REPUPLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

ECOLE NATIONALE POLYTECHNIQUE D'ALGER

Département de Génie Electrique

Spécialité : Automatique

Projet de fin d'études en vue de l'obtention du Diplôme d'Ingénieur d'Etat en Automatique

Commande et supervision du bâtiment intelligent Schneider Electric

Réalisé par :

Mr MEDJAOURI Anas

Mr ARIM Mohamed

Proposé et dirigé par :

Pr E.M.BERKOUK

Mr A.YOUNSI

Juin 2010

Ecole Nationale Supérieure Polytechnique, 10 Av. Hassan Badi, El Harrach, Alger, Algérie

REMERCIEMENTS

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

Nous remercions également notre co-promoteur Mr Abdelghani YOUNSI, de SHNEIDER ELECTRIC, pour son encadrement, ses 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, spécialementA.Mossab et S.Maaradji pour nous avoir tant appris.

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 tenons aussi à remercier Abdelmalek et son employé Amine de Sirus Automation pour nous avoir aidés lorsque nous avons eu besoin d'eux.

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

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 ma mère, mon père ainsi qu'à mes sœurs qui m'ont soutenu et aidé tout au long de mon parcours et sans qui je n'en serais pas la

A toute ma famille qui m'a tant aidé dans le besoin et l'adversité

A Selma pour son soutient, son aide et sa présence

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

A mon ami et camarade Anas avec qui j'ai travaillé avec grand plaisir pendant deux années

A tous mes cousins et cousines : Menad, Boudji et Lydia…

A mes amis qui m'ont tant supporté, spécialement Sphinxou, Krizou et Khaled...

A tous mes camarades de notre chère école avec qui j'ai passé de bons moments et appris beaucoup de choses

A tous ceux qui ont contribué de près ou de loin à notre travail

A tous ceux que je n'ai pas cités et qui sont présents dans mes pensées

Mohamed ARIM

DEDICACES

Je dédie ce modeste travail tout d'abord à ma très chère mère et à mon père qui ont su m'épauler dans les moments difficiles, qui m'ont tout appris, tant donné sans rien demandé en retour… sans eux je ne serais pas l'homme que je suis.

A mon frère et à ma sœur que j'adore tout simplement

A ma grand-mère, mon grand père qui j'espère, de la où ils sont me regardent avec fierté

A mon ami d'enfance, billel que j'ai toujours considéré comme mon grand frère
A mon ami et camarade Arim avec qui j'ai travaillé avec grand plaisir pendant deux années
A mes cousins : Lhadi , Ishak, Oussama, Zakaria
A mes amis : nouh, yacine, oussama, zeryab, nadjib, et anis…avec qui j'ai passé de bons moments.

À toute l'équipe Schneider Electric, surtout Samir, Ali et Abdelghani.
A tous ceux qui on contribué de loin ou de prêt à notre travail.
A ma famille et à tous ceux que je n'ai pas cités et qui sont présents dans mes pensées

Anas MEDJAOURI

TABLE DES MATIERES

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

GTB Gestion Technique de Bâtiment

IHM Interface Homme/Machine

LI langage liste instructions

LD langage à contacts LADDER

MAST tâche maître

RAM Random Access Memory

ROM Read Only Memory

ST langage structuré

TOR tout ou rien

INTRODUCTION GENERALE

Introduction générale :

L'efficacité énergétique du bâtiment joue un rôle croissant dans la lutte contre le réchauffement Climatique. Les technologies du bâtiment moderne jouent un rôle prépondérant dans la réduction de la consommation énergétique de l'habitat. Si l'on en croit des études en cours, les solutions domotiques globales de commande de l'éclairage et du confort thermique d'un bâtiment allégeraient la note d'électricité de près de 60 %.

L'automate programmable industriel A.P.I est aujourd'hui le constituant le plus répandu pour réaliser des tâches complexes et autonomes, il joue un rôle prépondérant dans l'automatisme des bâtiments intelligents. 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.

Dans le but d'amélioration des applications (programmes) de Gestion Technique de bâtiments, Schneider Electric nous a proposé la réalisation des DFB (blocs fonctions dérivées) permettent de structurer et d'optimiser ces applications. Dans ce cadre : Plusieurs DFB, un programme ainsi qu'une interface finale vont être mises en place englobant l'essentiel des systèmes que nous pourrions trouver dans un bâtiment intelligent.
Cette solution sera à base d'automate Schneider : Modicon M340 Ethernet.

Plan de travail :

Le travail est présenté en quatre chapitres complémentaires :

Dans le premier chapitre, nous avons parlé des bâtiments intelligents, de la gestion technique des bâtiments, de la solution Schneider Electric (les automates programmables en général, les capteurs…) et enfin de l'architecture des réseaux de communication.

Le second chapitre explicite le logiciel de programmation d'automates « Unity Pro ».
Ainsi que les différents langages de programmation qu'il admet. Plusieurs exemples seront réalisés et simulés grâce aux outils du logiciel.

Le troisième chapitre aborde le logiciel « Vijeo Citect » qui sert à programmer les interfaces Homme-machine (HMI). Un exemple avec plusieurs écrans, alarme, graphe… sera illustré dans ce chapitre.

Ces trois derniers chapitres constituent une préparation au projet qui nous a été soumis.
Le dernier chapitre aborde la partie application qui consiste à programmer et implémenter plusieurs tâches de domotique qu'on a appliquée au siège de Schneider Electric d'Alger, nous avons ensuite réalisé une interface homme machine facilitant la supervision et l'interaction, aboutissant à une gestion technique complète du bâtiment.

Nous finirons par une conclusion générale sur les résultats obtenus, les connaissances assimilées durant ce projet et les perspectives en vue.

CHAPITRE I

BATIMENT INTELLIGENT

Introduction :

Ces dernières années, la construction est devenue toujours plus complexe. En effet, les exigences des gens augmentent pratiquement en parallèle avec la pression sur les prix, surtout à cause des coûts d'exploitation et d'énergie. Pour satisfaire les besoins de ce marché, il faut savoir appliquer des nouvelles méthodes, technologies et produits. La gestion technique du bâtiment fait partie de ces nouvelles technologies. Dans les bâtiments fonctionnels modernes et les maisons individuelles d'un certain confort, les systèmes d'automatisation de bâtiments font déjà partie de l'équipement de base.

Sous la dénomination «Habiter intelligent», de nouvelles fonctions de l'habitation moderne apparaissent sur le marché. Cette appellation comprend toutes les installations allant de la domotique et de la sécurité classique jusqu'aux appareils ménagers en réseau en passant par la communication à large bande pour tous les médias («convergence») et l'électronique de loisir. Le système global qui en résulte offre plus de confort, d'efficience énergétique et de souplesse.

1. Le bâtiment intelligent [1] :

Un bâtiment intelligent est celui qui permet un environnement productif et rentable en misant sur l'optimisation et l'interrelation des quatre niveaux fondamentaux suivants : l'infrastructure, les systèmes, les services et la gestion. L'immeuble intelligent aide le propriétaire, le gestionnaire et les occupants à réaliser leurs objectifs de coûts, de confort, de services, de sécurité, de flexibilité à long terme et de mise en marché.

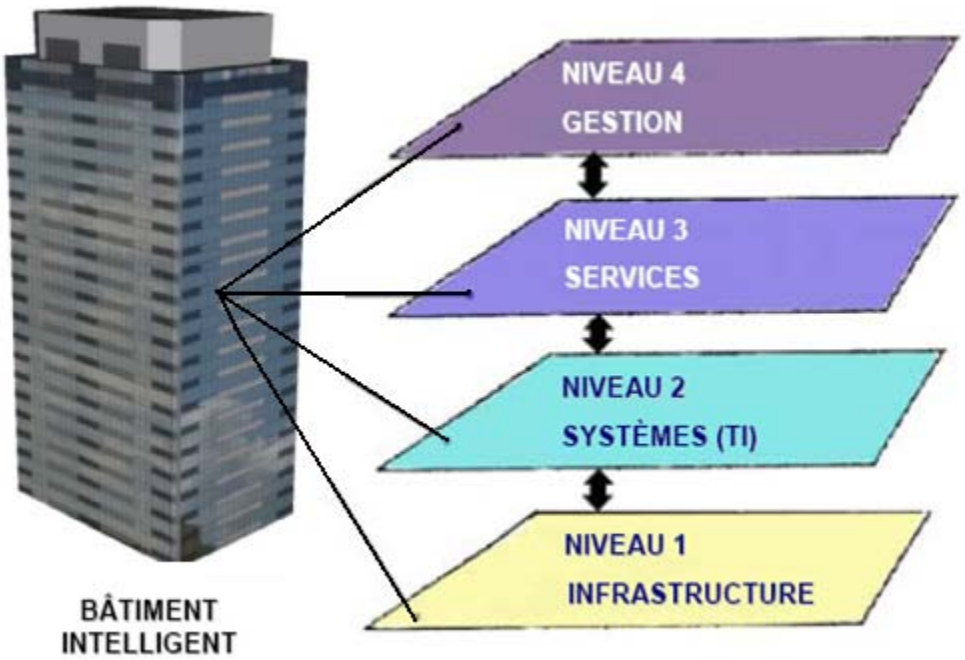


Figure I.1 : Les Niveaux du Bâtiment Intelligent

1.1. L'infrastructure :

C'est le niveau de base qui favorise l'interopérabilité des systèmes. Il est couramment identifié en tant que « réseau de câblage structuré de télécommunication ».
Il est formé d'éléments passifs tels des câbles en cuivre, des fibres optiques et des salles de répartition et d'équipements aménagées de façon à permettre l'établissement et le maintien des liens de communication.

Des organismes, tels l'ANSI et son pendant canadien la CSA Internationale, établissent, en collaboration étroite avec les manufacturiers d'équipements électroniques et de communication, des normes définissant la conformité de l'installation et des tests de performance. L'utilité des normes est d'assurer la pérennité des réseaux sur une période allant jusqu'à 25 ans, indépendamment des applications qu'ils devront supporter.

