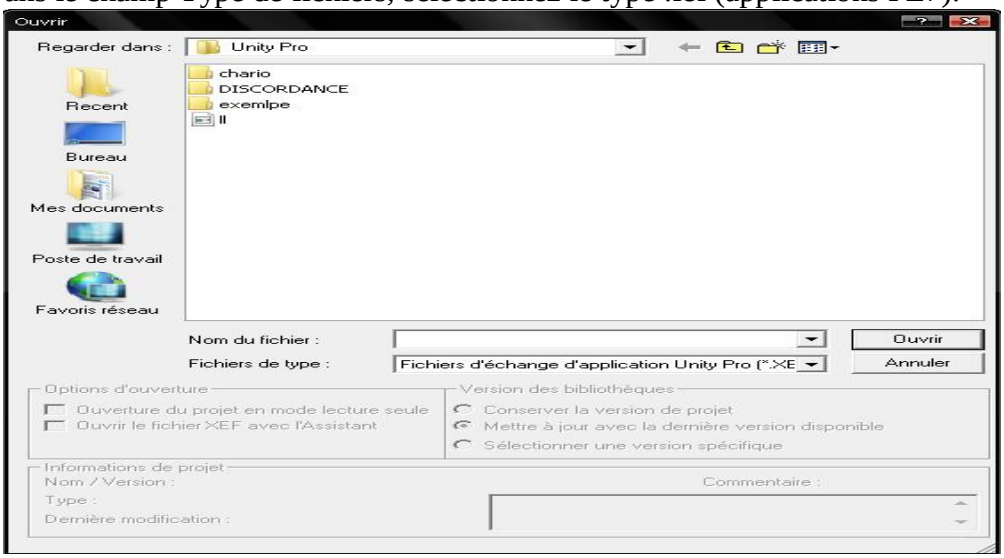
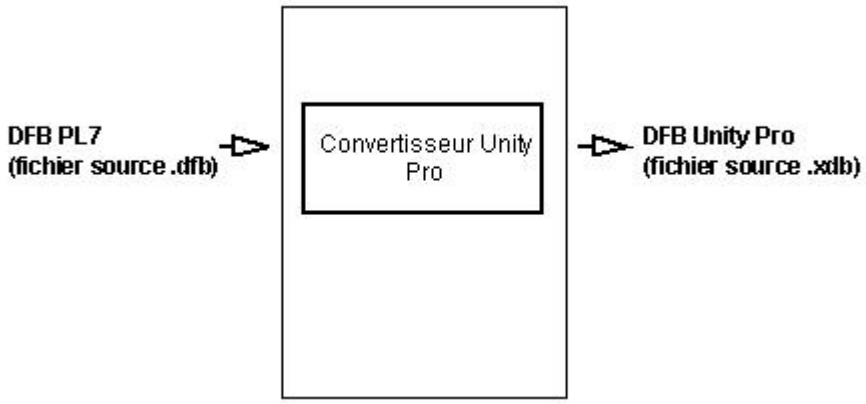
Etape	**Action**
1	Sélectionnez Ouvrir dans le menu Fichier.
2	Dans le champ Type de fichiers, sélectionnez le type .fef (applications PL7). 
3	Sélectionnez le disque dur et/ou le répertoire contenant le fichier à convertir.
4	Sélectionnez le fichier (.fef) à ouvrir (et donc à convertir). Le nom du fichier apparaît alors dans le champ Nom de fichier.
5	Validez en cliquant sur Ouvrir. Résultat : la procédure de conversion est lancée. La barre d'état indique la progression de l'opération.
6	L'importation automatique terminée, vous devez lancer la procédure d'analyse manuellement afin de vérifier la syntaxe de votre application. Remarque : Si, au cours de la phase d'importation ou d'analyse, une fenêtre de visualisation s'affiche, cela signifie que des erreurs se sont produites lors de la conversion. Dans ce cas, corrigez les erreurs

Tableau A.18: *la procédure de conversion du PL7 en Unity Pro* [8]

A.10.2.b Conversion semi-automatique

La procédure permettant de convertir un DFB PL7 est la suivante :

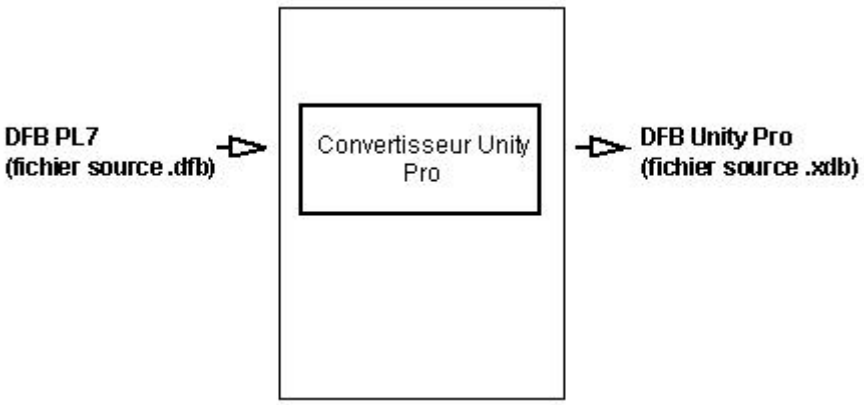


Le fichier source PL7 .dfb est converti en un fichier source Unity Pro .xdb.

En fin de procédure, le DFB PL7 converti est enregistré dans son format source. Afin que Unity Pro puisse utiliser ce DFB, il doit être importé manuellement dans une application Unity Pro.

A la suite de cette importation, vous devez lancer la phase d'analyse du projet manuellement afin de détecter toute erreur de conversion et les afficher à l'écran dans une fenêtre de visualisation

La procédure permettant de convertir un fichier .DAT PL7 est la suivante :



La conversion d'un fichier PL7 .dat en un fichier Unity Pro .dat se fait en ajoutant "_convert" à son nom. En fin de procédure, le fichier .dat PL7 converti est enregistré. Vous pouvez

ensuite l'utiliser dans Unity Pro à l'aide de la commande "Transfert des données du fichier vers l'automate" du menu Automate.

Procédure de conversion

Le tableau ci-dessous décrit la procédure permettant de convertir un DFB PL7 en son équivalent Unity Pro.

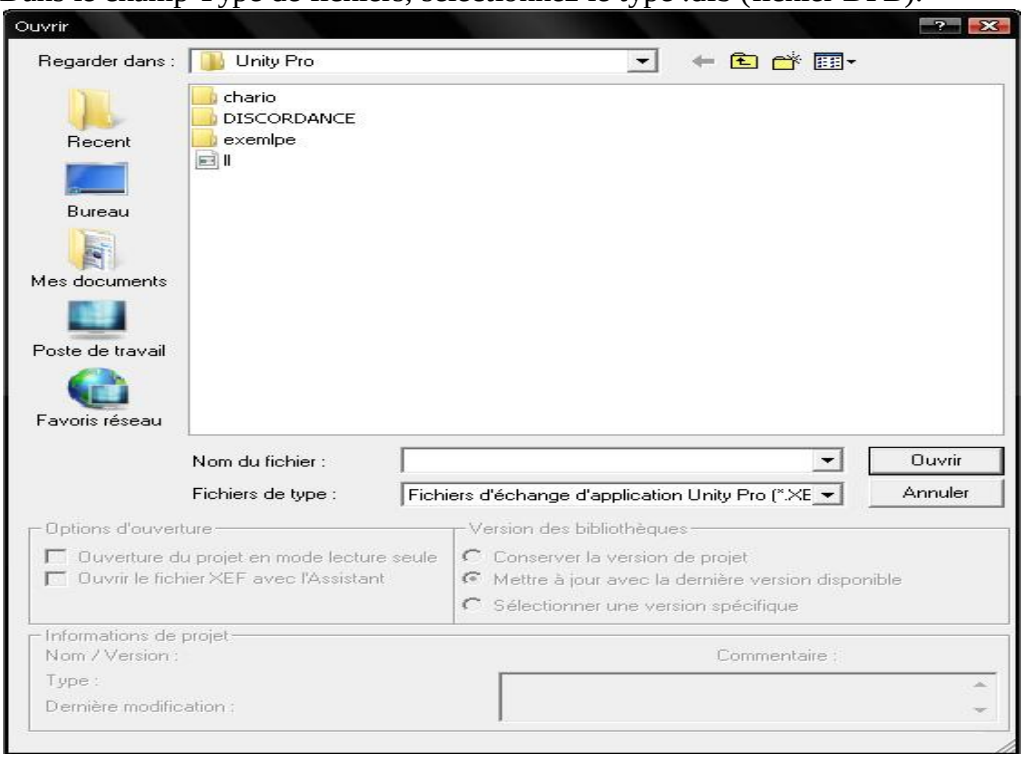
Etape	Action
1	Sélectionnez Ouvrir dans le menu Fichier.
2	Dans le champ Type de fichiers, sélectionnez le type .dfb (fichier DFB). 
3	Sélectionnez le disque dur et/ou le répertoire contenant le fichier à convertir.
4	Sélectionnez le fichier (.dfb) à ouvrir (et donc à convertir). Le nom du fichier apparaît dans le champ Nom de fichier.
5	Validez en cliquant sur Ouvrir.
6	La procédure de conversion est lancée. La barre d'état indique la progression de l'opération.
7	La conversion terminée, une fenêtre s'affiche et vous indique la fin de la procédure.

Tableau A.19: *la procédure de conversion dôun DFB PL7 en son équivalent Unity Pro* [8]

Analyse d'une application PL7 convertie dans Unity Pro

L'analyse permet de détecter les erreurs générées dans l'application au cours de la conversion.

les erreurs peuvent être détectées sont les suivantes :

- Erreurs de syntaxe ;
- Erreurs sémantiques ;
- Parties de programme manquantes ;
- Objets sans équivalent Unity Pro ;
- Objets graphiques ou schémas Grafcet sans équivalent Unity Pro ;
- Autres erreurs (EF développés par l'utilisateur, etc.).

Fenêtre de visualisation

Toutes les erreurs détectées au cours de l'analyse, quel que soit leur type, s'affichent automatiquement dans la fenêtre de visualisation.

Les erreurs nécessitant une correction manuelle sont signalées par le message "Converror".

Ce message, qui s'affiche entre guillemets dans la fenêtre de visualisation, vous permet d'accéder directement, par un double-clic gauche, à la partie du programme qui doit être corrigée .

A.10.3 Différences entre PL7 et Unity Pro

Dans cette section on décrit la correspondance et les différences entre PL7 et UnityPro et lôétat de conversion :

A.10.3.1 Différences entre les différentes structures d'application

A.10.3.1.a les éléments structurels

- **Les tâches, les événements et les SR**

Le tableau suivant décrit la correspondance et les différences éventuelles entre les tâches, les EVT, les SR PL7 et Unity Pro. [8]

	PL7	**Unity Pro**	**Etat**
Tâche MAST	Cyclique ou périodique	Cyclique ou périodique	Converti
Tâche FAST	Périodique	Périodique	Converti
Traitements événementiels : EVTi	Le nombre dôévénements disponibles dépend du	Le nombre dôévénements disponibles dépend du	Converti

	processeur	processeur	
	Les mots système gèrent les événements	Les mots système gèrent les événements	Converti (1)
	MASKEVT UNMASKEVT	MASKEVT UNMASKEVT	Converti (2)
EVTi : niveau de priorité	Gestion du niveau de priorité	Gestion du niveau de priorité	Converti
Sous-programmes : SRi	Sri	SR Section	Modifié (3)
Légende :			
(1)	Les mêmes objets système existent en Unity Pro.		
(2)	Les mêmes EF existent en Unity Pro.		
(3)	Le nom des SR est modifié mais le fonctionnement reste le même. En Unity Pro, le SRi devient une section SR nommée SRi().		

- **Les sections**

Le tableau suivant décrit la correspondance et les différences éventuelles entre les caractéristiques des sections PL7 et Unity Pro.

		PL7	**Unity Pro**	**Etat**
Sections		Oui	Oui	Converti
Condition d'activation		Oui	Oui	Converti
Objets (1)		%Si %Mi %MWi:Xj %SWi:Xj %KWi:Xj %Mi[%MWj] %Mi[%SWj] %Mi[%KWj] ...	Objets équivalents Unity Pro	Modifié (2)
Protection des sections		Ecriture Lecture / Ecriture Aucune	Ecriture Lecture / Ecriture Aucune	Converti
Attributs des sections				
Nom long		16 caractères	≥ 16 caractères	Modifié (3)
Nom court		8 caractères	≥ 8 caractères	Modifié
Commentaire		250 caractères	256 caractères	Modifié
Nombre de sections dans ...	MAST, FAST, AUXi	4096	Sans limitation	Modifié
	EVT	1	1	Converti

	DFB	1	≥ 1	Modifié
	SR	1	1	Converti
Langage		LD, ST, IL	LD, ST, IL	Converti
Légende :				
(1)	Objets qui définissent la condition dôexécution dôune section.			
(2)	Le convertisseur dôapplications PL7 remplace ces objets par leurs équivalents en Unity Pro.			
(3)	Le nom de la section ne peut pas être déjà utilisé pour définir une des variables de lôapplication.			

A.10.3.1.b les modules fonctionnels

La table suivante décrit les correspondances et les différences entre les caractéristiques des modules fonctionnels PL7 et celles des modules fonctionnels Unity Pro.

			PL7	**Unity Pro**	**Etat**
Imbrication de modules fonctionnels			Aucune limitation	Aucune limitation	Converti
Commentaire			0 à 127 caractères	0 à 255 caractères	Modifié
Taille du fichier de description			Aucune limitation	Aucune limitation	Converti
Capacité maximale d'un module fonctionnel	Nombre de modules fonctionnels		Aucune limitation	Aucune limitation	Converti
	Nombre de sections	LD, ST, IL	Aucune limitation	Aucune limitation	Converti
		Grafcet	1	Aucune limitation	Modifié (1)
	Nombre d'événements dans une section		Aucune limitation	Aucune limitation	Converti (2)
	Nombre de macroétapes :		Limité par le processeur	Aucune	Supprimé (1)
	Nombre de tables d'animation		Aucune limitation	Aucune limitation	Converti
	Nombre d'écrans d'exploitation		Aucune limitation	Aucune limitation	Converti
Légende :					
(1)	Le convertisseur d'applications PL7 ne convertit pas tous les types de modules fonctionnels.				

(2)	Les langages de programmation possibles sont les suivants : LD, ST, I

A.10.3.2 Différences entre PL7 et Unity Pro : types et tables

A.10.3.2.a Types

La table suivante décrit les correspondances et les différences entre les types PL7 et les types Unity Pro.

	PL7	**Unity Pro**	**Etat**
Type	BOOL	BOOL	Converti
	EBOOL	EBOOL	Converti
	WORD	INT	Modifié (1)
	DWORD	DINT	Modifié (1)
	REAL	REAL	Converti
Légende :			
(1)	Les types WORD et DWORD sont convertis en types INT et DINT.		

A.10.3.2.b Tableaux

La table suivante décrit les correspondances et les différences entre les tableaux PL7 et les tableaux Unity Pro. [8]

	PL7	**Unity Pro**	**Etat**
Tableau	Tableau booléen (EBOOL) %Mi:n	ARRAY [0..n-1] OF EBOOL	Modifié (1)
	Tableau de mots (WORD) %MWi:n	ARRAY [0..n-1] OF EBOOL	Modifié (1)
	Tableau de mots doubles (DWORD) %MDi:n	ARRAY [0..n-1] OF DINT	Modifié (1)
	Tableau de nombres à virgule flottante (REAL) %MFi:n	ARRAY [0..n-1] OF REAL	Modifié (1)
	Tableau d'octets %MBi:n	STRING [n]	Modifié (1)
Légende :			
(1)	Le convertisseur modifie la déclaration.		

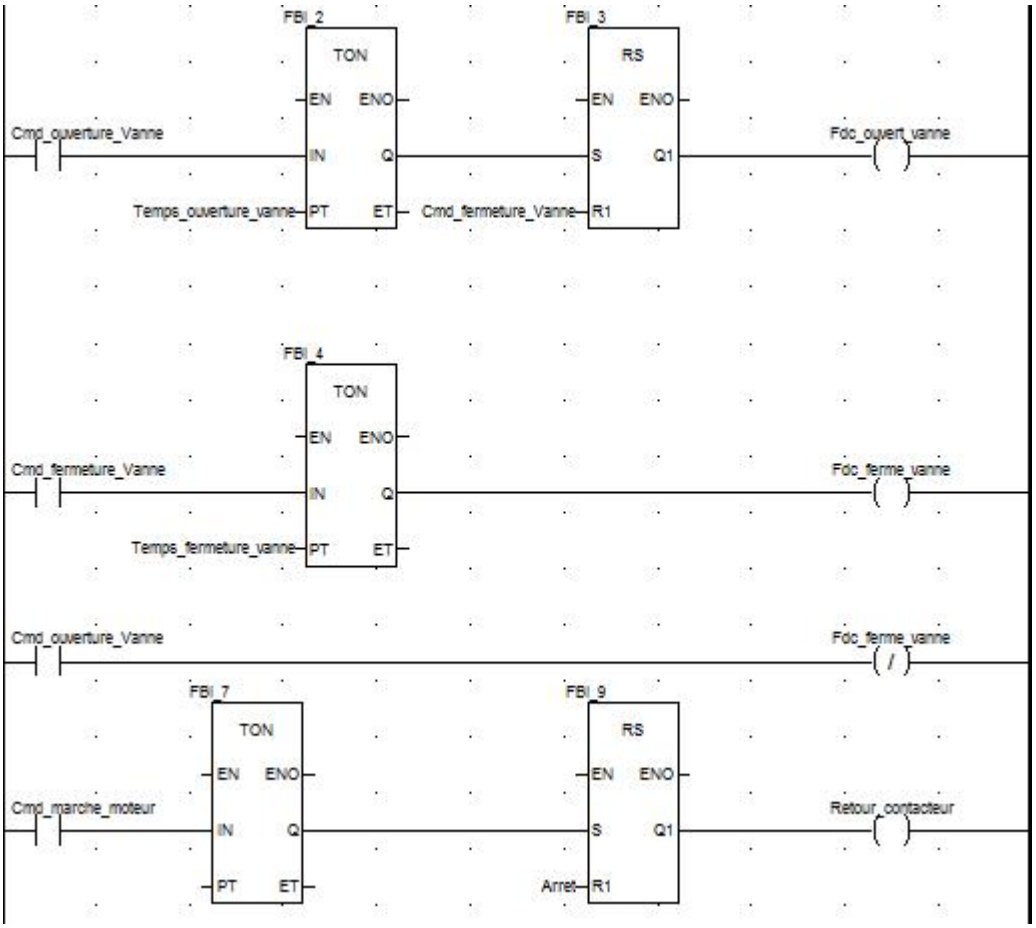
ANNEXE B

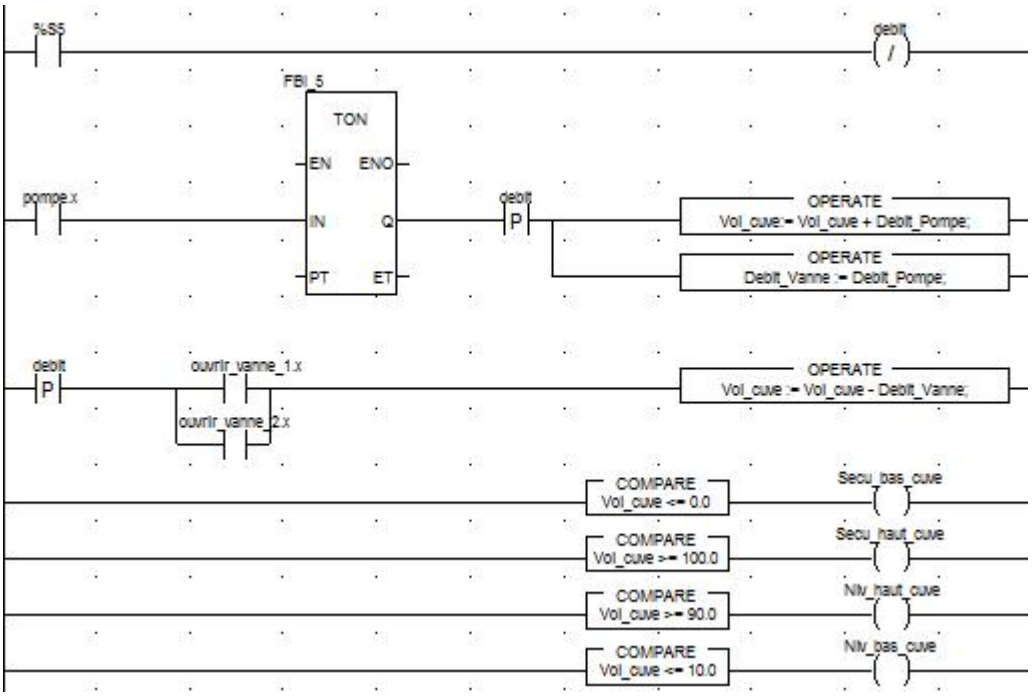
La simulation et visualisation de diagnostique de lôapplication UNITY PRO « Mise en ouvre dôune application»

B.1 Création du programme en LD pour la simulation de l'application

Cette section sert uniquement pour la simulation de l'application. Elle n'est donc pas à utiliser dans le cas d'une connexion à un automate.

Illustration de la section Simulation





Description de la section Simulation

- la première ligne sert à simuler la valeur de la variable Fdc_ouvert_vanne. Si on commande lôouverture (Cmd_ouverture_vanne = 1) on déclenche un temporisateur TON. Lorsque le temps PT est atteint la sortie du TON passe à "1" et fait monter à "1" la sortie Fdc_ouvert_vanne sauf si on commande en même temps la fermeture de la vanne.
- même principe pour les sortie Fdc_ferme_vanne et Retour_contacteur.
- la dernière partie de la section sert à la simulation du niveau de la cuve et au déclenchement des différents niveaux. Pour cela, on utilise des blocs OPERATE et COMPARE disponible dans la bibliothèque. [8]

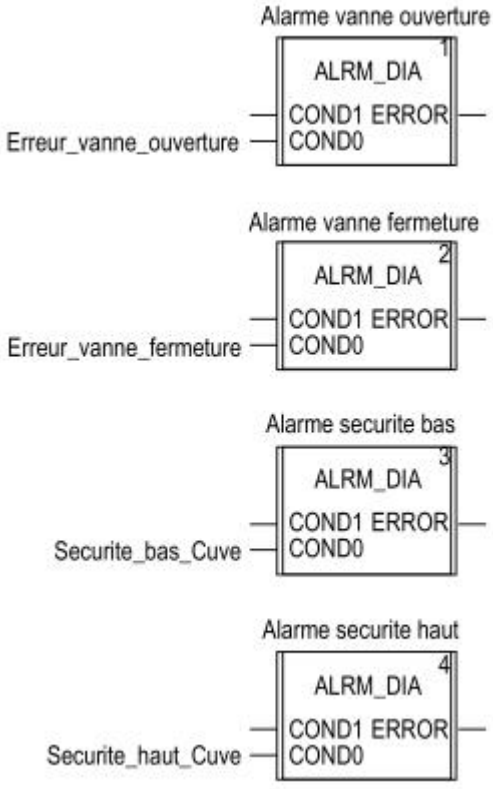
B.2 Création du programme en FBD pour le diagnostic de lôapplication

Cette section est utilisée pour déclarer les variables qui seront remontées dans le viewer de diagnostic en cas dôerreur.

Illustration de la section Diagnostic

Lôécran ci-dessous représente la section FBD utilisant les Blocs fonction (Voir Illustration des blocs fonction utilisés par lôapplication) Alarme_securite_bas, Alarme_securite_haut et erreur_vanne :

Description de la section Diagnostic



Le principe de cette section est basé sur lôutilisation des blocs fonction ALMR_DIA. Dans tous les blocs, on surveille le changement dôétat de la variable dôentrée. Les entrées étant toujours connectées à COND0, le déclenchement de lôaffichage dans la fenêtre du Viewer de diagnostic se fera lors dôun passage à 1 de la variable dôentrée.

B.3 Marche à suivre pour créer la section FBD

1° Dans le Navigateur de projet\Programme\Tâches, double-cliquez sur MAST.

2° Faîtes un clic droit sur Section puis choisissez Nouvelle section. Donnez le nom Diagnostic à cette section puis choisissez le langage FBD. La fenêtre dôédition sôouvre.

3° Pour utiliser le bloc fonction de type ALRM_DIA créé il faut lôinstancier. Faites un clic droit dans lôéditeur puis cliquez sur Sélection de ... données et sur . Cliquez sur lôonglet Blocs fonction et sélectionnez votre bloc puis validez par OK et placez-le dans lôéditeur FBD. Pour affecter une variable à une entrée ou une ... sortie, double-cliquez dessus, cliquez sur et dans lôonglet Variable choisissez votre variable.

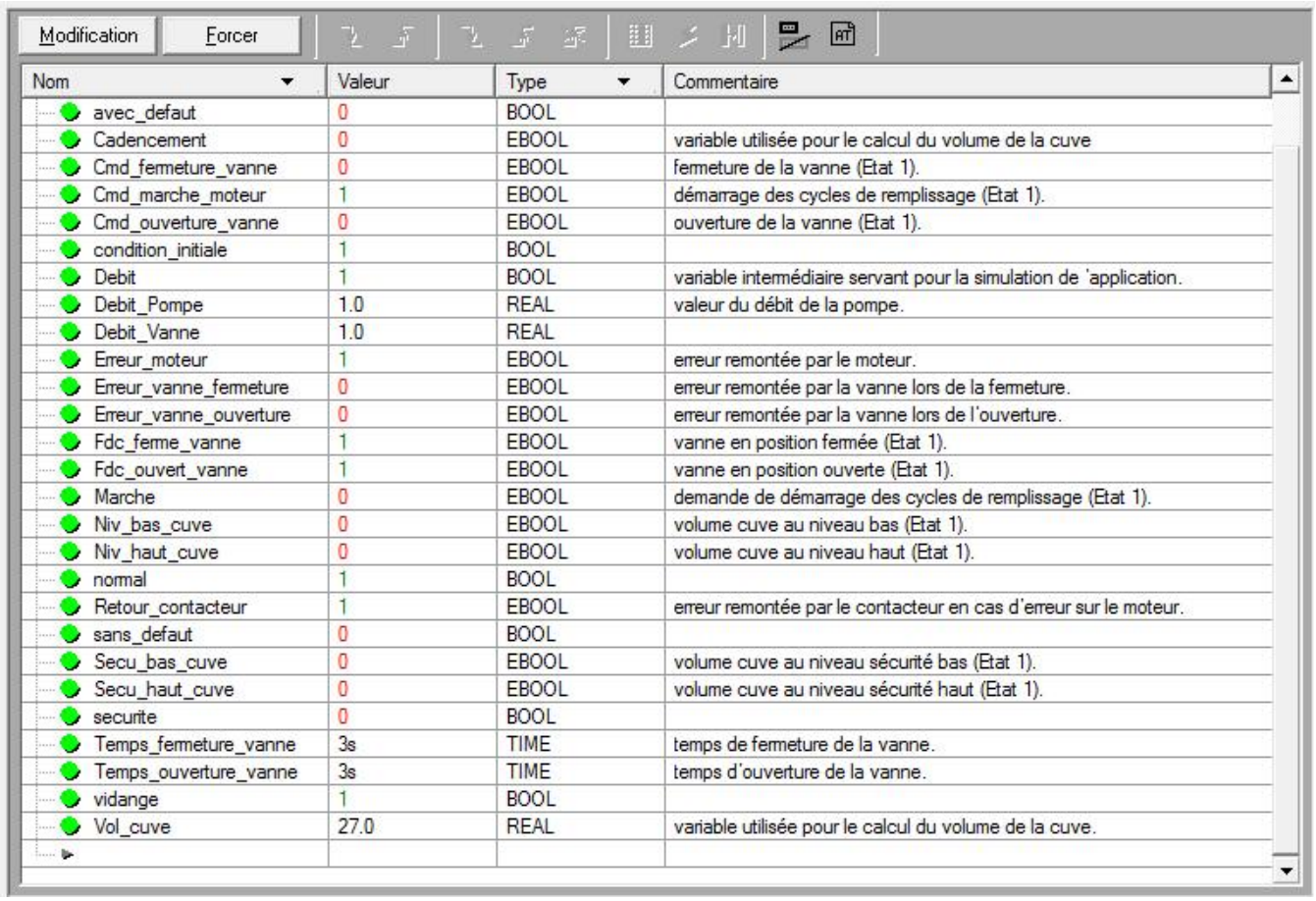
B.4 Création de la table dôanimation

La table dôanimation est utilisée pour surveiller des valeurs de variables, modifier et/ ou forcer des valeurs. Seules les variables déclarées dans Variables et instances FB peuvent être ajoutées dans la table dôanimation.

Marche à suivre pour créer la table dôanimation

1° Dans le Navigateur de projet, faites un clic droit sur Tables dôanimation, la fenêtre dôédition sôouvre.

2° Cliquez dans la première cellule de la colonne nom puis sur le bouton et rajoutez les variables de votre choix.

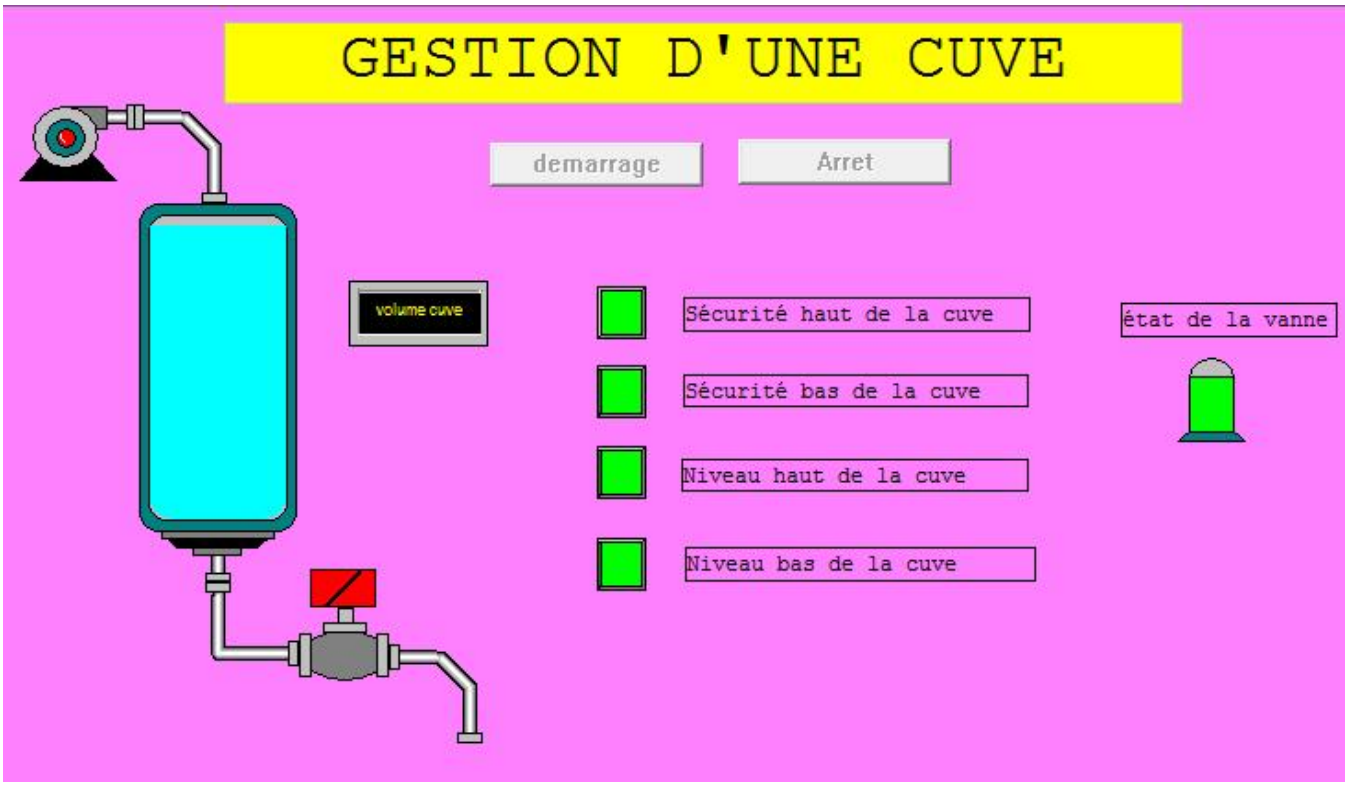


Modification | Forcer

Nom	Valeur	Type	Commentaire
avec_defaut	0	BOOL	
Cadencement	0	EBOOL	variable utilisée pour le calcul du volume de la cuve
Cmd_fermeture_vanne	0	EBOOL	fermeture de la vanne (Etat 1).
Cmd_marche_moteur	1	EBOOL	démarrage des cycles de remplissage (Etat 1).
Cmd_ouverture_vanne	0	EBOOL	ouverture de la vanne (Etat 1).
condition_initiale	1	BOOL	
Debit	1	BOOL	variable intermédiaire servant pour la simulation de 'application.
Debit_Pompe	1.0	REAL	valeur du débit de la pompe.
Debit_Vanne	1.0	REAL	
Erreur_moteur	1	EBOOL	erreur remontée par le moteur.
Erreur_vanne_fermeture	0	EBOOL	erreur remontée par la vanne lors de la fermeture.
Erreur_vanne_ouverture	0	EBOOL	erreur remontée par la vanne lors de l'ouverture.
Fdc_ferme_vanne	1	EBOOL	vanne en position fermée (Etat 1).
Fdc_ouvert_vanne	1	EBOOL	vanne en position ouverte (Etat 1).
Marche	0	EBOOL	demande de démarrage des cycles de remplissage (Etat 1).
Niv_bas_cuve	0	EBOOL	volume cuve au niveau bas (Etat 1).
Niv_haut_cuve	0	EBOOL	volume cuve au niveau haut (Etat 1).
normal	1	BOOL	
Retour_contacteur	1	EBOOL	erreur remontée par le contacteur en cas d'erreur sur le moteur.
sans_defaut	0	BOOL	
Secu_bas_cuve	0	EBOOL	volume cuve au niveau sécurité bas (Etat 1).
Secu_haut_cuve	0	EBOOL	volume cuve au niveau sécurité haut (Etat 1).
securite	0	BOOL	
Temps_fermeture_vanne	3s	TIME	temps de fermeture de la vanne.
Temps_ouverture_vanne	3s	TIME	temps d'ouverture de la vanne.
vidange	1	BOOL	
Vol_cuve	27.0	REAL	variable utilisée pour le calcul du volume de la cuve.