1.2. Les systèmes :

Le second niveau d'un bâtiment intelligent regroupe tous les éléments actifs à l'usage de ses occupants. C'est le domaine des Ti (technologies de l'information). Ordinateurs, serveurs, concentrateurs passerelles, routeurs, systèmes de téléphonie et de messagerie vocale, incluant les sans fil, font partie des appareils électroniques installés dans les salles d'équipement de l'infrastructure, ou aux postes de travail, qui procurent les moyens sophistiqués d'entrer en communication et de fournir les services.

Entrent également dans la catégorie des Ti tous les logiciels d'exploitation et d'application, qui permettent l'interface essentielle entre les appareils et les usagers potentiels. Par ailleurs, dans le domaine traditionnel de la construction d'un bâtiment, les systèmes électromécaniques utilisent des contrôleurs dotés de processeurs « intelligents », capables d'échanger de l'information entre eux, telle la température, l'humidité, etc., par le biais du réseau de câblage structuré. Ces contrôleurs sont dispersés dans les salles techniques qui abritent les systèmes.

Dans tous les cas, l'échange d'information et l'interopérabilité entre les appareils et systèmes sont rendus possibles par l'utilisation en commun d'un protocole d'encodage/décodage et d'une grande vitesse de transmission.

1.3. Les services :

Ce niveau du bâtiment intelligent est caractérisé par l'ensemble des tâches accomplies à l'aide des systèmes, pour rencontrer les besoins des occupants. Ils concernent :

- la sécurité : accès, vidéo, incendie.

- l'automatisation d'édifice : contrôle des ascenseurs, optimisation des paramètres de confort et d'opération des systèmes électromécaniques, économie d'énergie, maintien des actifs.

- les communications : téléphonie, appel de garde, visioconférence, messagerie vocale, internet, intranet, courrier électronique.

- l'administration : tous les moyens utilisés pour l'organisation du travail et la planification financière.

- les soins : tous les systèmes utilisés pour parvenir à diagnostiquer, guérir, maintenir à domicile et rendre les services sociaux (DPE, carte santé, PACS, imagerie, télémédecine, etc.)

1.4. La gestion :

C'est le niveau supérieur du bâtiment intelligent qui établit les processus requis pour rendre les services de façon efficace et efficiente. Essentiellement, il est formé par le personnel spécialisé qui recherche constamment à obtenir les meilleurs résultats avec la meilleure technologie disponible.

La gestion technique du bâtiment (GTB) signifie la commande et le réglage transversal de tous les éléments d'une installation tels que l'éclairage, l'ombrage, le chauffage, la ventilation, la climatisation et la sécurité. La GTB est réalisée avec des composants munis de microprocesseurs servant à la fonction locale ainsi qu'à la communication avec les autres composants. Les fonctionnalités transversales et locales sont assurées par le paramétrage des composants. Grâce aux systèmes de bus, le câblage et l'installation deviennent nettement plus simples. Par contre, il faut maîtriser les nouveaux outils de programmation.

Elle désigne une partie de l'automatisation de bâtiments. Elle est composée d'un réseau de capteurs, d'actionneurs et de contrôleurs décentralisés et mis en réseau pour intégrer dans un seul niveau de communication un maximum l'installation technique. La gestion technique du bâtiment est appliquée dans deux secteurs principaux, à savoir:

- Sous la dénomination «Habiter intelligent» dans l'habitation pour les besoins de la domotique.

- Sous forme de «Automatisation de bâtiments» dans les petits bâtiments fonctionnels et comme «Automatisation de locaux» en tant que partie de l'ensemble du système d'automatisation de bâtiments dans les moyens et grands bâtiments fonctionnels.

2. Habiter intelligent :

Dans ce qui suit, les applications et les notions de base essentielles sont expliquées.

En matière de domotique, les exigences des clients augmentent, surtout en ce qui concerne la maison individuelle. A l'image des secteurs de construction automobile ou de technologie informatique, les nouveaux propriétaires se sont habitués à recevoir davantage de fonctions aux prix constamment à la baisse.

A l'avenir, les fonctions des différents équipements seront de plus en plus intégrées et la domotique mise en réseau avec le monde de loisirs, d'informations et de services grâce aux technologies de communication modernes. Voici un exemple: en fermant la porte de la maison, l'installation d'alarme sera mise en vigilance, les lumières éteintes, la température de chaque pièce réglée au minimum, le simulateur de présence activé et les machine à haute-

tension coupées du secteur. Pour une utilisation encore plus aisée, les boutons-poussoirs seront souvent remplacés par des écrans tactiles, disposant de connexions par câble ou sans fil.

3. Avantages et installations liés à l'habitation intelligente :

Pour l'essentiel, les avantages liés à l'habitation intelligente peuvent être répartis dans les catégories de base suivantes:

- Confort plus élevé
- Efficience énergétique, durabilité
- Sécurité accrue
- Communication (interne et externe) par voix, données en tout genre
- Loisirs, information
- Dans les habitations mises en réseau, les installations peuvent être attribuées pour l'essentiel aux catégories suivantes:
- Commandes domotiques, tous les systèmes commandant ou réglant de l'énergie sous forme d'électricité, de mazout, de gaz, d'eau etc. pour le bien être, la sécurité et la durabilité.
- Systèmes de communication pour données et voix, internes et externes, câblé ou sans fil. Les installations l'interphone, avec ou sans vidéo, constituent un domaine particulier.
- Systèmes audio/vidéo pour le «Home cinéma» ainsi que les installations «multi room».

Les commandes domotiques concernent pour l'essentiel les parties suivantes:

- **Eclairage**

aujourd'hui, un bon éclairage fait partie de notre confort: varier la lumière depuis différents endroits, commande de scénarios (faire apparaître une luminosité prédéfinie pour différents groupes de lampes en appuyant sur un seul bouton-poussoir), enclencher ou déclencher un éclairage par des détecteurs de présence ou de mouvement, la commande et le réglage des installations en fonction de la lumière naturelle, la commande centralisée (par ex. éteindre toutes les lampes par un seul bouton-poussoir en quittant l'habitation), commande par télécommandes multifonctions à infrarouge, la liaison avec l'installation d'alarme et la simulation de présence etc.

- **Ombrage**

les entraînements électriques pour volets roulants, jalousies et stores, à commande locale ou centralisée, commande de scénarios, commande par détecteur de vent, de pluie ou de soleil, en liaison avec l'installation d'alarme et/ou la simulation de présence etc.

- **Chauffage, ventilation**

L'utilisation rationnelle de l'énergie avec des pompes à chaleur, la ventilation contrôlée dans les locaux d'habitation (dans les maisons bien isolées, une bonne ventilation devient très importante!), l'énergie solaire, le réglage individuel des locaux en signalant au chauffage les besoins actuels en chaleur, l'enclenchement par téléphone portable ou Internet etc.

- **Sécurité**

les installations de détection d'effraction et d'incendie (aussi comme détecteur individuel), la surveillance technique d'appareils (par ex. lors d'une absence, le message «défaut congélateur» est envoyé à un service de surveillance autorisé), la simulation de présence (activer les groupes de lumières comme si les locaux étaient habités), la commutation «panique» (un interrupteur central, par ex. dans la chambre à coucher des parents, qui enclenche tous les groupes importants d'éclairage dans la maison pour faire fuir un intrus), l'éclairage de choc (enclenchement par le détecteur de mouvement ou l'installation d'alarme) etc.

- **Efficience énergétique/gestion d'énergie**

Dans les préoccupations actuelles à propos des menaces engendrées par le changement climatique, les questions concernant les coûts et l'efficience énergétique gagnent toujours en importance. Grâce aux capteurs et actionneurs mis en réseau avec les commandes et réglages correspondants, il est possible d'optimiser durablement l'efficience énergétique; par exemple en utilisant des détecteurs de présence, des capteurs de luminosité ou encore des réglages individuels du chauffage dans les différents locaux.

- **Appareils électroménagers**

La manipulation et la surveillance d'appareils électroménagers tels que lave-linge, réfrigérateur ou congélateur par écrans tactiles; l'instruction, le programme et les recettes par Internet; les possibilités de donner l'alarme par téléphone et par Internet. L'intégration dans le système de gestion d'énergie.

4. Automatisation de locaux dans les moyens et grands bâtiments fonctionnels :

4.1. Nouvelles exigences posées aux bâtiments fonctionnels :

Dans les bâtiments fonctionnels modernes tels que bâtiments administratifs, écoles et bâtiments culturels, la fonctionnalité des locaux ainsi que le bien-être des personnes sont très importants. La conception d'un tel bâtiment doit tenir compte des aspects suivants:

- Dans la pratique, l'utilisation ultérieure d'un bâtiment n'est jamais connue. En fait, les changements rapides de l'économie, dictés par la globalisation ou la restructuration à grande échelle, font que les besoins en locaux industriels et administratifs peuvent changer très rapidement. C'est la raison pour laquelle la souplesse d'utilisation d'un bâtiment devient l'uns de ses caractéristiques principales dans l'optique de maintenir ou d'augmenter sa valeur marchande.

- Le bien-être dépend en grande partie de différents facteurs individuels des personnes concernées (physiologie, psychologie, environnement social, motivation, âge ou sensibilité esthétique). Les formes de travail glissent inévitablement de l'engagement indéterminé vers l'organisation en projets et des engagements limités dans le temps. De ce fait, le bureau personnel disparaîtra petit à petit pour faire place à une utilisation

temporaire de l'espace en alternance avec d'autres locaux, comme par ex. le bureau privé (home office). Dans des entreprises modernes avec beaucoup d'activités externes, les places de travail sont souvent partagées. Il s'agit donc de créer des locaux permettant d'influencer les facteurs individuels du bien-être.