B.5 Création de lôécran dôexploitation

Lôécran dôexploitation est utilisé pour animer des objets graphiques symbolisant lôapplication. Ces objets peuvent appartenir à la bibliothèque dôUnity Pro ou ils peuvent être créés à lôaide de lôéditeur graphique.



Marche à suivre pour créer lôécran dôexploitation

Pour créer le bouton Démarrage_cycle :

1°Dans le Navigateur de projet, faites un clic droit sur Ecrans dôexploitation et cliquez sur

Nouvel Ecran. Lôéditeur dôécran dôexploitation apparaît.

2° cliquez sur le bouton et placez-le dans lôéditeur de lôécran dôexploitation. Double-cliquez sur le

bouton et dans lôonglet Pilotage, sélectionnez la variable Marche en cliquant sur le bouton et validez par OK, puis entrez le nom du bouton dans la zone Texte. Le bouton est à présent affecté à la variable Marche .

Pour insérer et animer la cuve :

1° Dans le Navigateur de projet, faites un clic droit sur Ecrans dôexploitation et cliquez sur Nouvel Ecran. Lôéditeur dôécran dôexploitation apparaît.

2° cliquez sur le bouton et placez-le dans lôéditeur de lôécran dôexploitation. Double-cliquez sur le bouton et dans lôonglet Pilotage, sélectionnez la variable Marche en cliquant sur le

bouton et validez par OK, puis entrez le nom du bouton dans la zone Texte. Le bouton est à présent affecté à la variable Marche.

pour insérer et animer la cuve :

1° Dans le Navigateur de projet, faites un clic droit sur Ecrans dôexploitation et cliquez sur Nouvel Ecran. Lôéditeur dôécran dôexploitation apparaît.

2° dans le menu Outils, sélectionnez Bibliothèque des écrans dôexploitation. La fenêtre sôouvre, double-cliquez sur Fluides puis sur Cuve. Sélectionnez la cuve dynamique de lôécran dôexploitation, et faîtes un Copier (Ctrl + C) puis Coller (Ctrl + V) dans le dessin dans lôéditeur de lôécran dôexploitation (pour revenir sur votre écran, cliquez sur le menu Fenêtre puis Ecran).

- la cuve est maintenant dans votre écran dôexploitation. Il faut maintenant une variable pour animer le niveau. Dans le menu Outils, cliquez sur Fenêtre des variables. La fenêtre apparaît sur la gauche et dans la colonne Nom on trouve le mot %MW0. Pour avoir la partie animée de lôobjet graphique (ici la cuve), il faut double-cliquer sur %MW0. Une partie de la cuve est sélectionnée, faîtes un clic droit sur cette partie puis cliquez sur Caractéristiques. Sélectionnez lôonglet Animation et entrez la variable concernée en cliquant sur le bouton  (à la place de %MW0). Dans notre application se sera Vol_cuve.
- il faut définir les minimum et maximum de la cuve. Dans lôonglet Type dôanimation, cliquez sur Bargraphe puis sur le bouton  et rentrez les champs en fonction de la cuve.
- validez par appliquer et OK

pour insérer et animer la vanne :

1° Dans le Navigateur de projet, faites un clic droit sur Ecrans dôexploitation et cliquez sur Nouvel Ecran. Lôéditeur dôécran dôexploitation apparaît.

2° Dans le menu Outils, sélectionnez Bibliothèque des écrans dôexploitation. La fenêtre sôouvre, double-cliquez sur Actionneurs puis sur Vanne. Sélectionnez une vanne dynamique (de lôécran dôexploitation) et faîtes un Copier (Ctrl + C) puis Coller (Ctrl + V) dans le dessin dans lôéditeur de lôécran (pour revenir sur votre écran, cliquez sur le menu Fenêtre puis Ecran).

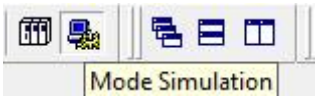
- Sélectionnez la vanne, faîtes un clic droit dessus et cliquez sur Dissocier, sélectionnez le rectangle rouge et déplacez le de manière à voir lôautre rectangle vert dessous. Double-cliquez sur le rectangle vert, cliquez sur lôonglet Animation et ajoutez la variable Cmd_ouverture_vanne. Toujours dans la fenêtre Propriétés de lôobjet, dans la zone Condition dôaffichage, sélectionnez Bit = 1. Ce paramétrage rends visible le rectangle vert lorsque le bit %M2 = 1 sinon ce rectangle est invisible.
- Même procédure pour le rectangle rouge, mais avec une condition dôaffichage Bit = 0. Si lôanimation ne fonctionne pas, mettez en arrière-plan le rectangle se trouvant au premier plan.

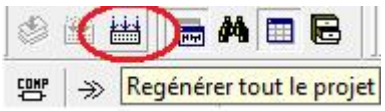
B.6 Mise en route de lôapplication

B.6.1 Exécution de lôapplication en mode simulation

Il est possible de vous connecter au simulateur dôAPI qui permet de tester une application sans raccordement à lôautomate et autres matériels.

Exécution de lôapplication

1° cliquez sur lôicône Mode Simulation, 

2 °Dans le menu Génération, cliquez sur Regénérer tout le projet. 

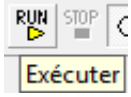
Votre projet est généré et prêt à être transféré sur le simulateur. Lorsque vous générez le projet, vous apercevez la fenêtre de résultats. En cas de dôerreur dans le programme, Unity Pro indique lôemplacement en double-cliquant sur la phrase surligné.

3° cliquez sur lôicône Connexion. 

Vous êtes maintenant connecté au simulateur.

4° cliquez sur Transférer le projet vers lôautomate,. 

Lôapplication est transférée dans le simulateur dôautomate.

5° cliquez sur Exécuter,. 

Lôapplication est en cours dôexécution (mode RUN) dans le simulateur.

B.6.2 Exécution de lôapplication en mode standard

Le mode standard impose lôutilisation dôun automate et de modules dôE/S TOR et ANA pour affecter les sorties aux différents capteurs et actionneurs. Les variables utilisées dans le mode

simulation doivent être modifiées. En effet, en mode standard , les variables sont obligatoirement localisées afin dôêtre associées à des E/S physiques.

Configuration matérielle de lôapplication

1° Dans la fenêtre Navigateur de projet double-cliquez sur Configuration puis sur 0:Bus X et sur 0:TSX RKY ÅÅÅ(0 étant le numéro du rack).

2° Dans le fenêtre Bus X choisissez un emplacement, par exemple 3 et double-cliquez dessus.

3° Insérez un module dôentrées TOR, par exemple TSX DEY 16A5.

4° Validez par OK. Ce module dôentrée est utilisé pour câbler les entrées de type EBOOL de lôapplication.

Affectation des variables au module dôentrées

1° Dans la fenêtre Navigateur de projet et dans Variables et instances FB double-cliquez sur Variables élémentaires.

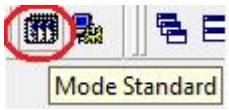
2° Dans la colonne Adresse entrez lôadresse Rack\Module\Voie\Donnée associée a la variable.

Exemple : Dans le module TSX DEY 16A5, il y a 2 voies, la voie 0 et la voie 8. La voie 0 comprend les entrées 0 à 7 et la voie 8 les entrées 8 à 15. Si on câble sur lôentrée 0 du module la sortie du fin de course fermeture vanne, on obtient dans la colonne adresse de lôéditeur de la variable Fdc_fermeture_vanne lôadresse %I0.3.0.0 ; Illustration :

3° Procédez de la même manière pour toutes les variables localisées.

Exécution de lôapplication

1° cliquez sur Mode Standard,



2 ° cliquez sur Regénérer tout le projet. Votre projet est généré et prêt à être transféré sur lôautomate. Lorsque vous générez le projet, vous apercevez la fenêtre de résultats. En cas de dôerreur dans le programme, Unity Pro indique lôemplacement en cliquant sur la phrase surligné.

3° Dans le menu Automate, cliquez sur Connexion. Vous êtes maintenant connecté à lôautomate.

4° Dans le menu Automate, cliquez sur Transférer le projet vers lôautomate, la fenêtre Transfert du projet vers lôautomate sôouvre, cliquez sur Transférer. Lôapplication est transférée dans lôautomate.

5° Dans le menu Automate, cliquez sur Exécuter, la fenêtre Exécuter sôouvre, cliquez sur OK. Lôapplication est en cours dôexécution (mode RUN) dans lôautomate.

B.7 Viewer de diagnostic

La visualisation du diagnostic permet de surveiller des variables lorsque celles-ci sont associées à des blocs fonction de type diagnostic (ALMR_DIA par exemple).

Création du diagnostic

1° Dans le menu Outils, cliquez sur Viewer de diagnostic, la fenêtre sôaffiche à lôécran.

2° Dés que les variables Secu_bas_cuve ou Secu_haut_cuve ou Erreur_ouverture_vanne ou Erreur_fermeture_vanne passeront de 0 à 1, un message apparaîtra dans le diagnostic.

ANNEXE C

Description des principaux outilles du Logitiel Vijeo Designer

---

Annexe C .

C.1 Système requis pour installation de Vijeo Designer et Vijeo Designer Runtime

	Vijeo-Designer (environnement de développement)	**Vijeo-Designer Runtime (environnement d'exécution)**
Plate-forme	PC	XBTG, XBTGT, Compact iPC, Smart iPC
Processeur	Intel Celeron 566MHz ou supérieur (Pentium III 1 GHz ou supérieur recommandé)	Reportez-vous au manuel de l'utilisateur de votre machine cible XBTG, XBTGT, Compact iPC, ou Smart iPC.
Mémoire	128 Mo minimum (512 Mo minimum recommandés)	
Espace disque disponible	400 Mo minimum sur le disque dur	
Système d'exploitation	Microsoft Windows2000 (SP4 ou plus récent) ou Microsoft Windows XP (SP2 ou plus récent	
Navigateur Internet	Microsoft Internet Explorer 5.0 ou version ultérieure	

Tableau C.1 : *Tableau des exigences système : Vijeo-Designer et Vijeo-Designer runtime*[11]

C.2 Machines cibles prises en charge

Les Machines cibles représentent des interfaces homme-machine (IHM) qui exécutent des applications utilisateur transférées depuis Vijeo-Designer.

Machine cible	Abréviation	Série	Modèle
Magelis iPC Series	iPC	Compact iPC*1	Compact MPC KT52 NAX (1024x768)
			Compact MPC KT22 NAX (1024x768)
		Smart iPC*1	Smart MPC ST52 NDJ (1024x768)
			Smart MPC ST21 NAJ (800x600)
Magelis XBTGT Series	XBTGT, XBTGT2000 Series ou plus	XBTGT7000 Series	XBTGT7340 (1024x768)
		XBTGT6000 Series	XBTGT6340 (800x600)
			XBTGT6330 (800x600)
		XBTGT5000 Series	XBTGT5340 (640x480)
			XBTGT5330 (640x480)
			XBTGT5230 (640x480)
		XBTGT4000 Series	XBTGT4340 (640x480)
			XBTGT4330 (640x480)
			XBTGT4230 (640x480)
		XBTGT2000 Series	XBTGT2330 (320x240)
			XBTGT2220 (320x240)
			XBTGT2130 (320x240)

			XBTGT2120 (320x240)
			XBTGT2110 (320x240)
	XBTGT, XBTGT1000 Series	XBTGT1000 Series	XBTGT1100 (320x240)
			XBTGT1130 (320x240)
Magelis XBTG Series	XBTG	XBTG Series	XBTG6330 (800x600)
			XBTG5330 (640x480)
			XBTG5230 (640x480)
			XBTG4330 (640x480)
			XBTG4320 (640x480)
			XBTG2330 (320x240)
			XBTG2220 (320x240)
			XBTG2130 (320x240)
			XBTG2120 (320x240)
			XBTG2110 (320x240)

Tableau C.2 : *Tableau des machines cibles prises en charge par Vijeo-Designer Runtime*[11]

Vijeo-Designer propose six types d'outil dont les icônes apparaissent dans la barre d'outils. Pour afficher ou masquer un outil, cliquez sur l'icône correspondante

Icône	Outil	Description
	Navigateur	Affiche les informations du projet sélectionné sous forme d'arborescence. Cet outil s'utilise essentiellement pendant le développement d'un projet. Vous pouvez définir les paramètres de la machine cible, de l'équipement, des opérations de transfert, des alarmes et des variables.
	Inspecteur de propriétés	Affiche les propriétés de l'objet sélectionné et permet de les modifier. Si vous sélectionnez plusieurs objets simultanément, la fenêtre d'outils n'affiche que les paramètres qui leur sont communs.
	Informations	Affiche les rapports lorsque la fonctionnalité Rapports est activée. Permet également de parcourir le Web. Pour accéder à des sites Web : 1. Cliquez sur  dans la barre d'outils.  2. Dans la boîte de dialogue qui s'affiche, saisissez une

		adresse URL ou sélectionnez-en une dans la liste. 
	Bibliothèque d'objets	Affiche les objets et les graphiques. Pour utiliser des objets, faites-les glisser depuis la Bibliothèque d'objets vers un écran. Vous pouvez aussi utiliser cette bibliothèque pour stocker les objets que vous créez, par exemple, les graphiques, les écrans, les scripts, les groupes d'alarmes et les chaînes de caractères. Il est ensuite possible d'importer ou exporter l'intégralité des dossiers de la Bibliothèque d'objets afin de les partager entre les concepteurs.
	Liste des graphiques	Affiche la liste des graphiques dans l'écran actif avec les informations suivantes : ordre des dessins, nom de l'objet, coordonnées x et y, animation et variables. Un objet sélectionné dans l'écran est mis en surbrillance dans la Liste des graphiques. Lorsque des objets sont associés, la liste affiche le groupe puis les objets qui le constituent. Vous pouvez trier la Liste des graphiques en cliquant sur l'une des colonnes situées dans le haut de la fenêtre.
	Compte-rendu	Affiche les messages d'erreur et indique la progression des opérations de validation, génération et transfert. En cas de problème, les erreurs s'affichent en rouge et les avertissements en jaune. Pour accéder à l'emplacement de l'erreur, appuyez sur F4 ou cliquez deux fois sur le message d'erreur..

Tableau C.3 : *Principaux outils de Vijeo Designer* [11]

C.3 Manipulation des cibles

Ajout, suppression, modification ou copie de cibles

Vous avez la possibilité d'ajouter, supprimer, modifier ou copier des machines cibles pour un ou plusieurs projets; et au sein d'un projet, vous pouvez créer des applications pour une ou plusieurs machines cibles. Le fait de créer plusieurs applications utilisateur dans un même projet permet d'autoriser l'accès aux données par toutes ces applications. [11]

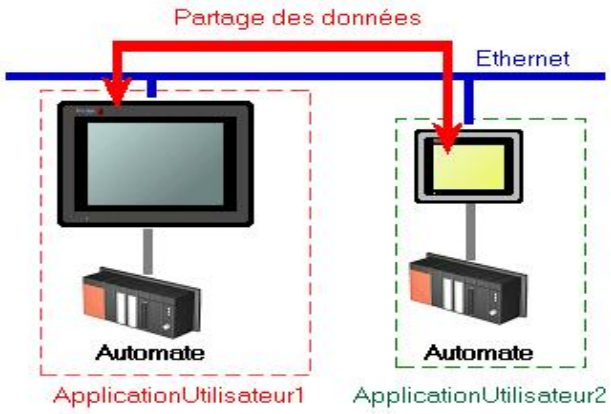
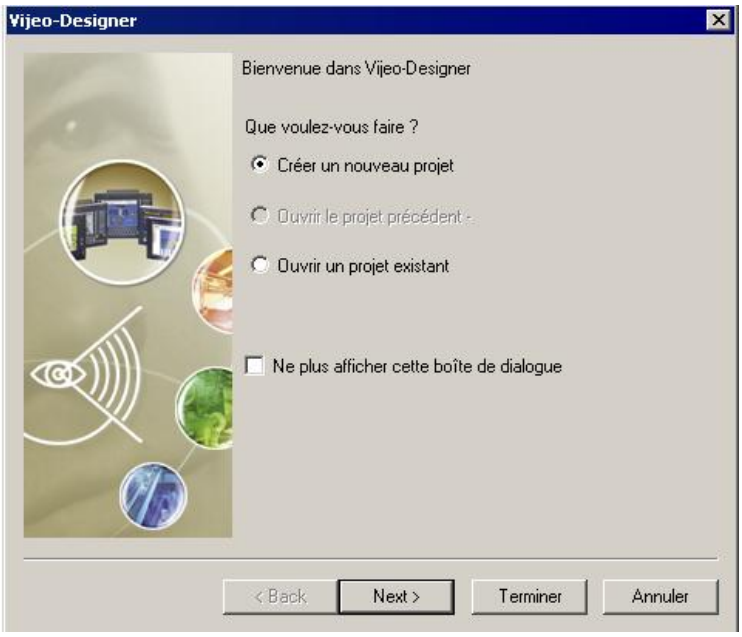


Figure C.1: *partage de données sur plusieurs machines cibles* [11]

C.4 création dûun projet

Les étapes de création dûun projet par l'assistant Vijeo-Designer sont les suinants.

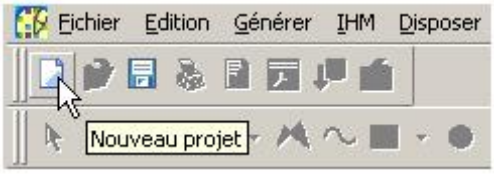
1. Pour créer un projet, sélectionnez Créer un projet et cliquez sur Suivant.



on peut créer un projet dans Vijeo-Designer par :

- Dans l'onglet Vijeo-Manager de la fenêtre du Navigateur, cliquez avec le bouton droit de la souris sur Vijeo-Manager, puis sur Nouveau projet

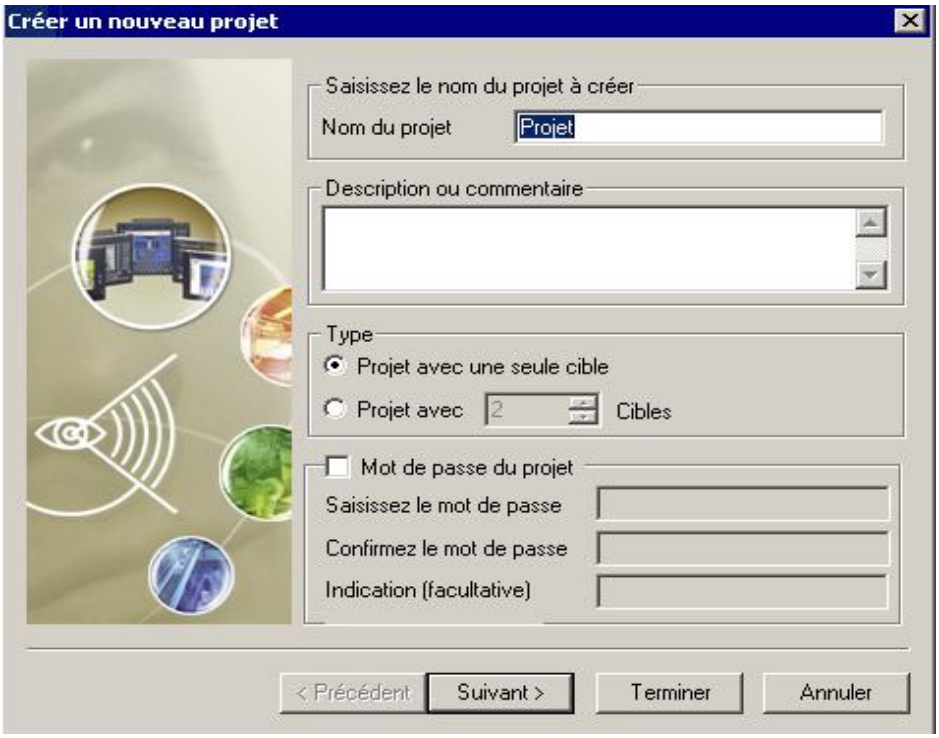
Cliquez sur l'icône Nouveau projet



Dans le menu Fichier, cliquez sur Nouveau projet

2. Dans la boîte de dialogue Créer un nouveau projet, configurez les champs suivants :

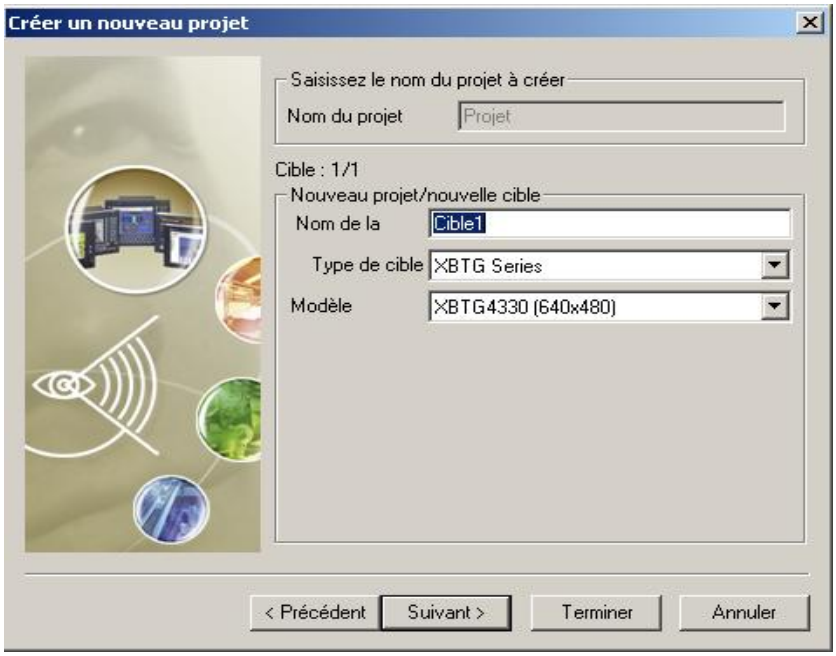
- Nom du projet : Saisissez un nom pour le projet
- Description ou commentaire : Saisissez la description du projet, si nécessaire. (Ce champ est limité à 255 caractères.)
- Type : Indiquez si votre projet dispose d'une ou de plusieurs cibles. Si vous avez plusieurs cibles, indiquez le nombre.
- Mot de passe du projet : Sélectionnez si votre projet requiert une sécurité de projet. Si la sécurité est activée, saisissez un mot de passe et une indication de mot de passe.



3. Cliquez sur Suivant.

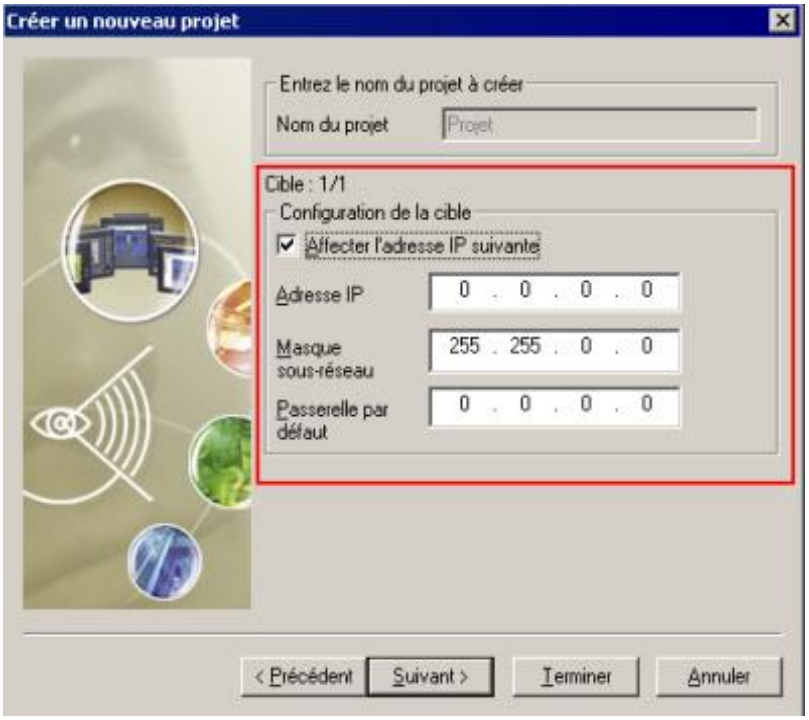
4. Configurez les champs suivants pour cette boîte de dialogue :

- Nom de la cible : Saisissez un nom pour votre cible.
- Type de cible : Sélectionnez le type de cible à partir de la liste déroulante de types de cible.
- Modèle : Sélectionnez votre modèle de cible à partir de la liste déroulante de cibles.



5. Cliquez sur Suivant.

6. Saisissez l'adresse IP de la machine cible et cliquez sur Suivant

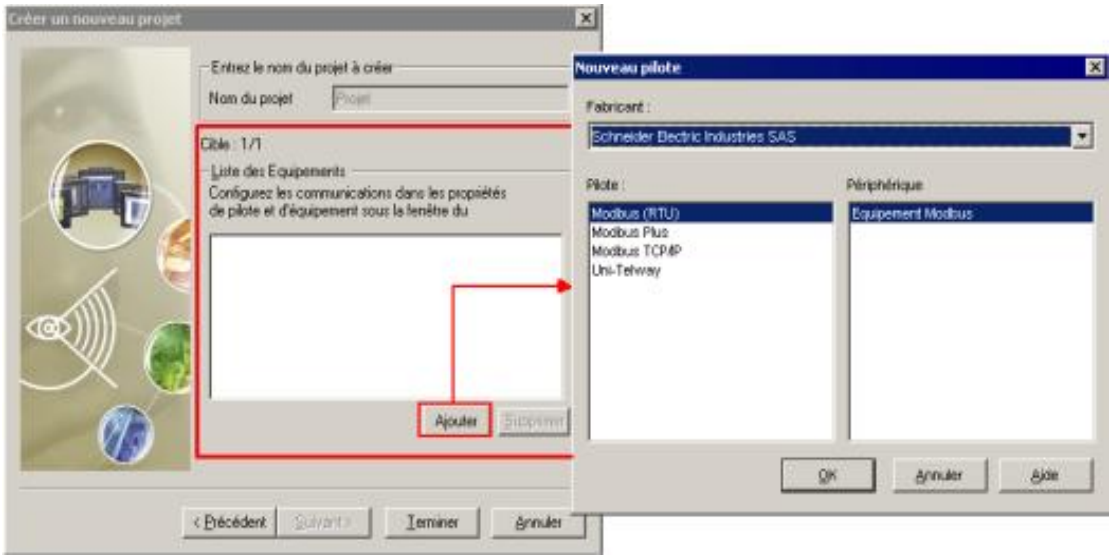


7. Sélectionnez Ajouter un équipement. Cliquez sur Ajouter pour ouvrir la boîte de dialogue Nouveau pilote. Sélectionnez l'équipement que vous souhaitez ajouter, puis cliquez sur Terminer.

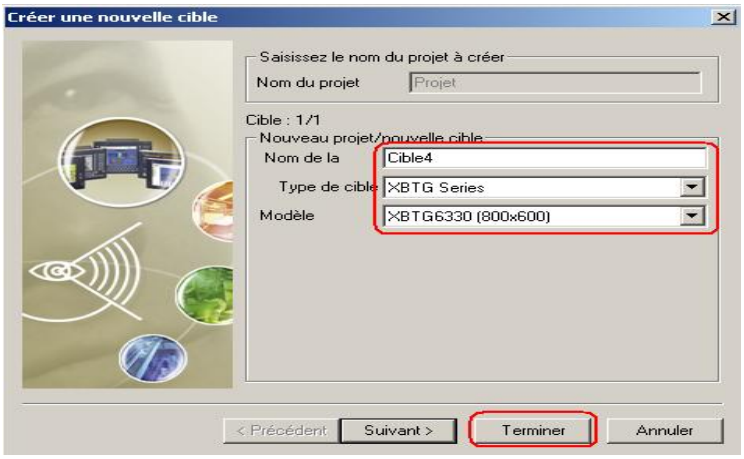
C.5 manipulation des cibles

C.5.1 Ajout d'une cible

1. Dans l'onglet Projet de la fenêtre du navigateur, cliquez à droite sur Projet et sélectionnez Nouvelle cible**.**

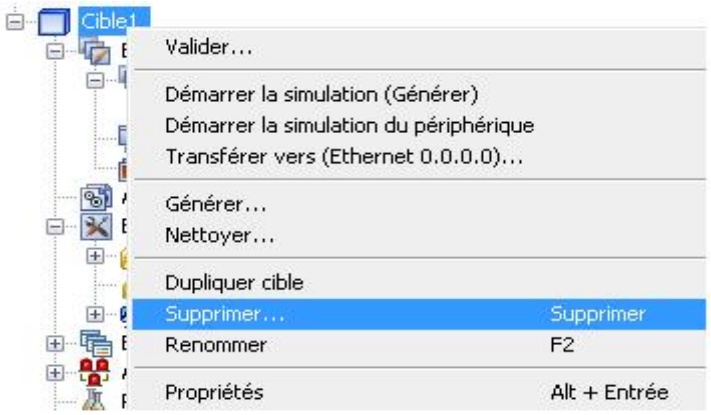


2. Dans la boîte de dialogue Créer une nouvelle cible, sélectionnez le type et le modèle de la cible, puis cliquez sur Suivant pour définir les propriétés de la cible puis Cliquez sur Terminer.

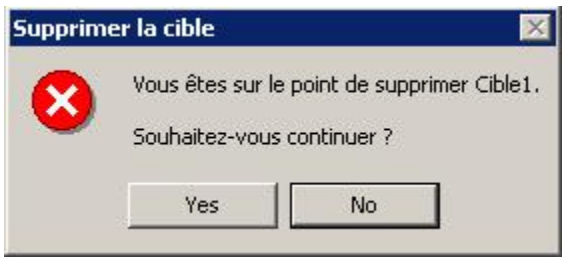


C.5.2 Suppression d'une cible

1. Dans l'onglet Projet de la fenêtre du Navigateur, cliquez avec le bouton droit de la souris sur la cible à supprimer et sélectionnez Supprimer. [11]



La boîte de dialogue Supprimer la cible s'affiche.



2. Cliquez sur Oui.

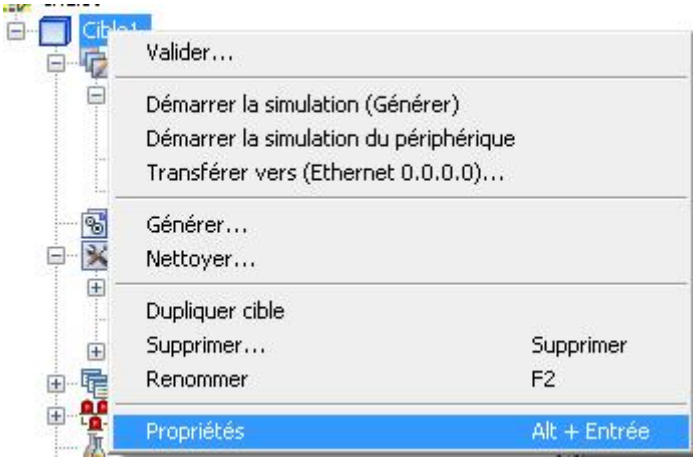
C.5.3 Copie d'une cible

1. Dans l'onglet Projet de la fenêtre du Navigateur, cliquez avec le bouton droit de la souris sur la cible à copier et sélectionnez Dupliquer cible.

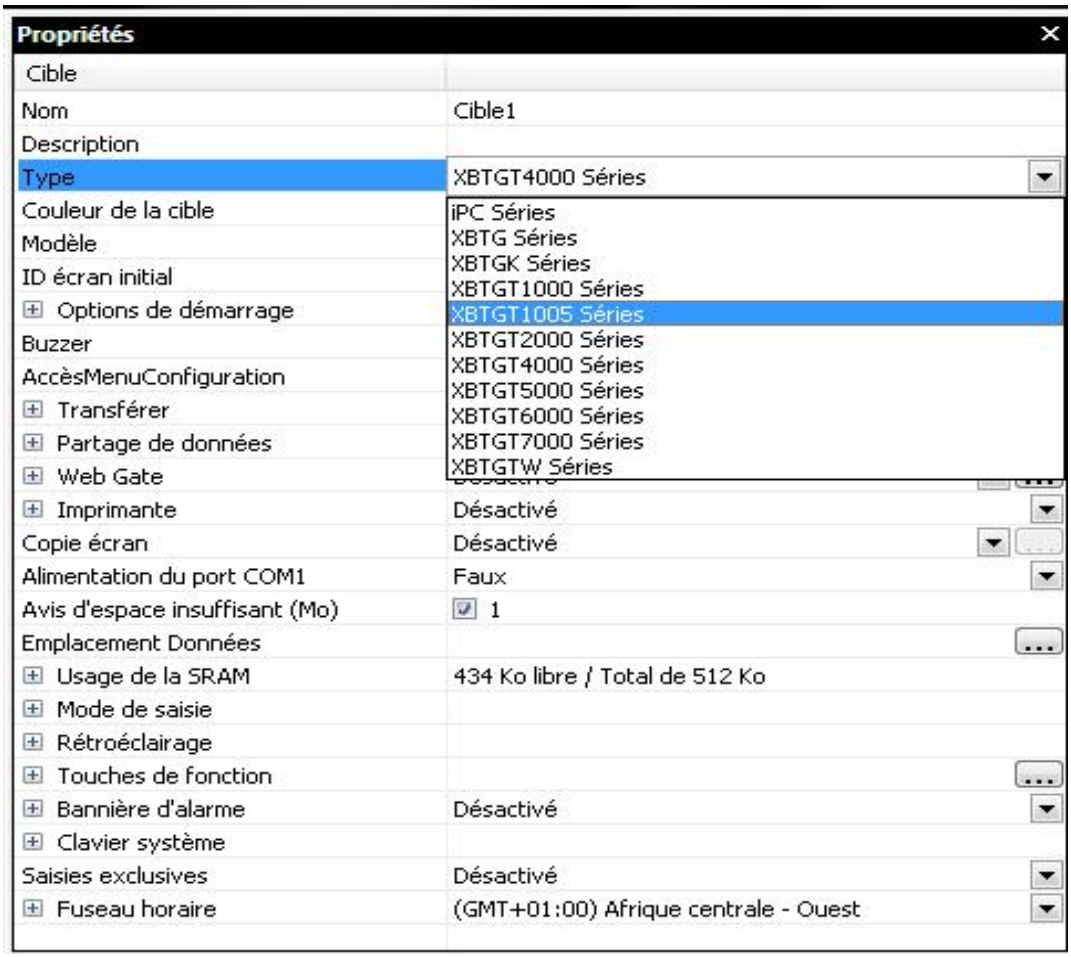
C.5.4 Modification d'une cible

Puisque Vijeo-Designer prend en charge le développement sur plates-formes multiples, vous pouvez modifier la cible sans limitation, même après la fin du développement du projet.