Ces exigences ne peuvent être réalisées qu'avec une combinaison bien équilibrée d'architecture de qualité et d'une bonne automatisation de bâtiments. Par ailleurs, les nouvelles méthodes appliquées en matière de construction industrielle vont exactement dans cette direction. Même si les bâtiments portent toujours la griffe de l'architecte en ce qui concerne le design et l'organisation des espaces, les méthodes de planification et de réalisation n'ont pas besoin d'être redéfinies à chaque fois.

4.2. Efficience énergétique grâce à l'automatisation de bâtiments :

A l'avenir, les bâtiments auront leur fiche technique indiquant leur classe en fonction de leur consommation d'énergie totale comme c'est déjà le cas aujourd'hui pour les voitures ou les appareils électroménagers. Une meilleure classification signifiera donc un avantage concurrentiel. En Europe par exemple, 40% de l'ensemble de la consommation totale d'énergie concerne l'alimentation des bâtiments en énergie thermique et électrique. En appliquant les principes de l'automatisation de bâtiments, cette consommation sera réduite de façon durable. Par le biais de la technique de mesure, de commande et de réglage optimisé, l'efficacité d'énergie des installations de chauffage, de climatisation, d'eau chaude, d'éclairage et d'ombrage pourra être augmentée jusqu'à 25%. **[1]**

Est-il encore possible d'économiser beaucoup d'énergie une fois toutes les parois et tous les toits isolés, toutes les fenêtres munies d'un triple vitrage et étanchées ainsi que toutes les installations de chauffage renouvelées? La réponse est clairement affirmative. En effet, les potentiels d'économie sont même encore très importants, surtout dans les domaines de la climatisation et de l'éclairage.

4.3. Automatisation de locaux – une partie importante de l'automatisation de bâtiments :

L'automatisation de bâtiments s'est développée sur les bases de la technique de mesure, de commande et de réglage d'installations de ventilation. Il y a 30 ans en effet, on a commencé à climatiser individuellement des locaux par des régulateurs séparés. La révolution de la microélectronique et de la technologie informatique a permis, par l'automatisation de bâtiments, d'accéder aux systèmes de transmission et de traitement d'informations.

Les efforts consentis dans la rationalisation de la maintenance des installations techniques ont largement contribué au développement de l'automatisation de bâtiments; sans cette dernière, la gestion technique des moyens et grands bâtiments fonctionnels ne serait aujourd'hui tout simplement pas possible.

Pour atteindre les objectifs fixés, l'automatisation de bâtiments moderne utilise de plus en plus les éléments suivants :

- Systèmes décentralisés de commande et de réglage. Ces systèmes sont mis en réseau avec la plupart des installations. Cet ensemble est appelé aujourd'hui **automatisation de locaux.**

- Planification en modules multifonctions: ces petites unités multifonctions forment la structure de base pour toutes les installations et peuvent être facilement combinées avec les locaux concernés. Ceci permet de réaliser une gestion souple de surface, la séparation des locaux est effectuée par des parois légères sans installation. Lors d'une modification de la disposition des locaux, l'adaptation de la gestion technique se fait par simple reprogrammation. De plus en plus, cette dernière est assurée par des outils, accessibles aussi aux non-spécialistes, qui règlent automatiquement et en arrière-plan les nouveaux rapports fonctionnels correspondants entre les appareils de bus.

- Utilisation rationnelle des énergies naturelles (lumière du jour, chaleur solaire, refroidissement pendant la nuit, corps du bâtiment comme accumulateur de chaleur et de froid – systèmes d'éléments de construction thermoactifs TABS – etc.) et aération et climatisation décentralisées.

- Capteurs tels que les détecteurs de présence qui surveillent étroitement les besoins réels en énergie pour les signaler à toutes les installations.

- L'utilisateur dispose d'une manipulation confortable de toutes les installations afin d'adapter l'éclairage, l'ombrage et la climatisation locale à ses besoins. Les commandes s'effectuent par l'écran du PC sur la place du travail.

- Visualisation des procédés locaux sur l'outil centralisé de gestion de bâtiment (appelé «SCADA – Supervisory Control and Data Acquisition») permettant une intervention rapide en cas de défaillance, l'enregistrement à long terme des valeurs d'énergie, la gestion de la maintenance et, plus généralement, l'optimisation des procédés. Ainsi, l'automatisation de bâtiments devient un outil indispensable de gestion d'exploitation.

Figure I.2: Habiter intelligent

5. Exemple des bâtiments intelligents :

5.1. The Senset Millennium Building :

The Senset Millennium Building à Hollywood, Californie, est un bâtiment construit il y'a 20 ans a abordé le problème d'un système de surveillance inefficace d'oxyde de carbone en installant des systèmes autonome de contrôle pour garder ses ventilateurs du fonctionnement sans interruption. Précédemment, le propriétaire de bâtiment payait environ $242.000 pour garder les ventilateurs pour le système de surveillance courant 24h x7j. L'installation des systèmes des bâtiments intelligents a aidé à réduire la consommation d'électricité d'un ordre de 90 %

Figure I.3: Le Sunset Millennium Building

5.2. Connolly Middle School:

Le Collège de Connolly à Tempe, Ariz. A été Construit en 1972, il a adopté une modification de commande de climat qui a été Automatisée avec LonWorks, fournissant plus de contrôle des températures, des niveaux d'anhydride carbonique et ainsi de suite. Le système a sauvé des $58.000 environ en coûts d'électricité en six premiers mois.

Figure I.4: Connolly Middle School

5.3. Le bâtiment 1120 de Vermont :

1120 Vermont Ave, est un bâtiment du nord-ouest de la ville de Washington, il y'a 20 ans de ça il a été équipé de 700 Automates programmables qui sont liés à l'outil de gestion du réseau pour commander les réfrigérateurs, les pompes, les tours de refroidissement, les chaudières et les générateurs à l'intérieur. L'épargne annuelle dans la consommation d'énergie est de $500.000.

Figure I.5: 1120 Vermont Ave. in Washington.

6. La Solution Schneider pour le bâtiment intelligent [4] :

Schneider Electric est le spécialiste mondial de la gestion de l'énergie, propose des solutions intégrées pour rendre l'énergie sûre, fiable, efficace et productive sur les marchés du résidentiel, des bâtiments, des centres de données et réseaux, de l'industrie et de l'énergie et des infrastructures. Avec un chiffre d'affaires de 17,3 milliards d'euros en 2007, les 120 000 collaborateurs du Groupe dans 102 pays vous aident à tirer le meilleur de votre énergie.

Schneider Electric a prouvé qu'il est possible de réduire significativement les dépenses énergétiques des bâtiments. La diminution des frais d'exploitation induit non seulement une augmentation du bénéfice d'exploitation à court terme mais également un meilleur retour sur investissement pour les promoteurs

6.1. Mesurer, analyser, sensibiliser :

Pour la mesure Schneider Electric offre une gamme de produits de mesure électrique pour la consommation et la qualité elle assure des solutions adaptées à chaque type d'installation électrique ainsi que des logiciels pour élaborer les tableaux de bord

Perspectives

- Solution adaptée pour le résidentiel
- Une mesure toutes énergies
- Diagnostic avancé

Figure I.6 : Les Appareils de mesure

6.2. Réaliser des fonctions de contrôle simple :

On peut les classer dans quatre catégories :

- Mono-application, kit simple de produits, option ‘sans fil ajouté ’
- Commande automatique d’éclairage : présence, luminosité
- Régulation thermique
- Commandes centralisées, scènes, programmation temporelle

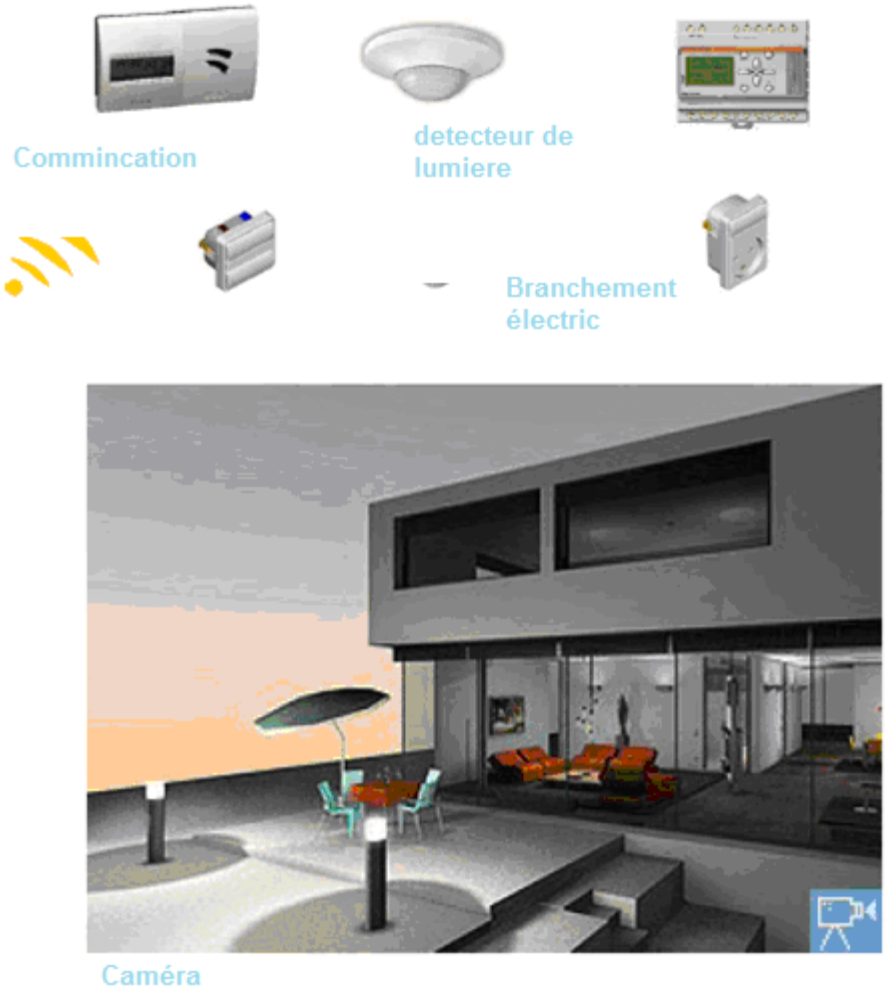


Figure I.7: les équipements Intelligents

6.3. Concevoir des systèmes de contrôle avancés et de supervision :

Afin de donner une certaine autonomie aux bâtiments intelligents et assurer une gestion technique complète, la société Schneider Electric a conçu des matériels d'automatisation, en l'occurrence les automates programmables industriels, ainsi des outils pour la programmation tel que le logiciels Unity Pro, et de supervision tel que le logiciel Vijeo Citect.

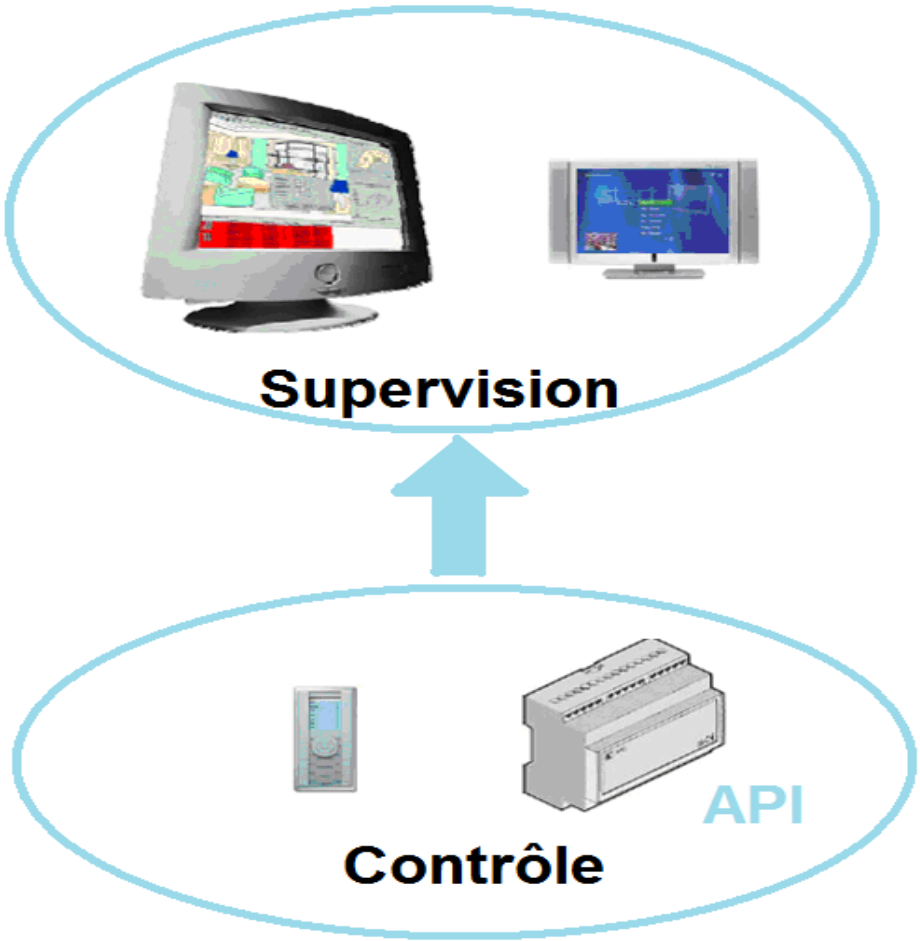


Figure I.8: Système de contrôle et de supervision

7. Architecture de communication [10] :

Quand on parle d'Architecture, on se réfère à une structure d'éléments définissant un système complexe. Dans le langage courant, l'architecture est "l'art de concevoir et de construire un bâtiment selon des règles techniques" (Le Petit Larousse). Pour un informaticien, il est fait souvent référence à l'architecture du calculateur, ensemble structuré d'éléments électroniques et logiques.

L'Architecture de Communication définit l'ensemble des entités nécessaires à la Communication ainsi que les règles régissant les échanges entre elles. On parle aussi d'Architecture de Réseau.

7.1. Les services offerts par une architecture de communication :

- Transmission physique;
- Contrôle d'erreurs;
- Routage;
- Régulation de flux (congestion);
- Séquencement;

- Contrôle de bout en bout;
- Gestion du dialogue;
- Reprise sur incidents;
- Transformation de l'information (codage, compression, cryptage...);
- Synchronisation des processus;

7.2. Définition d'un réseau :

Un réseau est un ensemble de matériels et de logiciels permettant à des équipements de communiquer entre eux.

L'objectif d'un réseau est le partage des ressources matérielles (disques durs, imprimantes) et des ressources logicielles (fichiers, applications).

Les réseaux regroupent un ensemble hétérogène d'architectures, du filaire au sans-fil, du LAN au WAN.

Les principales topologies de réseaux existantes sont :

- en étoile
- en bus
- en anneau
- maillé

Ces éléments de base sont combinés pour former des réseaux complexes.

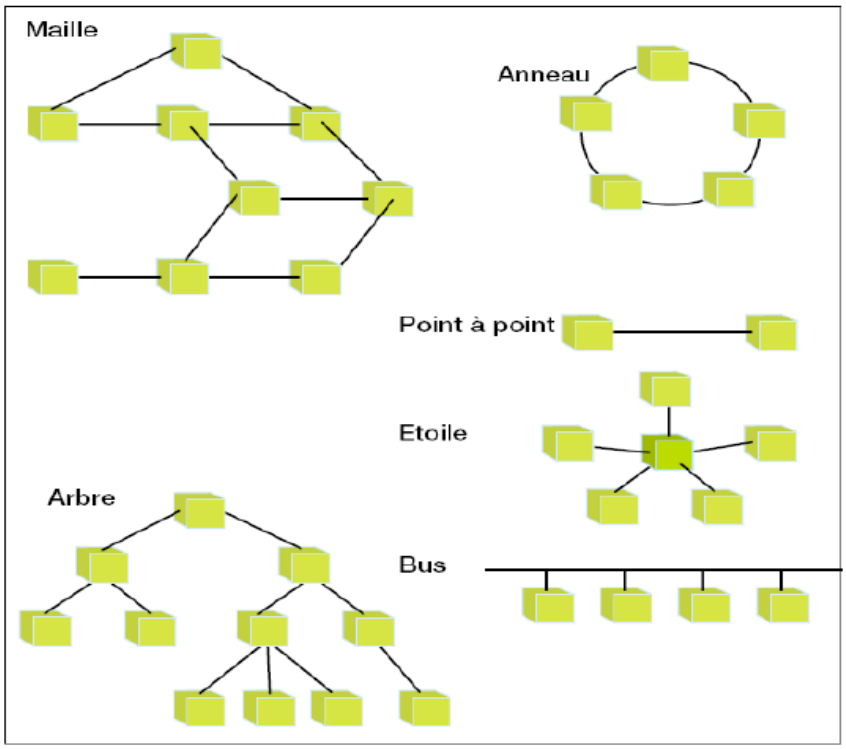


Figure I.9 : Représentation des différentes topologies de réseaux.

7.2.1. Le modèle OSI :

Le modèle OSI (de l'anglais Open Systems Interconnection, « Interconnexion de systèmes ouverts ») d'interconnexion en réseau des systèmes ouverts est un modèle de communications entre ordinateurs proposé par l'ISO (Organisation internationale de normalisation). Il décrit les fonctionnalités nécessaires à la communication et l'organisation de ces fonctions.

7.2.1.1. Architecture en couches :

Le modèle comporte 7 couches succinctement présentées ci-dessous de bas en haut et détaillées dans leur articles respectifs. Ces couches sont parfois réparties en 2 groupes.
Les 4 couches inférieures sont plutôt orientées communication et sont typiquement fournies par un système d'exploitation

Les 3 couches supérieures sont plutôt orientées application et plutôt réalisées par des bibliothèques ou un programme spécifique. Dans le monde IP, ces 3 couches sont rarement distinguées. Dans ce cas, toutes les fonctions de ces couches sont considérées comme partie intégrante du protocole applicatif.

Par ailleurs, les couches basses sont normalement transparentes pour les données à transporter, alors que les couches supérieures ne le sont pas nécessairement, notamment au niveau présentation.

Dans une telle architecture, une « entité » de niveau (N+1) envoie des données avec la primitive « data.request » de l'entité de niveau (N) en lui fournissant comme données un (N+1)-PDU qui sera typiquement, à son tour encapsulé dans un (N)-PDU. Côté récepteur, chaque entité analyse l'enveloppe protocole correspondant à sa couche et transmet les données à la couche supérieure sous la forme d'une primitive« data.indication ».
Certaines fonctions comme la détection des erreurs de transmission et leur correction, le contrôle de flux peuvent être présent dans plusieurs couches. Ces fonctions sont décrites globalement plus loin.

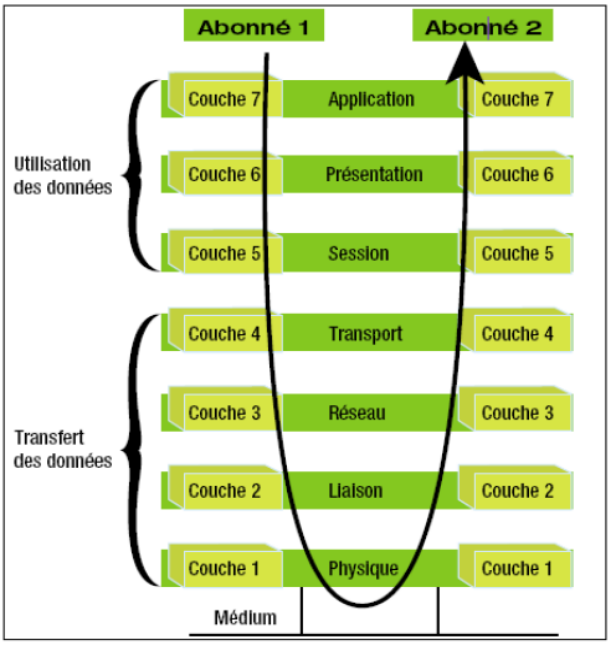


Figure I.10 : Le model OSI

7.2.1.2. Caractérisation résumée des couches :

La caractérisation donnée ici est tirée du chapitre 7 de ISO 7498-1. La description originelle donne en plus pour chaque couche les fonctions de manipulation de commandes ou de données significatives parmi celles décrites plus bas.