1. Dans l'onglet Projet de la fenêtre du Navigateur, sélectionnez la cible à modifier et par cliquant droit sélectionnez propriétés.



2. Dans la propriété Type, modifiez la série de la cible, et dans la propriété Modèle, sélectionnez le modèle de la cible pour terminer la transformation. Vous pouvez maintenant transférer le projet vers la machine cible définie.



C.6 Manipulation des outils de communication

C.6.1 Ajout d'un pilote de périphérique

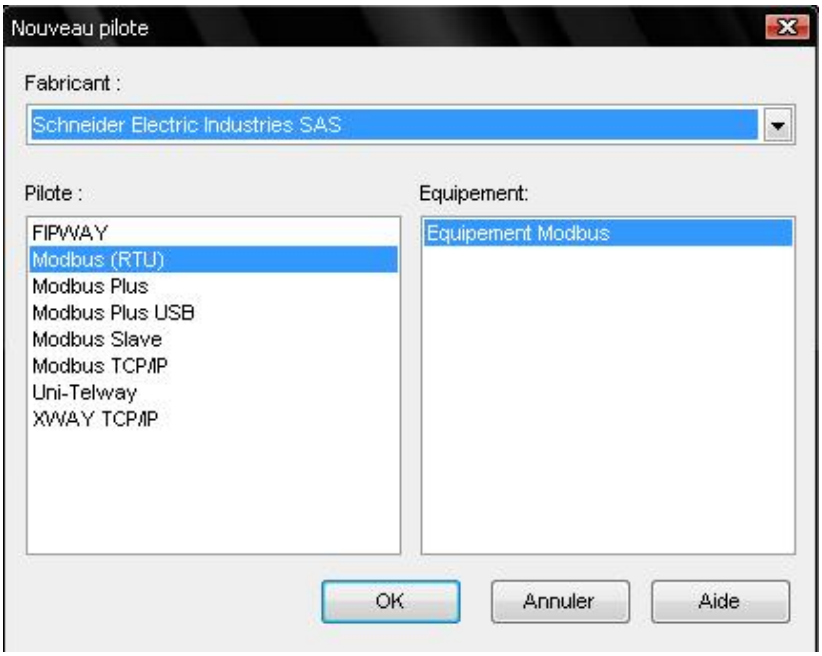
Pour pouvoir échanger des données, les pilotes s'appuient sur des programmes qui se comportent comme des ponts entre l'environnement d'exécution et l'équipement. Il n'est donc pas nécessaire de créer un programme complexe de communication. Il suffit d'ajouter et de configurer le pilote pour permettre à la machine cible et à l'équipement.

La procédure elle est comme suite :

1. Dans la fenêtre du Navigateur, cliquez avec le bouton droit de la souris sur le Nî ud Gestionnaire E/S, puis cliquez sur Nouveau pilote.

2. Dans la boîte de dialogue Nouveau pilote, sélectionnez le fabricant, le pilote et l'équipement.

- Le champ **Fabricant** identifie le fabricant de l'équipement. Pour connecter un lecteur de code à barres, sélectionnez Generic.
- Le champ **Pilote** indique le nom du pilote. Pour plus d'informations, reportez vous à l'Annexe 2, Manuels des pilotes de périphérique.
- Le champ **Equipement** identifie l'équipement relié à la machine cible.



C.6.2 Configuration de paramètres de communication

Après avoir ajouté un pilote à votre projet, configurez les paramètres de communication. Les paramètres de communication concernent le pilote et L'équipement.

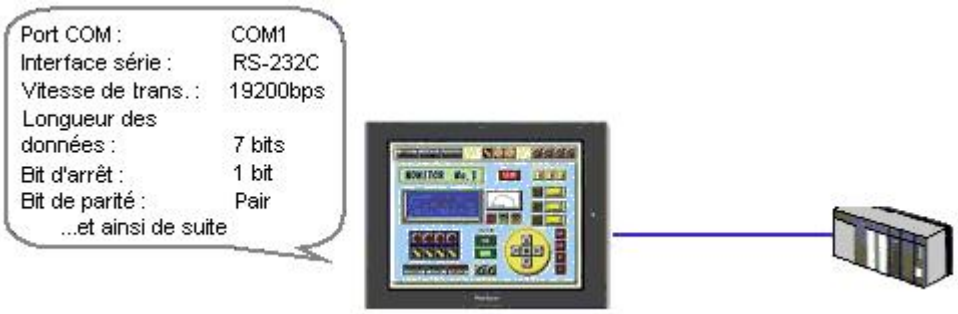
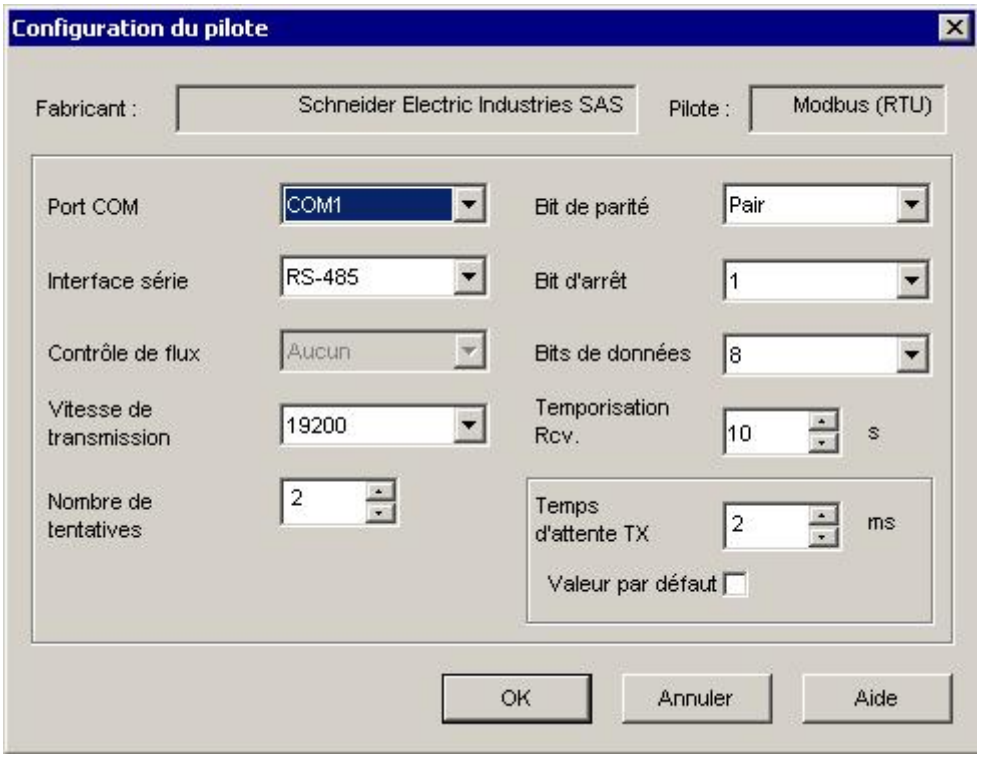


Figure C.2: *exemple d'une configuration des paramètres de communication* [5]

La procédure elle est comme suite :

1. Cliquez avec le bouton droit de la souris sur le Nî ud Pilote, puis cliquez sur Configuration.

2. Définissez les paramètres de communication dans la boîte de dialogue Configuration du pilote. Selon le pilote sélectionné, différents champs apparaissent



3. Cliquez sur OK

4. Cliquez avec le bouton droit de la souris sur le Nî ud Equipement, puis cliquez sur Configuration.

5.Définissez les paramètres de communication dans la boîte de dialogue Configuration de l'équipement.

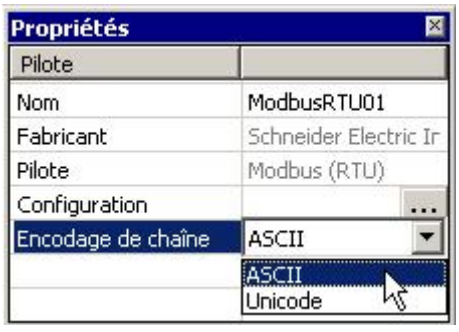
La boîte de dialogue Configuration de l'équipement ci-dessous concerne :

- **Fabricant :** Schneider Electric Industries SAS
- **Pilote :** Modbus (RTU)
- **Equipement :** Equipement Modbus

➢ Après avoir configuré les paramètres de communication dans Vijeo-Designer, vous devez les configurer sur l'équipement.

C.6.3 Configuration de codes caractère

Vous pouvez configurer des codes de caractère dans la propriété Encodage de chaîne du pilote. Sélectionnez ASCII, Unicode ou ANSI.



Le code de caractère sélectionné dans la propriété Encodage de chaîne est utilisé lors de l'écriture de données dans l'équipement à partir du clavier et lors de l'affichage de données à partir d'adresses de périphérique. [11]

Code	Propriétés
ASCII	Toutes les données textuelles échangées entre l'équipement et une machine cible sont lues/écrites sous la forme des codes ASCII d'un octet. En règle générale, vous pouvez lire et écrire les codes de caractères compris entre 00 et 7F dans les affichages numériques. Vous pouvez également lire et écrire les codes de caractères compris entre 80 et FF selon la police que vous utilisez. Pour une liste de tous les caractères ASCII utilisés lors du runtime.
Unicode	Les données textuelles échangées entre l'équipement et une machine cible sont lues/écrites sous la forme d'un code Unicode de deux octets.
ANSI	ANSI utilise une version étendue du tableau de code ASCII. Lorsque les caractères ANSI sont activés, toutes les données texte communiquées entre l'équipement et la machine cible sont lues/écrites comme des codes ASCII mono-octet. Pour une liste de tous les caractères ANSI.

C.6.4 Modification des paramètres de communication lors du runtime

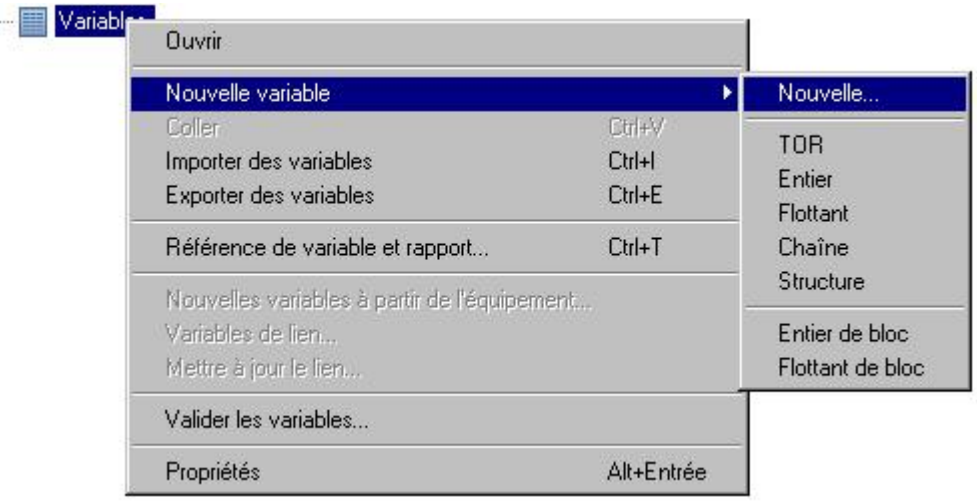
Après avoir configuré les paramètres de communication pour l'équipement dans Vijeo-Designer, et transféré l'application utilisateur dans les machines cibles, il se peut que vous mettiez à jour le réseau ou remplaciez le matériel existant. Cette action nécessitera peut-être la reconfiguration des paramètres de communication de l'équipement.

Au lieu de modifier les paramètres dans Vijeo-Designer et transférer l'application à nouveau, vous pouvez modifier les paramètres directement sur la machine cible.

C.7 manipulation des variables

C.7.1 Création de variables

1. Cliquez à droite sur le nî ud Variable et sélectionnez Nouvelle variable.

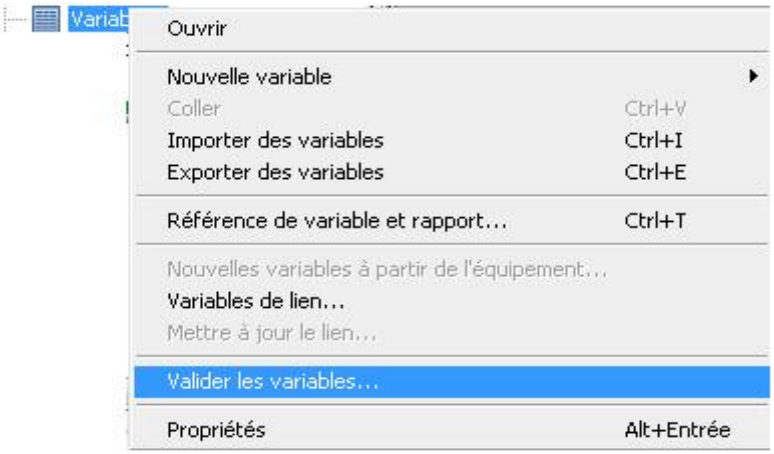


2. Dans l'onglet Propriétés de base, définissez le nom de la variable, sa description, son type de données, les dimensions du tableau et les propriétés de la source de données.

	Nom	Type de données	Source de donn...	Groupe de scrut...	Adresse du péri...	Groupe d'alarmes	Groupe de journ...
1	A	TOR	Interne			Désactivé	Aucun
2	B	TOR	Interne			Désactivé	Aucun
3	C	TOR	Interne			Désactivé	Aucun
4	D	TOR	Interne			Désactivé	Aucun
5	marche	TOR	Interne			Désactivé	Aucun
6	position	Entier	Interne			Désactivé	Aucun

C.7.2 Validation de variables

Pour valider vos variables ,cliquez à droite sur le nî ud Variable et sélectionnez Valider les variables. La fenêtre Compte-rendu s'affiche et contient des messages concernant le processus de validation de la variable.



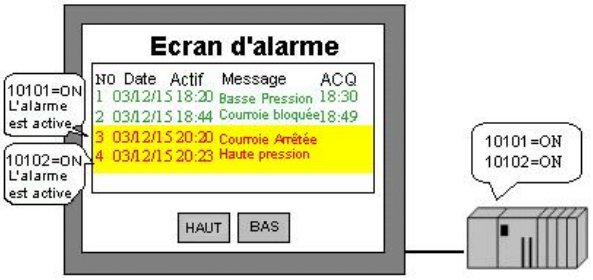
C.8 Les alarmes

Dans Vijeo-Designer, on peut indiquer les alarmes à l'utilisateur sous divers formats d'affichage.

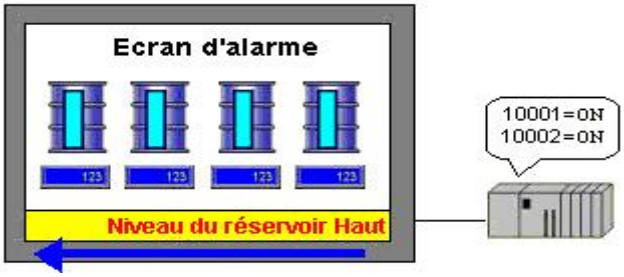
- **Affichage du résumé des alarmes**

Sur ce format on peut afficher une liste d'alarmes sur un écran à l'aide du résumé des alarmes. Les alarmes peuvent avoir quatre états : actif, acquitté (ACQ), non acquitté (NONACQ), et retombée. Le résumé d'alarmes peut afficher des alarmes qui sont dans ces quatre états. Vous pouvez imprimer ces messages d'alarme ou les enregistrer dans un fichier .csv.

Bannière d'alarme

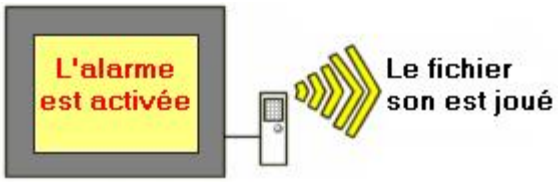


Vous pouvez afficher des messages Actif et NONACQ d'alarme sur un écran à l'aide du format de bannière d'alarme. Si plusieurs alarmes deviennent actives au même moment, les messages d'alarme de la bannière s'affichent par ordre d'activation. Vous pouvez désactiver l'affichage des messages d'alarme en configurant la propriété Message affiché de la bannière d'alarme pour le nî ud cible.



Son

Vous pouvez utiliser le son pour avertir les usagers d'une activation d'alarme.