- La couche « physique » est chargée de la transmission effective des signaux entre les interlocuteurs. Son service est typiquement limité à l'émission et la réception d'un bit ou d'un train de bit continu (notamment pour les supports synchrones).
- La couche « liaison de données » gère les communications entre 2 machines adjacentes, directement reliées entre elles par un support physique.
- La couche « réseau » gère les communications de proche en proche, généralement entre machines : routage et adressage des paquets (cf. note ci-dessous).
- La couche « transport » gère les communications de bout en bout entre processus (programmes en cours d'exécution).
- La couche « session » gère la synchronisation des échanges et les « transactions », permet l'ouverture et la fermeture de session.
- La couche « présentation » est chargée du codage des données applicatives, précisément de la conversion entre données manipulées au niveau applicatif et chaînes d'octets effectivement transmises.
- La couche « application » est le point d'accès aux services réseaux, elle n'a pas de service propre spécifique et entrant dans la portée de la norme.

7.2.1.3. Les Protocoles :

Les protocoles de communication sont les conventions qui permettent aux outils de transfert de l'information d'interagir de façon efficace, uniforme et sécuritaire.

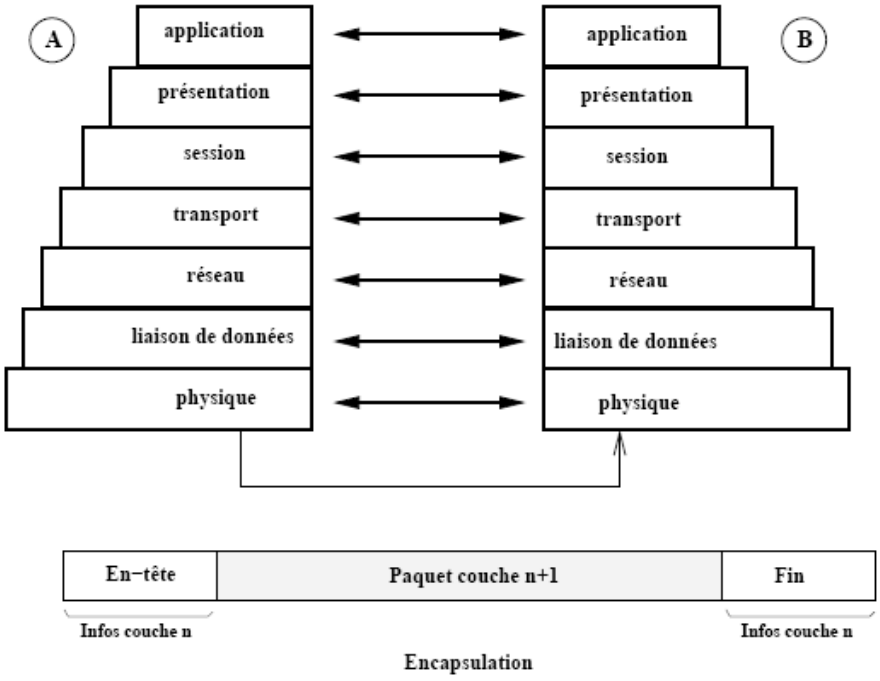


Figure I.11 : Les Protocoles

7.3. Bus et Réseaux de terrain :

7.3.1. Les bus de terrain :

Pour diminuer les coûts de câblage des entrées / sorties des automates (systèmes étendus), sont apparus les bus de terrains. L'utilisation de blocs d'entrées / sorties déportés à permis tout d'abord de répondre à cette exigence.

Avant :

Les capteurs / préactionneurs distants impliquaient de grandes longueurs de câbles.
Avec l'avènement des ASICs, les capteurs, détecteurs ... sont devenus "intelligents" et ont permis de se connecter directement au bus (médium).

Pour assurer le "multiplexage" de toutes les informations en provenance des capteurs / préactionneurs ont été développés plusieurs protocoles de communication et des standards sont apparus (normalisés ou standards de fait).

Exemple :

Le bus ASi (Actuators Sensors interface) est un bus de capteurs/actionneurs de type Maître / Esclave qui permet de raccorder 31 esclaves (capteurs ou préactionneurs) sur un câble spécifique (deux fils) transportant les données et la puissance.

Ce bus est totalement standardisé et permet d'utiliser des technologies de plusieurs constructeurs (interopérabilité). L'automate est pour cela doté d'un coupleur ASi.

1ère évolution :

Les interfaces d'entrées/ sorties sont déportées au plus près des capteurs.

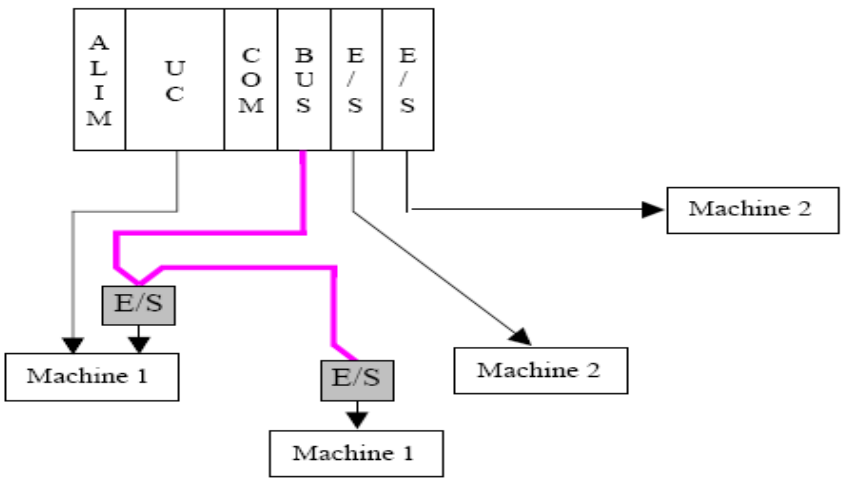


Figure I.12: 1ère évolution des Bus de communication

Aujourd'hui :

Les capteurs et les prés actionneurs "intelligents" (IHM, variateurs, distributeurs ...) permettent la connexion directe au bus.

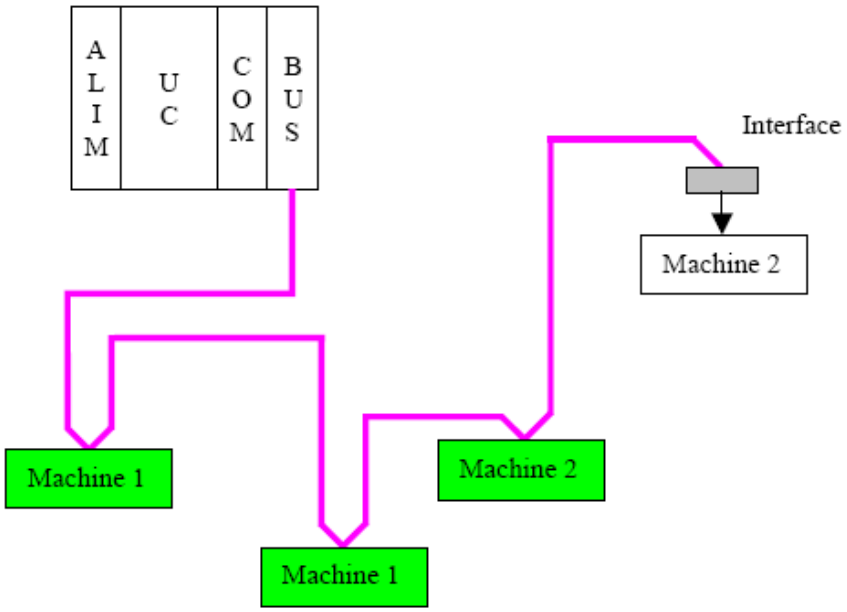


Figure I.13 : Les Bus

7.3.1.1. Avantages et inconvénients des bus de terrain :

Avantage :

- Réduction des coûts de câblage et possibilité de réutiliser le matériel existant
- Réduction des coûts de maintenance
- Possibilités de communication

Inconvénients :

- Taille du réseau limitée
- Adaptabilité aux applications à temps critique
- Coût global

Autres bus de terrain : Batibus (norme EIB), Interbus-S, CANopen

7.3.2. Les réseaux de terrain :

L'émergence de ces nouvelles technologies à conduit à la définition de plusieurs catégories de réseaux locaux industriels (pyramide CIM) :

- les réseaux de terrain,
- les réseaux de cellule,
- les réseaux de supervision et de commande