C.9 Manipulation Des Recettes

C.9.1 Fonctions clés des recettes

Vijeo-Designer fournit certaines fonctions clés dans les recettes :

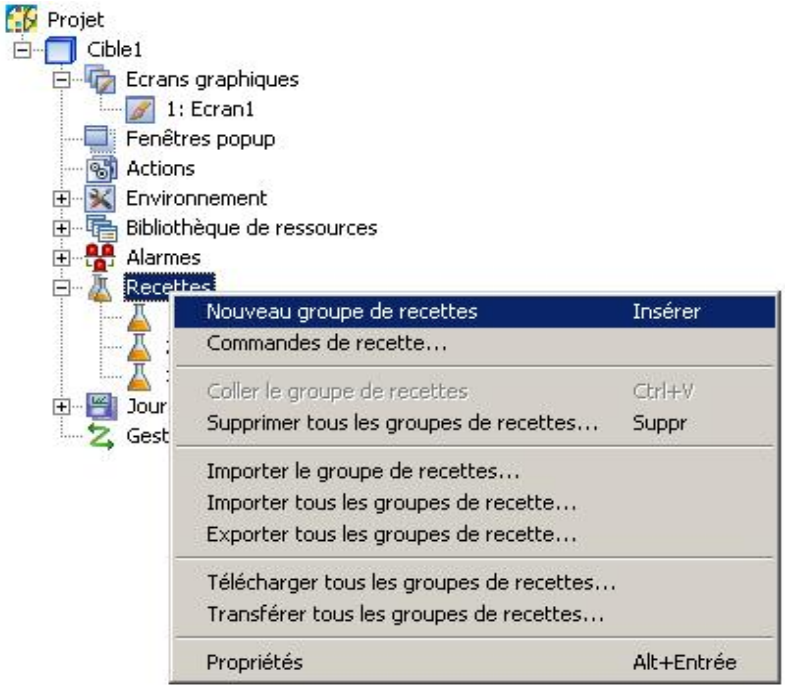
- Les recettes peuvent être automatisées ou contrôlées en utilisant les boutons dans l'écran. En utilisant les objets de recette, vous pouvez visualiser et modifier la liste de groupes de recette et les recettes.
- Pour chaque groupe de recettes, il existe un fichier de groupe de recettes unique qui contient les recettes et d'autres informations reliées aux recettes. Vous pouvez créer ou modifier les recettes dans l'Editeur de groupe de recettes ou lors de runtime, qui peut être exigé lorsque le matériel opère de façon différente d'un site à l'autre. En utilisant la fonction Téléchargement, vous pouvez remplacer les recettes dans l'Editeur avec les recettes modifiées lors de runtime.
- Les objets de la bibliothèque d'objets sont disponibles dans Vijeo-Designer afin de faciliter la gestion des groupes de recette dans une application ou pour effectuer les opérations de recette. Vous trouverez un ensemble d'objets, comme un gestionnaire de

recettes et le statut des recettes, et des boutons pour charger, enregistrer, envoyer et prendre des images instantanée.

C.9.2 Création d'un groupe de recettes

Pour créer et gérer un groupe de recettes, utilisez le Nî ud Recettes dans la fenêtre Navigateur. Vous pouvez créer jusqu'à 32 groupes de recette pour chaque cible.

1. Cliquez avec le bouton droit de la souris sur le Nî ud Recettes, puis sélectionnez Nouveau groupe de recettes.



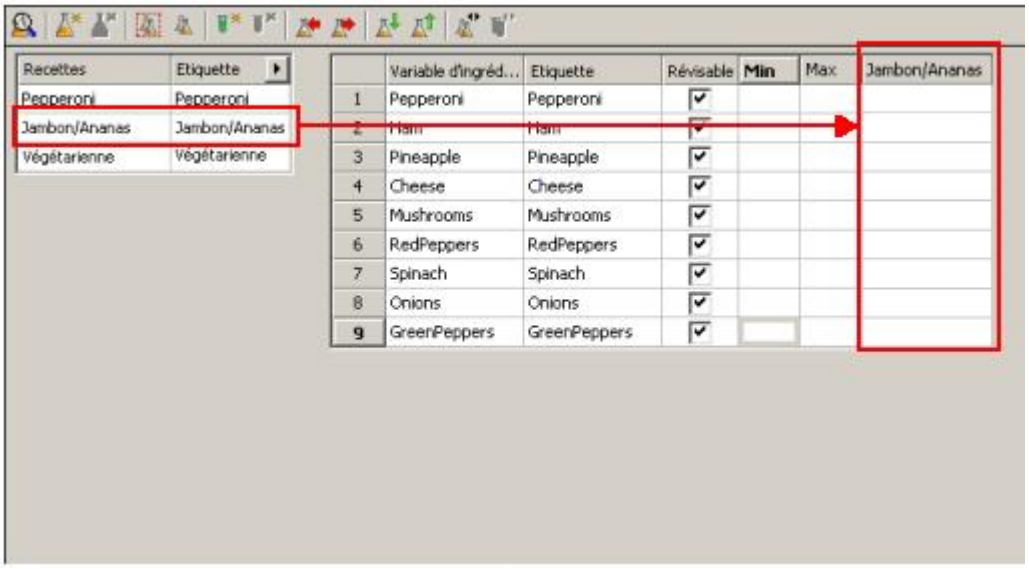
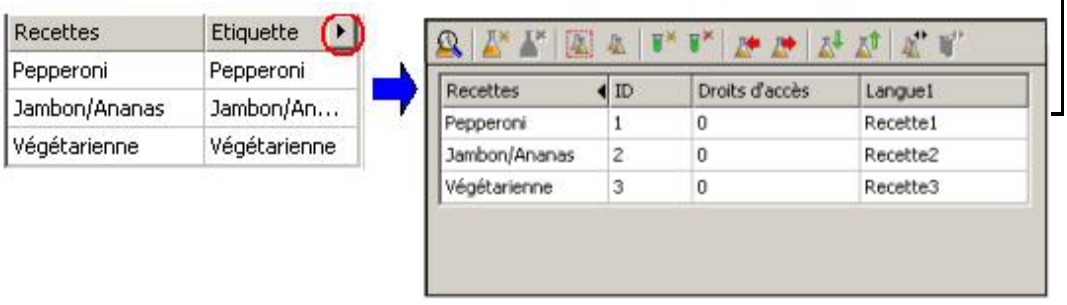
C.9.3 Modification d'un groupe de recettes

1. Sélectionnez le groupe de recettes que vous souhaitez modifier.

2. Dans la fenêtre de l'Editeur Groupe de recettes, modifiez les propriétés pour un groupe de recettes existant ou définissez les propriétés pour un nouveau groupe de recettes.

Les tableaux suivants présentent les propriétés de l'éditeur du groupe de recettes.

Volet gauche

Propriété	Description
Recettes	Affiche les valeurs de recette. Pour ajouter une nouvelle recette, cliquez sur l'icône Nouvelle recette . Cliquez pour sélectionner une recette dans la liste de recettes et pour l'afficher dans l'Editeur du groupe de recettes. Pour faire afficher plus d'une recette et les valeurs de l'ingrédient, maintenez la touche CTRL enfoncée et cliquez pour sélectionner les recettes désirées. Les valeurs d'ingrédient pour les recettes sélectionnées sont affichées dans une colonne.  Les valeurs que vous saisissez devraient être comprises entre la plage minimale et maximale définie. Les valeurs de recette non valides sont affichées en rouge.
Etiquettes	Affiche l'étiquette de recette dans la langue active. Lorsque l'affichage des étiquettes de recette est développé, toutes les langues sont affichées pour les étiquettes de recette. Cliquez sur ▶ pour développer l'affichage des étiquettes de recette, dans lequel vous pouvez gérer les ID, affecter le droit d'accès pour chaque recette, et afficher les étiquettes de recette dans des langues différentes. 

	• Recettes : Affiche les noms des recettes, mais seulement dans l'Editeur. Lors du runtime, la machine cible affiche le nom selon la langue affectée à la recette. Vous pouvez saisir jusqu'à 32 caractères pour chaque nom de recette. • ID : Affiche une ID de recette unique. Vous pouvez saisir une valeur comprise entre 1 et 65535. • Droits d'accès : Indique les droits d'accès pour la recette. Saisissez une valeur comprise entre 0 et 65535. • Langue : Affiche le nom de la recette comme il apparaîtra lors du runtime.

Volet droit

Propriété	Description
Variable d'ingrédient	Affiche une liste de variables associées avec un ingrédient. La création d'un groupe de recettes ne générera pas nécessairement des variables d'ingrédient automatiquement. Cliquez sur l'ic  Nouvel ingrédient pour créer une variable d'ingrédient. Chaque ligne affiche une variable pour un ingrédient unique, à l'exception des variables de bloc. Lorsque vous ajoutez une variable de bloc, chaque élément dans le bloc est ajouté comme un ingrédient. Cliquez sur  pour ouvrir l'Editeur d'expressions, où vous pouvez sélectionner une variable existante ou créer une nouvelle variable..
Etiquette	Affiche le nom de l'ingrédient qui est présenté sur l'opérateur de l'écran. L'étiquette peut contenir jusqu'à 32 caractères. Cliquez sur ▶ pour développer l'affichage, où vous pouvez changer

	une étiquette d'ingrédient pour plusieurs langues. Dans l'exemple suivant, les étiquettes sont faites pour trois langues : l'anglais, le français, et l'espagnol.  Etiquette Pepperoni Jambon Ananas Fromage Champignons Poivrons Rouges Epinards Oignons Poivrons Verts Langue1 / Langue2 / Langue3 Pepperoni / Pepperoni / Pepperoni Jambon / Ham / Jamon Ananas / Pineapple / Pina Fromage / Cheese / Queson Champignons / Mushrooms / Champinones Poivrons Rouges / Red Peppers / Pimentones Rojos Epinards / Spinach / Espinaca Oignons / Onions / Cebolla Poivrons Verts / Green Peppers / Pimentones Verdes
Modifiable	Sélectionnez si vous voulez que l'ingrédient soit modifiable lors du runtime. Retirez la coche de la case si vous voulez que l'ingrédient soit fixe. Les variables de bloc sont configurées comme un bloc complet. Ce paramètre n'est pas disponible pour les adresses particulières dans le bloc.
Min.	Précisez les valeurs minimale et maximale de l'ingrédient. Les valeurs saisies ici contrôlent les valeurs qui peuvent être saisies par l'opérateur pour la valeur d'ingrédient lors du runtime. Ces paramètres ne sont pas disponibles pour les variables TOR, de chaîne, ou de bloc.
Max.	

Tableau C.3 : *les propriétés de l'éditeur du groupe de recettes.* [11]

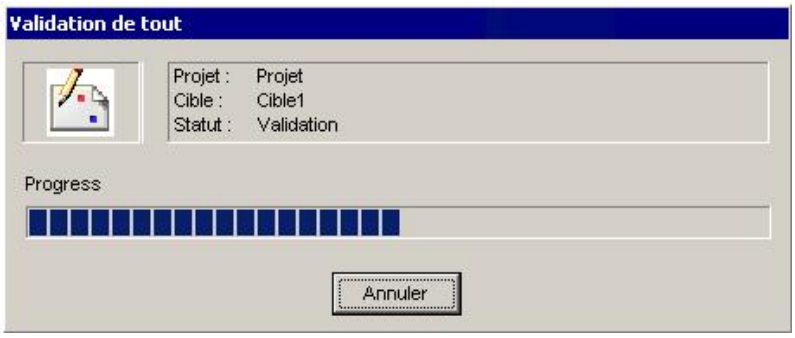
C.10 Recherche des erreurs avant un transfert

Pour rechercher les éventuelles erreurs avant de générer et de transférer un projet, vous pouvez :

- Valider le projet.
- Simuler les opérations de Runtime.

La validation de l'application se fait comme suite :

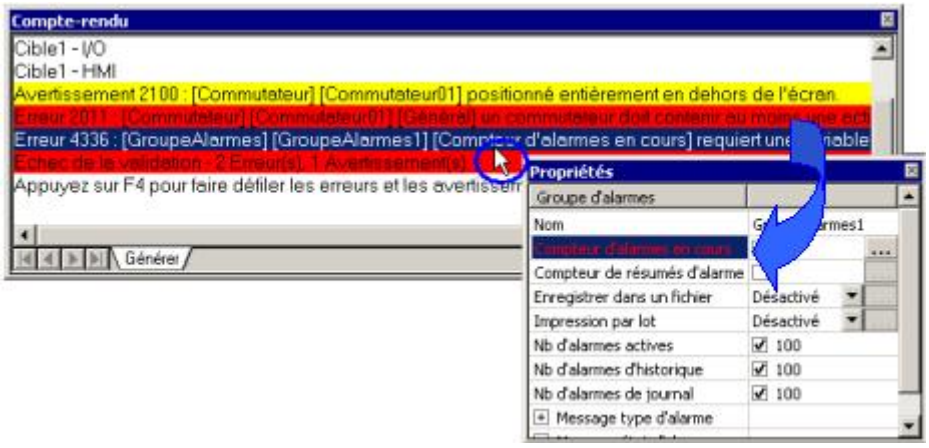
1. Dans l'onglet Projet de la fenêtre du Navigateur, cliquez avec le bouton droit de la souris sur le Nî ud de la cible, puis sélectionnez Valider.



La boîte de message de progression apparaît

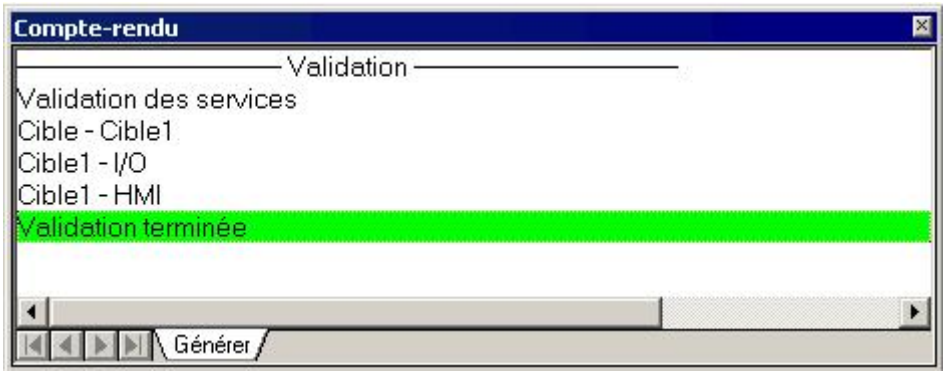
2. La fenêtre Compte-rendu s'affiche et contient des messages concernant le processus de validation. Les erreurs sont surlignées en rouge et les avertissements en jaune.

En cas d'erreurs ou d'avertissements, cliquez deux fois sur le message d'erreur, ou appuyez sur F4, pour afficher l'emplacement de l'erreur correspondante



3. Corrigez les erreurs et procédez à une nouvelle validation. Recommencez la procédure jusqu'à ce que toutes les erreurs soient résolues.

Une fois toutes les erreurs résolues, le message surligné en vert "Validation terminée" s'affiche, indiquant la réussite de l'opération.

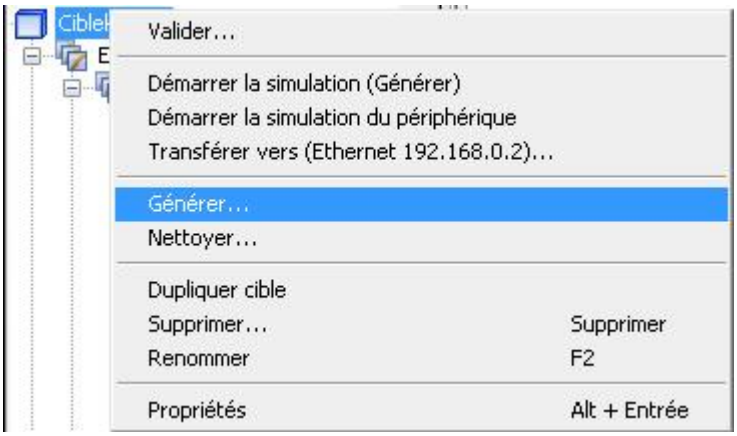


C.11 Génération du projet

Après avoir validé le projet et corrigé toute erreur, vous pouvez générer l'application utilisateur exécutée dans Vijeo-Designer Runtime. Le processus de génération contrôle la présence d'éventuelles erreurs dans le projet, en plus de rapporter les erreurs de compilation Java.

L'application utilisateur doit être générée avant de pouvoir être exécutée en simulation ou sur la machine cible.

Dans l'onglet Projet de la fenêtre du Navigateur, cliquez avec le bouton droit de la souris sur le nî ud de la cible, puis sélectionnez Générer[11]

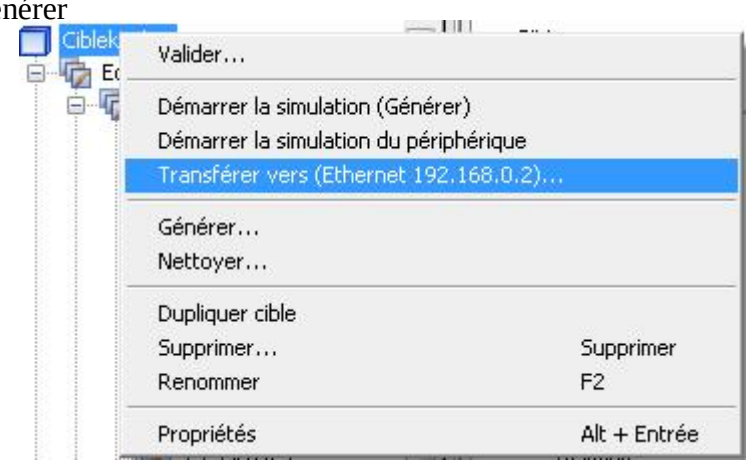


C.12 Transfert du projet

Avant de pouvoir transférer un projet, celui-ci doit avoir été correctement validé et généré. L'opération de transfert permet de transférer tous les fichiers nécessaires vers la machine cible afin d'exécuter l'application utilisateur.

Procédure du transfert

Dans l'onglet Projet de la fenêtre du Navigateur, cliquez avec le bouton droit de la souris sur le nî ud de la cible, puis sélectionnez Générer



ANNEXE D

Description de lôautomate Modicon M340

D.1 Plate-forme dôautomatisme Modicon M340

D.2 Processeurs Modicon M340

D.3 Structure mémoire

D.4 Module dôalimentation

D.5 Configuration monorack

D.6 La communication

D.7 Mise en place des processeurs

Annexe D .

D.1 Plate-forme dôautomatisme Modicon M340

Modicon M340 est un concentré de puissance et d'innovation offrant des réponses optimales aux besoins des constructeurs de machines. Il est également le compagnon idéal de Modicon Premium et Modicon Quantum pour satisfaire les exigences dôautomatisation des procédés industriels et des infrastructures[6]

Figure D.1 : *Automate Modicon M340* [6]

D.2 Processeurs Modicon M340

Les processeurs Standard et Performance de la plate-forme dôautomatisme Modicon M340 gèrent l'ensemble d'une station monorack automate dont 11 emplacements maximum peuvent être équipés de :

- modules d'entrées/sorties ñTout ou Rienò,
- modules d'entrées/sorties analogiques,
- modules métiers (comptage, communication Ethernet TCP/IP).

Quatre processeurs sont proposés, ils se différencient par leurs capacités mémoire, vitesses de traitement, nombre dôE/S et nombre et type de ports de communication. De plus, selon le modèle, ils proposent au maximum et dôune manière non cumulative :

- de 512 à 1024 entrées/sorties ñTout ou Rienò,
- de 128 à 256 entrées/sorties analogiques,
- de 20 à 36 voies métiers comptage,
- de 0 à 2 réseaux Ethernet TCP/IP (avec ou sans port intégré et un module réseau).

Selon les modèles, les processeurs Modicon M340 intègrent :

- un port Ethernet TCP/IP 10BASE-T/100BASE-TX,
- un bus machines & installations CANopen,

- une liaison série Modbus,
- une prise TER de type USB (pour connexion dôun terminal de programmation).

Chaque processeur est fourni avec une carte mémoire permettant :

- la sauvegarde de lôapplication (programme, symboles et constantes)
- lôactivation dôun serveur Web de base du port Ethernet intégré de classe

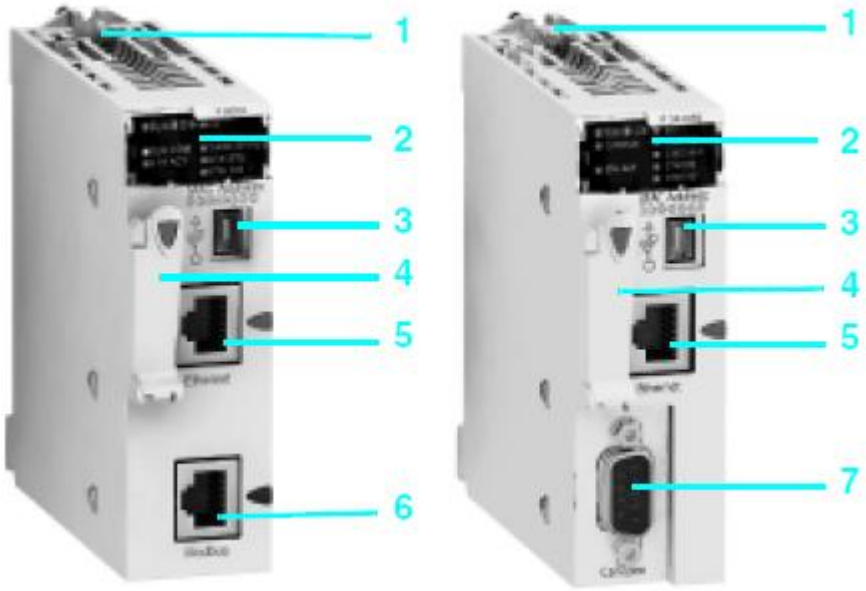


Figure D.2 : *Processeurs : BMX P34 2020 et BMX P34 2030*[6]

Description des processeurs avec port Ethernet TCP/IP intégré BMX P34 2020/2030

Les processeurs Performance BMX P34 2020/2030 simple format comprennent en face avant :

1 Vis de sécurité pour verrouillage du module dans son emplacement (repère 0) du rack.

2 Un bloc de visualisation comprenant, selon modèle 8 ou 10 voyants :

- Voyant RUN (vert) : processeur en fonctionnement (exécution du programme),
- Voyant ERR (rouge) : défaut processeur ou défaut système,
- Voyant I/O (rouge) : défaut provenant des modules dôentrées/sorties,
- Voyant SER COM (jaune) : activité sur la liaison série Modbus,
- Voyant CARD ERR (rouge) : absence ou défaut de la carte mémoire,
- Voyant ETH ACT (vert) : activité sur le réseau Ethernet TCP/IP,
- Voyant ETH STS (vert) : état du réseau Ethernet TCP/IP,
- Voyant ETH 100 (rouge) : débit binaire sur le réseau Ethernet TCP/IP (10 ou 100 Mbit/s,

Avec en plus, pour le modèle BMX P34 2030 :

- Voyant CAN RUN (vert) : bus machine/installation intégré opérationnel.
- Voyant CAN ERR (rouge) : défaut bus machine/installation intégré.

3 Un connecteur type USB mini B pour le raccordement d'un terminal de programmation (ou dôun terminal de dialogue opérateur Magelis XBT GT/GK/GTW).

4 Un emplacement équipé de sa carte mémoire Flash pour la sauvegarde de lôapplication. Un voyant, situé au dessus de cet emplacement indique la reconnaissance ou lôaccès à la carte mémoire.

5 Un connecteur type RJ45 pour le raccordement au réseau Ethernet TCP/IP 10BASE-T/100BASE-TX.

Avec selon modèle :

6 Processeur BMX P 34 2020 : un connecteur type RJ45 pour liaison série Modbus ou liaison mode caractères (RS 232C/RS 485, 2 fils, non isolée),

7 Processeur BMX P 34 2030 : un connecteur type SUB-D 9 contacts pour bus machines & installations CANopen maître.

En face arrière : 2 commutateurs rotatifs pour lôattribution de lôadresse IP. Cette attribution est définie selon 3 modes :

- adresse fixée par la position des 2 commutateurs,
- adresse fixée par les paramètres de lôapplication,
- adresse fixée par le serveur BOOTP du réseau Ethernet TCP/IP.

D.3 Structure mémoire

RAM interne application

La mémoire application se décompose en zones mémoire, réparties physiquement dans la mémoire RAM interne du processeur Modicon M340 :

1 Zone des données de l'application de 2 types possibles :

- Données localisées (located data) correspondant aux données définies par une adresse (exemple %MW237) à laquelle peut être associée un symbole (exemple Comptage_rebus).
- Données non localisées (unlocated data) correspondant à des données définies uniquement par un symbole. Lôutilisation des données non localisées supprime les contraintes de gestion de la

localisation mémoire du fait de lôattribution automatique des adresses et permet également la structuration et la réutilisation des données. La sauvegarde de cette zone de données est assurée automatiquement sur mise hors tension de lôautomate par la duplication de son contenu dans une mémoire interne non volatile de 256 K octets, intégré au processeur. par ailleurs, il est également possible de réaliser à tout moment un ñbackupò de cette mémoire par programme utilisateur.

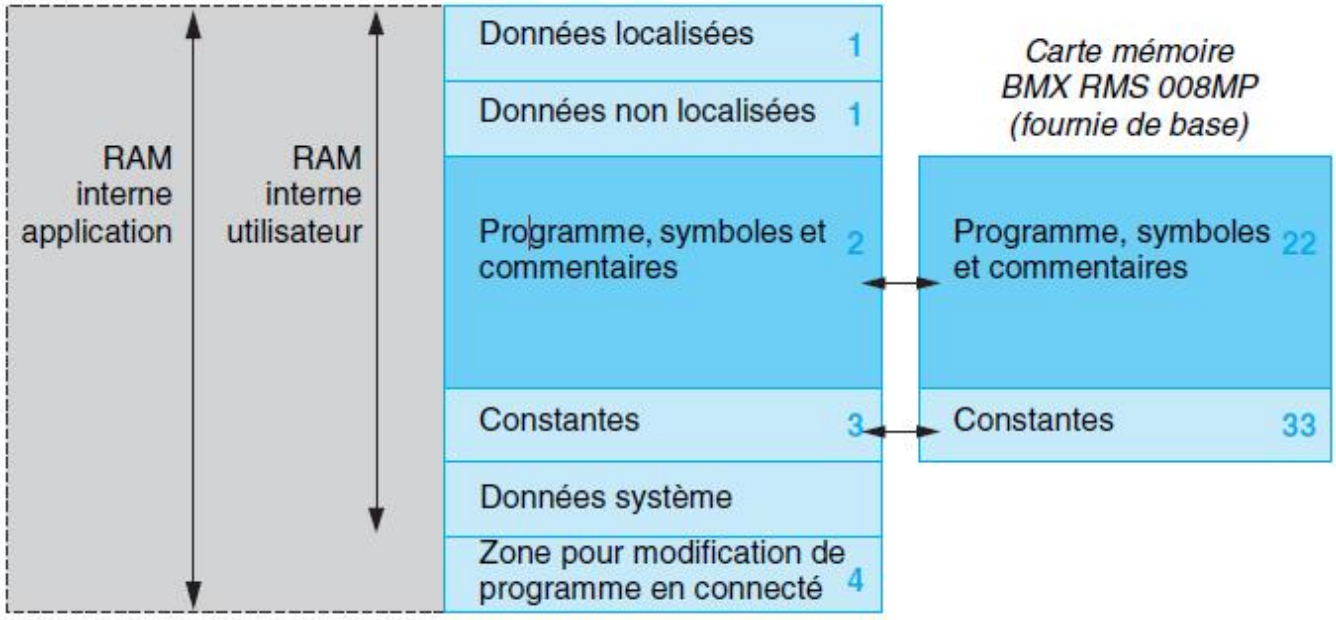


Figure D.3 : *Structure de la mémoire* [6]

2 Zone programme, symboles et commentaires. Cette zone contient au niveau du programme son code binaire exécutable et son code source IEC.

3 Zone constantes, cette zone supporte les données localisées de type constantes (%KWi).

4 Zone pour modification de programme en mode connecté

D.4 Module dôalimentation

Présentation

Les modules alimentation BMX CPS ppp0 sont destinés à l'alimentation de chaque rack BMX XBP pp00 et de ses modules installés. Deux types de modules alimentation sont proposés :

- modules alimentation pour réseau à courant alternatif,
- modules alimentation pour réseau à courant continu.

Description

Le module alimentation est choisi en fonction :

- du réseau d'alimentation électrique : c 24 V, c 48 V ou a 100é 240 V,
- de la puissance nécessaire.

Les modules alimentation BMX CPS ppp0 disposent en face avant de :

1 Un bloc de visualisation comprenant :

- un voyant OK (vert), allumé si les tensions racks sont présentes et correctes,
- un voyant 24 V (vert), allumé lorsque la tension capteur est présente (uniquement pour les modules alimentation à réseau courant alternatif BMX CPS 2000/3500).

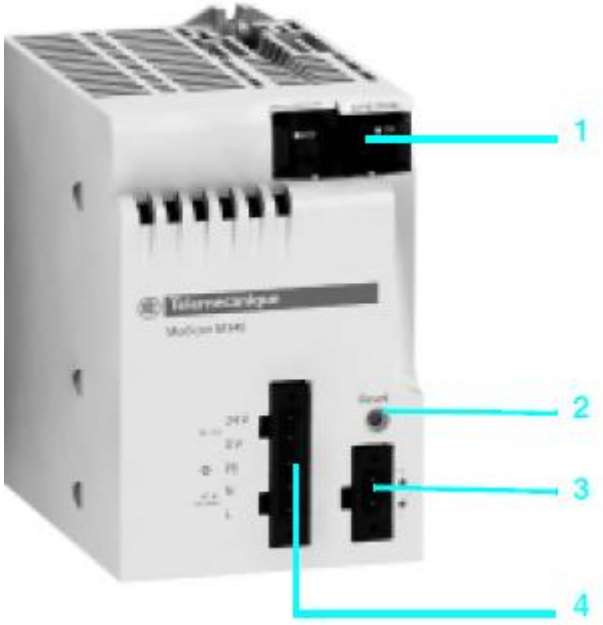


Figure D.4 : *Module dôalimentation* [6]

2 Un bouton-poussoir RESET à pointe de crayon provoquant une reprise à froid de l'application.

3 Un connecteur 2 contacts recevant un bornier débrochable (à vis à cage ou à ressorts) pour le raccordement du relais alarme.

4 Un connecteur 5 contacts recevant un bornier débrochable (à vis à cage ou à ressorts) pour le raccordement :

- du réseau dôalimentation CC ou AC,
- de la terre de protection,

- de la tension c 24 V dédiées à lôalimentation des capteurs dôentrées (uniquement avec modules alimentation courant alternatif BMX CPS 2000/3500).

D.5 Configuration monorack

Présentation

Le rack BMX XBP pp00 constitue lôélément de base de la plate-forme dôautomatisme Modicon M340 en configuration monorack. Ces racks assurent les fonctions suivantes :

- Fonction mécanique : ils permettent la fixation de l'ensemble des modules d'une station automate (alimentation, processeur, entrées/sorties ñTout ou Rienò, entrées/sorties analogiques et métiers). Ces racks peuvent être fixés sur panneau, platine ou profilé DIN :
- dans des armoires,
- dans des bâtis de machines, é
- Fonction électrique : les racks intégrent un bus X. Ils permettent :
- la distribution des alimentations nécessaires à chaque module d'un même rack,
- la distribution des signaux de service et des données pour l'ensemble de la station automate,
- lôembrochage et le débrochage des modules sous tension et en fonctionnement.

Description

Les racks BMX XBP pp00 disponibles en 4, 6, 8 ou 12 emplacements comprennent :

1 Un support métallique assurant les fonctions suivantes :

- le support de la carte électronique bus X et la protection de celle-ci contre les perturbations de type EMI et ESD,
- le support des modules,
- la rigidité mécanique du rack.

2 Une borne de terre pour prise à la terre du rack.

3 Trous pour la fixation du rack sur un support. Ces trous permettent le passage de vis M6.

4 Points de fixation de la barre de reprise blindage.

5 Trous taraudés recevant la vis de verrouillage de chaque module.

6 Un connecteur pour module dôextension. Ce connecteur repéré XBE est non utilisé pour cette version.

7 Connecteurs ½ DIN 40 points femelles assurant la connexion entre le rack et chaque module. A la livraison du rack, ces connecteurs sont protégés par des caches qui devront être retirés avant la mise en place des modules. Fenêtres destinées à lœncrage des ergots des modules.