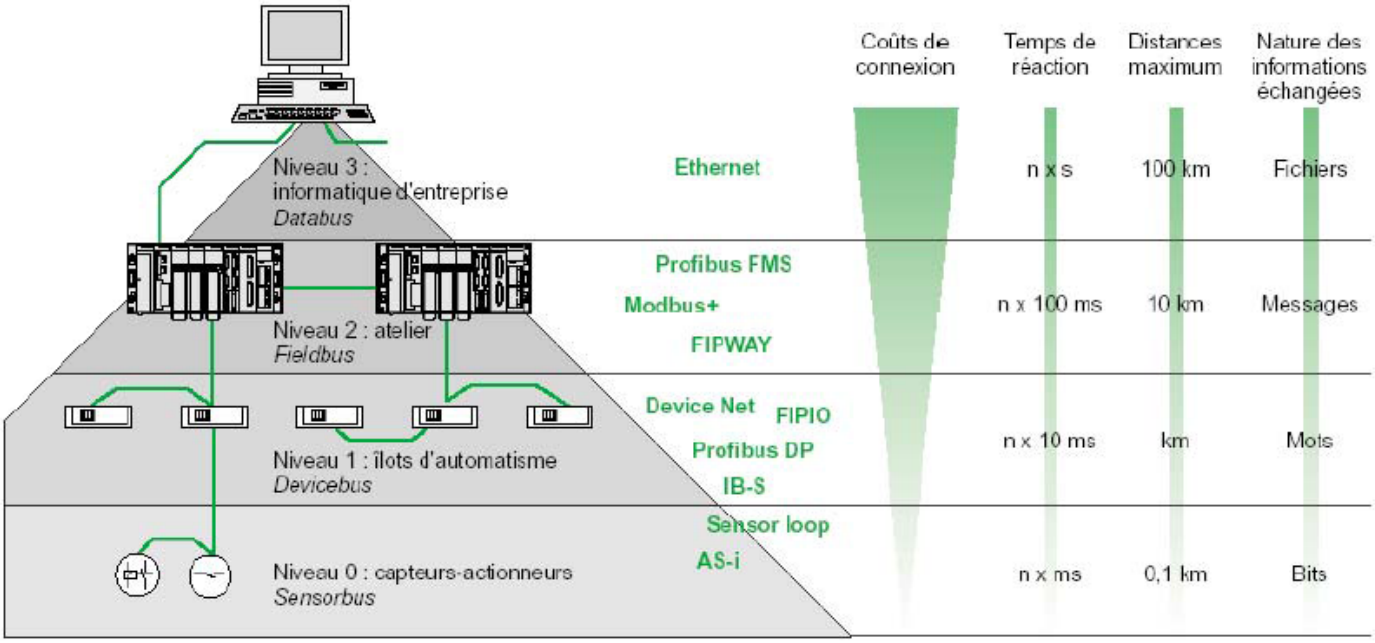


Figure I.14 : Caractéristiques des niveaux de réseaux

La nécessité de communication entre cellules (communication entre automates) a permis de voir apparaître de nombreuses normes de communication (Profibus, Fip ...).
Le déterminisme nécessaire pour certaines applications conduit à l'utilisation de réseaux Maître / Esclave.

Au niveau de l'entreprise, le temps n'est plus critique et la norme Ethernet à pu se développer rapidement, permettant ainsi la visualisation et la commande des process via le réseau Internet.

La tendance actuelle est à l'introduction des réseaux Ethernet au plus près des automatismes (exemple : norme Profinet).

8. Les protocoles Ethernet / Modbus :

8.1. Ethernet TCP/IP :

Ethernet a été spécifié dans les années 70 par Digital, Intel et Xerox)

Un réseau Ethernet est un réseau dont les équipements informatiques sont équipés d'une interface Ethernet**,** sur ce réseau, chaque station est identifiée par une adresse sur 6 octets. Ces adresses sont notées en hexadécimal. Par exemple 08:80:D3:A0:18:43.Cette adresse est statique et inscrite "en dur" sur la carte interface réseau.

Les trames sont diffusées sur l'ensemble du câble et sont "absorbées" par les extrémités munis de résistances.

Tous les postes sont en permanence à l'écoute des trames qui passent sur le câble, ils vérifient l'intégrité du paquet et analysent l'adresse du destinataire, ils transmettent au niveau supérieure du poste si la trame lui est destiné.

La méthode d'accès utilisé par Ethernet est appelée CSMA/CD pour : Carrier Sense Multiple Acces with Collision Detection (Accès Multiple avec Écoute de Porteuse avec Détection de Collision.

Les trames ont une longueur variable comprise entre 72 et 1526 octets, et la composition d'une trame Ethernet a le format suivant :

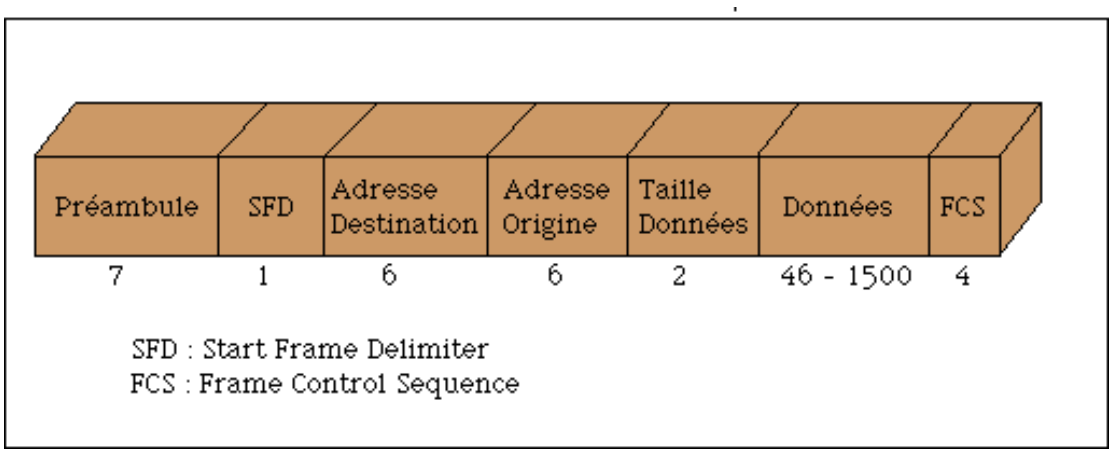


Figure I.15 : Composition d'une Trame

La vitesse de circulation des trames sur le câble est de 10 Mbits/s donc 10 bits/micro-secondes.

8.1.1. La détection de collision :

Une collision a lieu quand 2 trames circulent en même temps sur le câble. Dans ce cas les signaux électriques qui sont diffusés deviennent perturbés, donc inexploitables. Le principe est qu'avant d'émettre, la station vérifie si le câble est "libre" (si son état électrique indique qu'aucune trame ne circule), quand la station émet sa trame, il continue à vérifier si un autre trame circule sur le câble et ce pendant 51,2 micro-secondes.

Si la station détecte une autre trame, il stoppe la diffusion de sa propre trame qui ne fera donc pas la longueur "réglementaire" de 72 octets (en 51,2 micro-secondes la station n'a pu émettre plus de 64 octets). Pour éviter que cette trame soit trop courte, la station ajoute des signaux réguliers (des 1 et des 0) . Cette trame particulière dont la longueur maximale ne peut dépasser 64 octets sera donc automatiquement détectée comme étant le résultat d'une collision par toutes les autres stations du réseau.

Le but est donc de mettre en place une architecture de réseau ou on sera certain qu'une trame se sera propagée sur l'ensemble du câble en moins de 50 microsecondes. C'est ce facteur qui détermine la longueur maximum du réseau selon le type de câbles utilisés.

Toutes ces limitations sont liées à la technique CSMA/CD (Carrier Sense Multiple Access with Collision Detection) et à la vitesse de propagation des signaux électriques sur les médias ou dans les constituants, ainsi qu'aux phénomènes de réflexion.

Il existe une multitude de protocoles associables à Ethernet. Deux sont devenus des standards, à savoir : TCP et IP.

8.1.2. IP (Internet Protocol) :

IP est le protocole principal de la couche réseau qui est utilisé à la fois par TCP, UDP, ICMP et IGMP. Une application peut également accéder directement à IP (rare mais possible).

Chaque bloc de données qui circule sur l'Internet traverse la couche IP de tous les hôtes en extrémités du réseau ou routeurs intermédiaires. Il assure le routage des messages qui est direct si le destinataire est sur le même réseau ou indirect via routeur ou passerelle.

Le service est non fiable, il n'existe aucune garantie que le datagramme arrive à destination. Il fournit un service qualifié de "au moindre effort" ou "au mieux".

En cas de saturation de buffers, IP ne sait que rejeter un datagramme et essayer de prévenir l'émetteur via un message ICMP.

La fiabilité doit être assurée par les couches supérieures.

Exemple :

En fonction de la taille du réseau, trois classes d'adresses peuvent être utilisées :

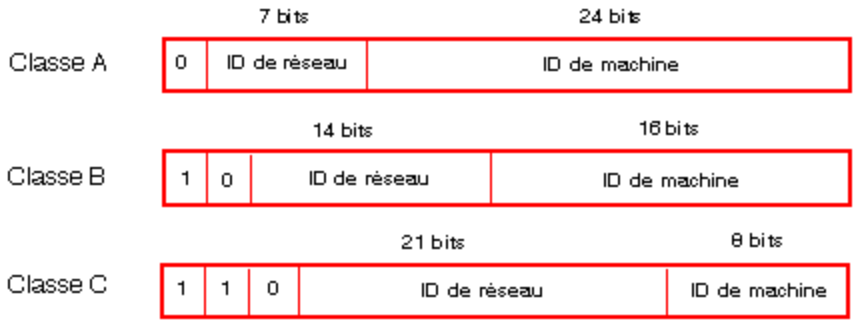


Figure I.16 : les classe de masques de sous réseau

Espaces réservés pour les différentes classes d'adresses IP :

Classe	**Plage**
A	0.0.0.0 à 127.255.255.255
B	128.0.0.0 à 191.255.255.255
C	192.0.0.0 à 223.255.255.255

- Les adresses de classe A sont destinées aux réseaux de grande échelle dotés d'un nombre important de sites connectés.
- Les adresses de classe B sont destinées aux réseaux de moyenne échelle dotés d'un nombre moindre de sites connectés.
- Les adresses de classe C sont destinées aux réseaux de petite taille dotés d'un faible nombre de sites connectés.