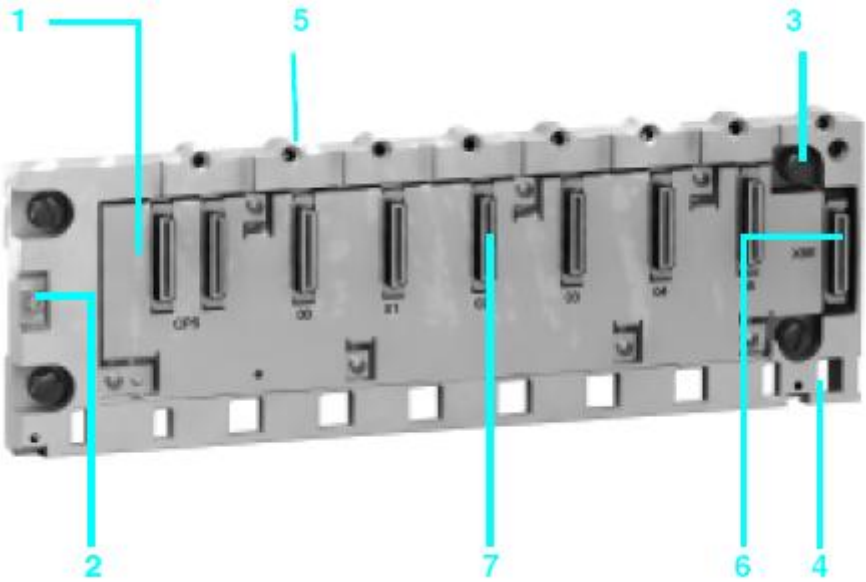


Figure D.5 : *Profilé à 6 emplacements BMX XBP 0600* [6]

D.6 La communication

Présentation

La liaison série Modbus permet de répondre aux architectures maître/esclave (il est néanmoins nécessaire de vérifier que les services Modbus utiles à l'application soient implémentés sur les équipements concernés). Le bus est composé d'une station maître et de stations esclaves. Seule la station maître peut être à l'initiative de l'échange (la communication directe entre stations esclaves n'est pas réalisable). Deux mécanismes d'échange sont possibles :

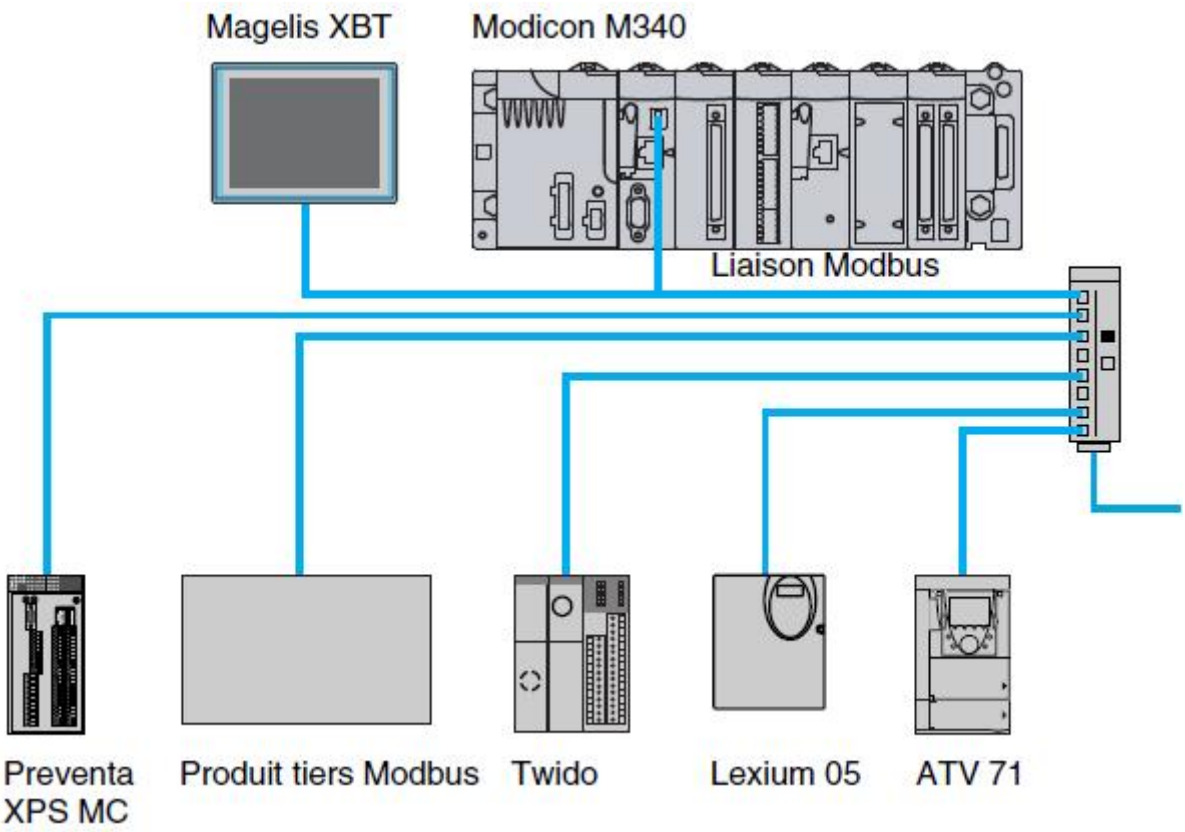


Figure D.6 : *présentation du réseau Modbus* [6]

- Question/réponse, les demandes du maître sont adressées à un esclave donné. La réponse est attendue en retour de la part de lœsclave interrogé.
- Diffusion, le maître diffuse un message à toutes les stations esclaves du bus. Ces dernières exécutent l'ordre sans émettre de réponse.

Description

Les processeurs BMX P34 1000/2010/2020 de la plate-forme Modicon M340 intègrent une liaison série pouvant être utilisée sous protocole Modbus maître/esclave RTU/ASCII ou sous protocole mode caractères. Relatif à ce port série, ces processeurs disposent en face avant de :

1 Un bloc de visualisation comprenant entre autres voyants :

- Voyant SER COM (jaune) : activité sur la liaison série (fixe) ou défaut sur un équipement présent sur la liaison (clignotant).

2 Un connecteur type RJ45 pour liaison série Modbus ou liaison mode caractères (RS 232C/RS 485 non isolée) avec son index 3 de couleur noire.

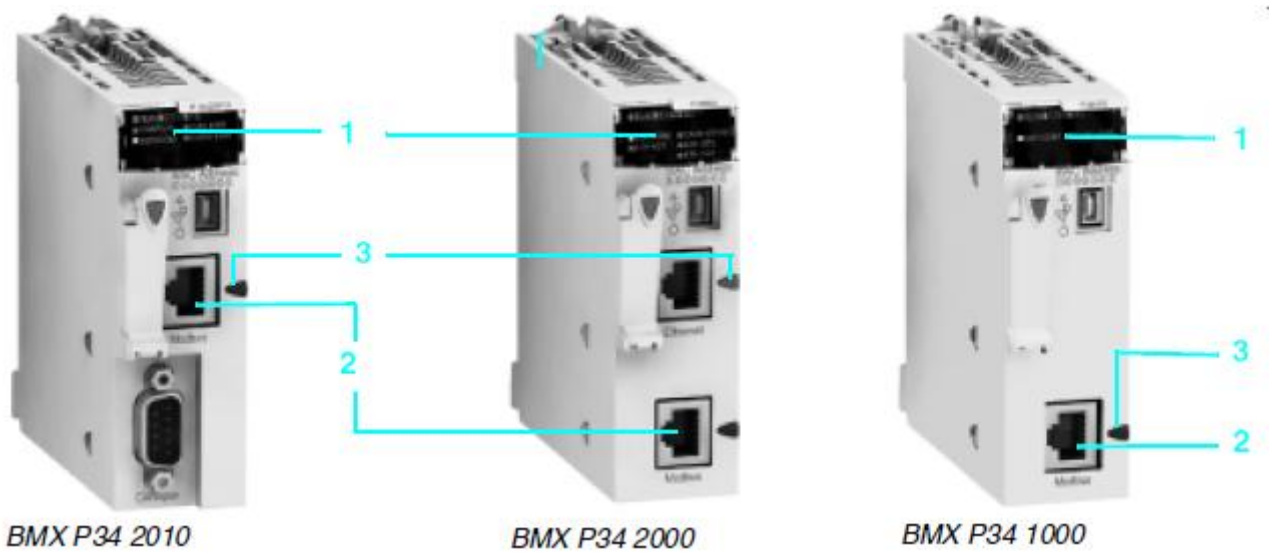


Figure D.7 : *description de la communication M340* [6]

Système de câblage

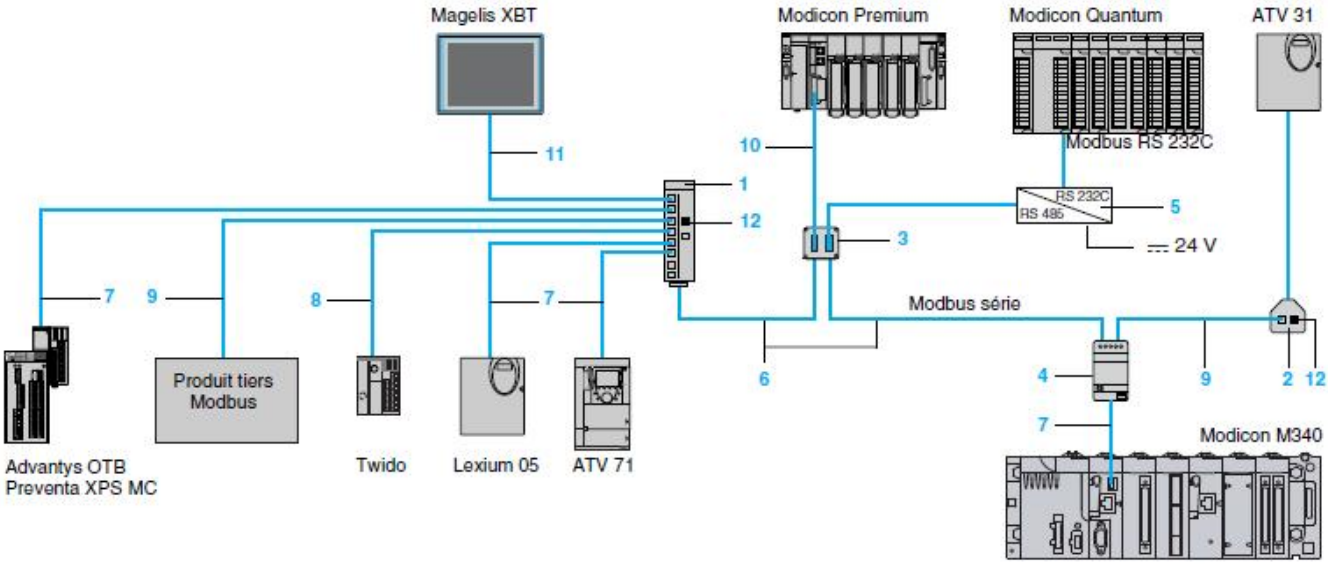


Figure D.8 : *système de câblage* [6]

Désignation	Description	Repère	Longueur	Référence unitaire
Répartiteur Modbus	- 10 connecteurs RJ45 - 1 bornier à vis	1	–	**LU9 GC3**
Tés de dérivation	- 2 connecteurs RJ45 - Un câble intégré avec connecteur RJ45 - Dédié variateurs Altivar et Lexium	2	0,3 m 1 m	**VW3 A8 306 TF03** **VW3 A8 306 TF10**
Boîtier de dérivation passif	- Dérivation et prolongation du bus - Adaptation fin de ligne	–	–	**TSX SCA 50**
Prise abonnés passive 2 voies 2 connecteurs SUB-D femelle 15 contacts et 2 bornier à vis	- Dérivation 2 voies et prolongation du câble principal - Codage d'adresse - Adaptation fin de ligne	3	–	**TSX SCA 62**
Boîtier de dérivation Bornier à vis pour câble principal 1 connecteur RJ45 pour dérivation	- Isolement de la liaison série RS 485 - Adaptation fin de ligne (R = 120 Ω, C = 1 nF) - Pré-polarisation de ligne *(1)* (2 R = 620 Ω) Alimentation ⎓ 24 V *(2)* Montage sur ⎍ 35 mm	4	–	**TWD XCA ISO**
Boîtier de dérivation 3 connecteurs RJ45	- Adaptation fin de ligne (R =120 Ω, C = 1 nF) - Pré-polarisation de ligne *(1)* (2 R = 620 Ω) Montage sur ⎍ 35 mm	–		**TWD XCA T3RJ**
Adaptateur Modbus / Bluetooth®	- 1 adaptateur Bluetooth® (portée 10 m, classe 2) avec 1 connecteur RJ45 - 1 cordon long. 0,1 m pour PowerSuite avec 2 connecteurs RJ45 - 1 cordon de long. 0,1 m pour TwidoSuite, avec 1 connecteur RJ45 et 1 connecteur mini DIN - 1 adaptateur RJ45/SUB-D mâle 9 contacts pour variateurs ATV	–		**VW3 A8 114**
Convertisseur de ligne RS 232C/RS 485 sans signaux modem	Alimentation ⎓ 24 V/20 mA, 19,2 Kbit/s Montage sur ⎍ 35 mm	5	–	**XGS Z24**
Adaptateur de fin de ligne	Pour connecteur RJ45 R = 120 Ω, C = 1 nF	12	*Vente par O. indiv*	**VW3 A8 306 RC**

Tableau D.1 : *Eléments de dérivation et d'adaptation pour liaison série RS* [6]

Désignation	Description	Repère	Longueur	Référence unitaire	Masse kg
Câbles principaux double paire torsadée blindée RS 485	Liaison série Modbus, livrés sans connecteur	6	100 m	**TSX CSA 100**	5,680
			200 m	**TSX CSA 200**	10,920
			500 m	**TSX CSA 500**	30,000
Cordons Modbus RS 485	2 connecteurs RJ45	7	0,3 m	**VW3 A8 306 R03**	0,030
			1 m	**VW3 A8 306 R10**	0,050
			3 m	**VW3 A8 306 R30**	0,150
	1 connecteur RJ45 et 1 connecteur SUB-D 15 contacts	–	3 m	**VW3 A8 306**	0,150
	1 connecteur mini-DIN pour contrôleur Twido et 1 connecteur RJ45	8	0,3 m	**TWD XCA RJ003**	0,040
			1 m	**TWD XCA RJ010**	0,090
			3 m	**TWD XCA RJ030**	0,160
	1 connecteur RJ45 et 1 extrémité fils libres	9	3 m	**VW3 A8 306 D30**	0,150
	1 connecteur miniature et 1 connecteur SUB-D 15 contacts	10	3 m	**TSX SCP CM 4530**	0,180
Cordons pour afficheurs et terminaux Magelis XBT	1 connecteur RJ45 et 1 connecteur SUB-D 25 contacts pour : - XBT N200/N400/NU400 - XBT R410/411 - XBT GT2...GT7 (port COM1) *(1)*	11	2,5 m	**XBT Z938**	0,210
	2 connecteurs RJ45 pour : - XBT GT1 (port COM1) - XBT GT2...GT7 (port COM2)	11	3 m	**VW3 A8 306 R30**	0,150

Tableau D.2 : *Câbles et cordons de raccordement pour liaison série RS* [6]

D. 7 Mise en place des processeurs

Les processeurs BMX P34 1000/2000/2010/2020/2030 sont alimentés par le bus du rack.

Les opérations de mise en place (implantation, montage et démontage) sont détaillées ci-après [7].

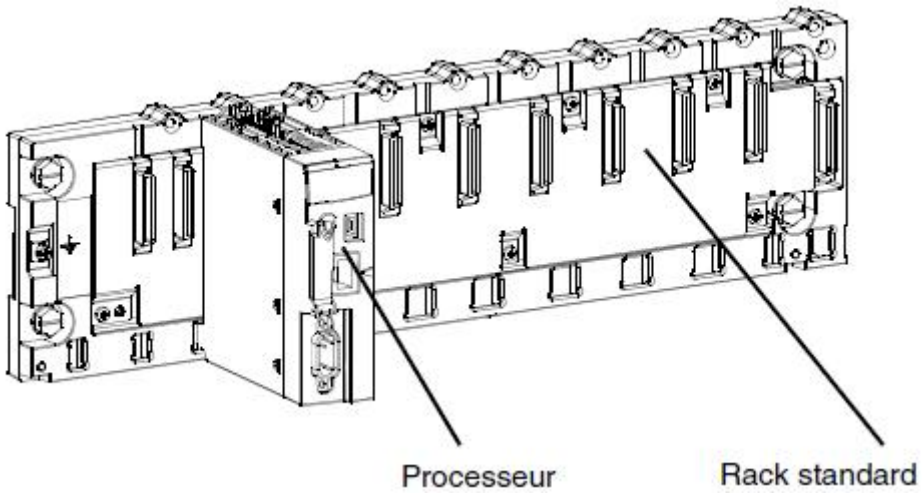


Figure D.9 : *un processeur BMX P34 2010 monté dans un rack BMX XBP 0800* [7]

Annexe D .

Le tableau ci-dessous présente la procédure d'installation d'un processeur sur un rack.

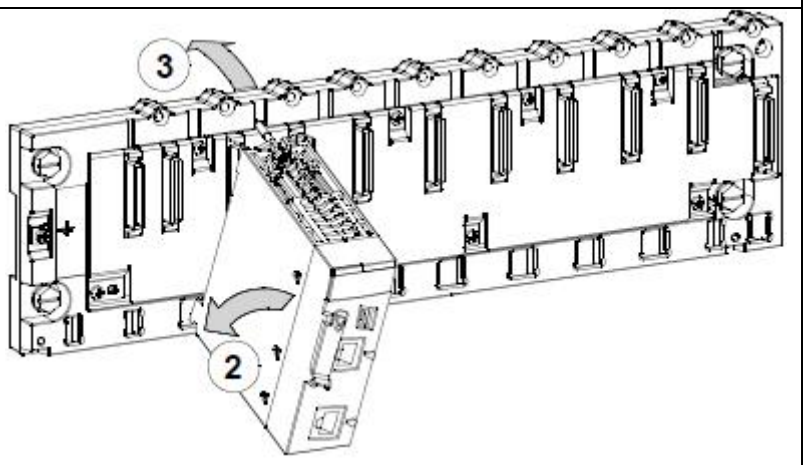
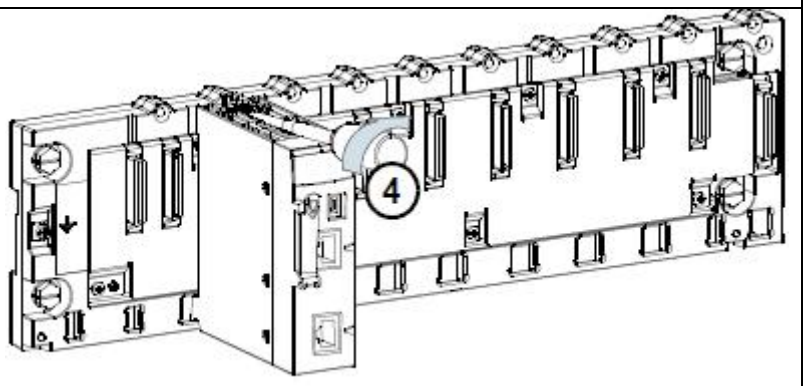
Etape	Action	Illustration
1	Vérifiez que l'alimentation est hors tension et qu'une carte mémoire adaptée est utilisée.	
2	Placez les deux ergots situés à l'arrière du module (dans la partie inférieure) dans les emplacements correspondants du rack.	
3	Faites pivoter le module vers le haut du rack de façon à plaquer le module sur le fond du rack. Il est alors maintenu en position.	
4	Serrez la vis de sécurité pour assurer le maintien en position du module sur le rack.	

Tableau D.3 : *procédure d'installation d'un processeur sur le rack* [7]