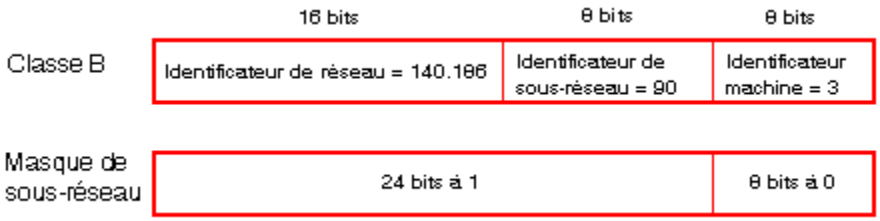
8.1.3. Sous-adressage et masque de sous-réseau :

Une adresse IP est composée de deux identificateurs, un qui identifie le réseau et l'autre qui identifie la machine connectée. En réalité, l'identificateur de la machine peut également contenir un identificateur de sous-réseau.

Dans un environnement ouvert, ayant reçu un identificateur de l'autorité appropriée, l'administrateur du système local a la possibilité de gérer plusieurs réseaux. Cela signifie que des réseaux locaux peuvent être installés sans affecter le monde extérieur, qui ne voit qu'un seul réseau désigné par son identificateur.

Le masque de sous-réseau permet de visualiser le nombre de bits attribués respectivement à l'identificateur du réseau et à l'identificateur du sous-réseau (bits à 1), puis à l'identificateur de la machine (bits à 0).

Exemple : 140.186.90.3



La segmentation permet d'obtenir 254 sous-réseaux possibles avec 254 machines de sous-réseau, la valeur du masque de sous-réseau doit être choisie afin d'être cohérente avec la classe d'adresses IP.

Le masque de sous-réseau aura la valeur suivante :

- Pour une adresse de classe A : 255.xxx.xxx.xxx,
- Pour une adresse de classe B : 255.255.xxx.xxx,
- Pour une adresse de classe C : 255.255.255.xxx,

xxx est une valeur arbitraire qui peut être choisie par l'utilisateur.

8.1.4. TCP (Transmission Control Protocol) :

TCP est destiné à être implémenté sur la couche transport du modèle OSI. C'est un protocole de transport fiable orienté connexion et flux de données. TCP se charge de traiter la

non fiabilité d'IP.

L'offre Ethernet TCP/IP de Schneider Electric implémente sur la couche application les protocoles applicatifs natifs que sont UNI-TE et Modbus.

Ceci afin de satisfaire aux besoins de dialogue d'application à application et ainsi assurer l'interopérabilité des différentes plates-formes automates.

De nombreux services sont disponibles sur les coupleurs Ethernet de Schneider Electric, à savoir :

- applets de diagnostic embarqués afin de pouvoir assurer un diagostic de l'automate et de sa configuration de manière simple et transparente au travers d'un navigateur Internet,
- applet d'éditeur de données dont l'objectif est de pouvoir accéder à la base de données (protégée par mot de passe) de l'automate par un navigateur Internet,
- applet d'éditeur graphique permettant de visualiser et commander, depuis un navigateur Internet, sous forme graphique (paragraphes, curseur, rotacteur, afficheur, courbes...) les données du procédé,
- applet "diag viewer" autorisant le report d'alarmes générées par un Premium et d'en gérer les acquis. Ceux-ci seront distribués sur le réseau Ethernet TCP/IP pour ainsi en faire bénéficier la totalité des acteurs. Diag viewer est accessible par un navigateur Internet.

Ces fonctions sont fournies avec les coupleurs concernés et ne nécessitent aucun développement.

De plus, certains coupleurs disposent d'une mémoire utilisateur permettant le chargement de pages HTML et ainsi autoriser la conception d'un dialogue homme/machine temps réel embarqué.

8.2. PROTOCOLE MODBUS :

Ce protocole, développé par la société MODICOM, bien que datant de plusieurs années, reste un des protocoles les plus répandus dans le domaine des réseaux de terrain, il est indépendant de l'interface électrique. Un équipement MODBUS peut donc utilisé une E/S série V24, V11, RS485 ou fibre optique.

8.2.1. Définition :

Le protocole MODBUS est un protocole de transmission de données régissant le dialogue entre une station « Maître » et des stations « Esclaves ».

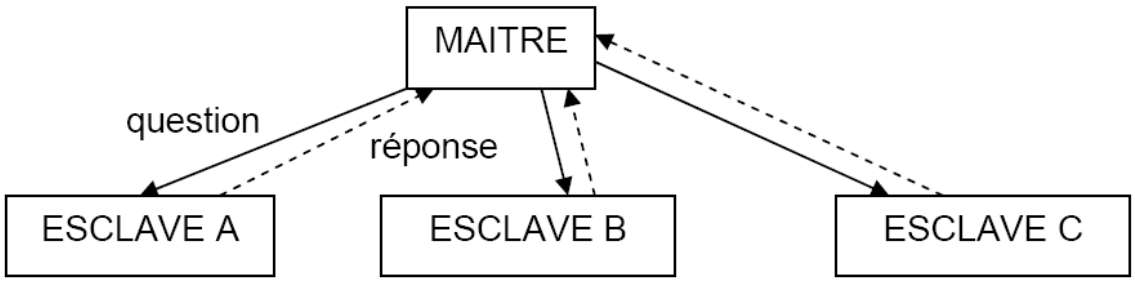
L'échange Maître-Esclave s'effectue par l'envoi de trames MODBUS/JBUS dont le format de base est le suivant :

Adresse	Requête	Données	CRC16

L'adresse correspond à l'adresse de la station Esclave destinataire de la requête. La requête ou code fonction détermine le type de commande (lecture mot, écriture mot, etc..). Le champ de données contient l'ensemble des paramètres et informations liés à la requête. Le contrôle de redondance cyclique (CRC16) permet à la station destinatrice de vérifier l'intégrité de chaque trame.

A chaque réception d'une trame, la station adressée envoie une trame de réponse, dont le format est identique à celui de la trame émise par la station Maître avec selon le type de commande un champ de données plus ou moins important.

Le protocole MODBUS consiste en la définition de trames d'échange.



Le maître envoie une **demande** et attend une **réponse**.

Deux esclaves ne peuvent dialoguer ensemble.

Le dialogue maître – esclave peut être schématisé sous une forme successive de liaisons point à point.

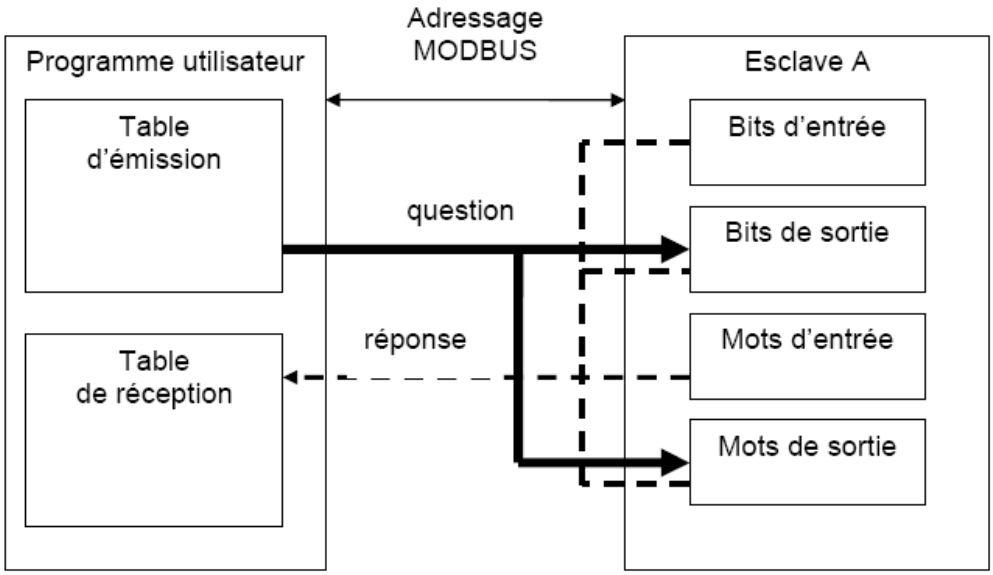


Figure I.17: Principe des échanges MODBUS

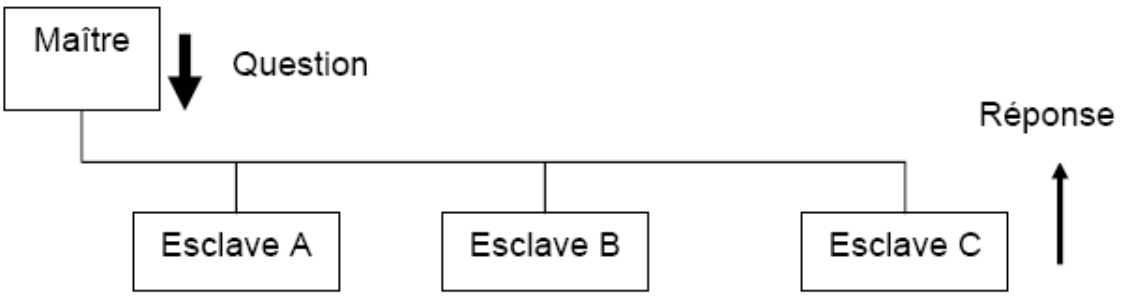
8.2.2Adressage :

Les abonnés du bus sont identifiés par des adresses attribuées par l'utilisateur.
L'adresse de chaque abonné est indépendante de son emplacement physique.
Les adresses vont de 1 à 64 et ne doivent pas obligatoirement être attribuées de manière séquentielle.

Deux abonnés ne peuvent avoir la même adresse.

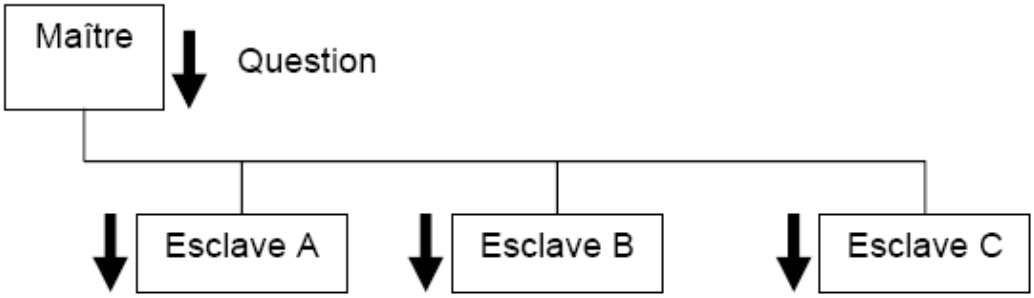
8.2.3 Echange maître vers 1 esclave :

Le maître interroge un esclave de numéro unique sur le réseau et attend de la part de cet esclave une réponse.



8.2.4. Echange Maître vers tous les esclaves :

Le maître diffuse un message à tous les esclaves présents sur le réseau, ceux-ci exécutent l'ordre du message sans émettre une réponse.



8.2.5. Trame d'échange question/réponse :

La question :

Elle contient un code fonction indiquant à l'esclave adressé quel type d'action est demandé.
Les données contiennent des informations complémentaires dont l'esclave a besoin pour exécuter cette fonction.

Le champ octets de contrôle permet à l'esclave de s'assurer de l'intégralité du contenu de la question.

N° d'esclave	Code fonction	Information spécifique concernant la demande	Mot de contrôle
1 octet	1 octet	n octets	2 octets

La réponse

N° d'esclave	Code fonction	Données reçues	Mot de contrôle
1 octet	1 octet	n octets	2 octets

Si une erreur apparaît, le code fonction est modifié pour indiquer que la réponse est une réponse d'erreur.

Les données contiennent alors un code (code d'exception) permettant de connaître le type d'erreur.

Le champ de contrôle permet au maître de confirmer que le message est valide.

N° d'esclave	Code fonction	Code d'exception	Mot de contrôle
1 octet	1 octet	1 octet	2 octets

Code	Nature des fonctions MODBUS	TSX 37
H'01'	Lecture de n bits de sortie consécutifs	*
H'02'	Lecture de n bits de sortie consécutifs	*
H'03'	Lecture de n mots de sortie consécutifs	*
H'04'	Lecture de n mots consécutifs d'entrée	*
H'05'	Ecriture de 1 bit de sortie	*
H'06'	Ecriture de 1 mot de sortie	*
H'07'	Lecture du statut d'exception	
H'08'	Accès aux compteurs de diagnostic	
H'09'	Téléchargement, télé déchargement et mode de marche	
H'0A'	Demande de CR de fonctionnement	
H'0B'	Lecture du compteur d'événements	*
H'0C'	Lecture des événements de connexion	*
H'0D'	Téléchargement, télé déchargement et mode de marche	
H'0E'	Demande de CR de fonctionnement	
H'0F'	Ecriture de n bits de sortie	*
H'10'	Ecriture de n mots de sortie	*
H'11'	Lecture d'identification	*
H'12'	Téléchargement, télé déchargement et mode de marche	
H'13'	Reset de l'esclave après erreur non recouverte	

Figure I.18: Services supportés par MODBUS

8.2.6. Topologie réseau MODBUS :

Les réseaux de terrain, utilisant le protocole MODBUS ont une configuration multipoint, principalement à base de liaisons RS485.

Le réseau Modbus est un réseau composé d'un seul Maître et d'Esclaves, qui communiquent par une liaison série asynchrone:

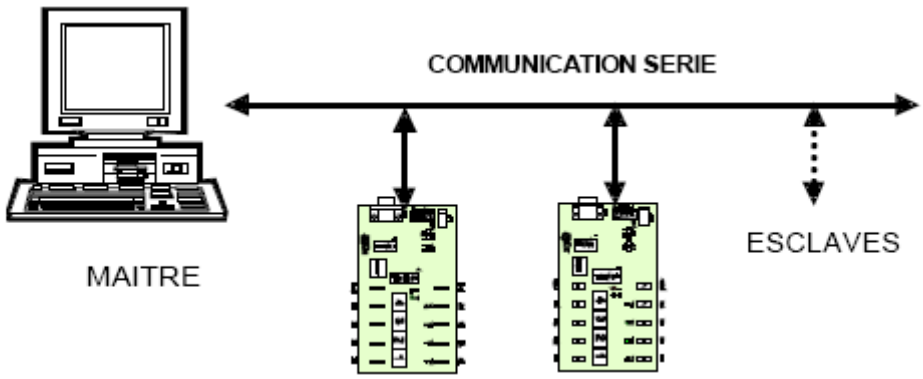


Figure I.19 : Communication série asynchrone

Le support électrique de la liaison série suit une des deux normes courantes. La liaison type RS232 (+12V,-12V), la seule disponible sur PC, n'est utilisable que dans le cas d'un réseau avec un seul esclave. Cette liaison sera utilisée pour l'initiation au fonctionnement du réseau.

9. Le protocole Modbus sur TCP/IP :

Le protocole Modbus sur TCP/IP est défini comme suit.

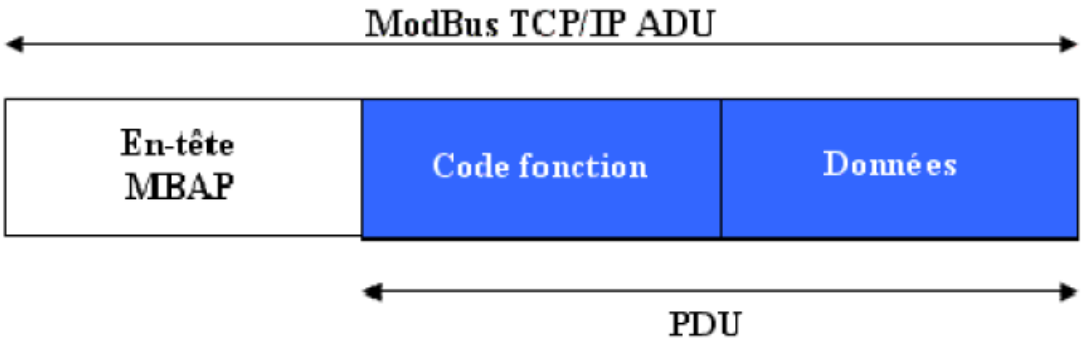


Figure I.20 : Protocole Modbus TCP/IP

ADU : Application Data Unit
PDU : Protocol Data Unit
Toutes les requêtes et réponses de Modbus sont conçues de telle manière que le destinataire puisse vérifier qu'un message est fini.

10. La stratégie de communication de Schneider :

Elle est basée sur des normes ouvertes (coeur de l'offre) comme les suivantes :

- Ethernet Modbus TCP/IP,
- CANopen,
- AS-Interface,
- Modbus Link Series.

Cela n'a pas toujours été le cas et on compte un grand nombre de bases installées sur des réseaux ou bus propriétaires tels que Modbus Plus, Fipway, Ethway, X-Way sur TCP/IP, Fipio, Symax et Uni-telway.

10.1. Architecture globale :

Le schéma suivant présente un exemple d'architecture de communication globale avec bus AS-i :

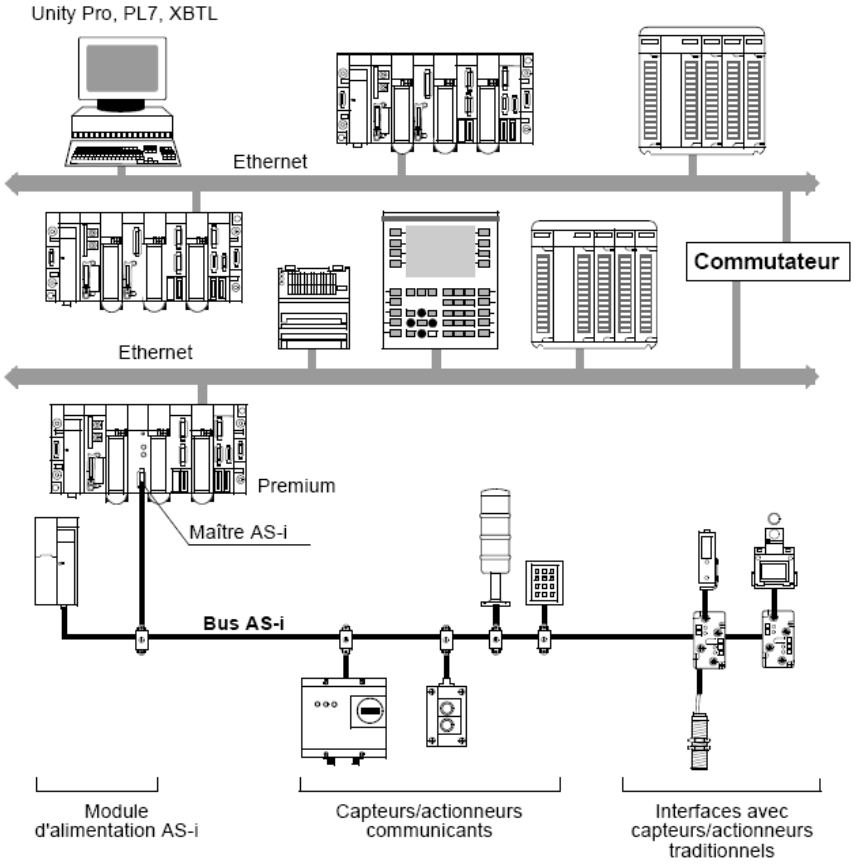


Figure I.21: Architecture Globale.

En fonction du type de réseau utilisé, l'interconnexion est réalisée directement via un automate qui dirige les informations (Ethernet/Uni-Telway) ou via un équipement supplémentaire tel qu'un pont (Ethernet/Modbus) ou un commutateur (Ethernet/Ethernet).

Techniquement parlant, des solutions sophistiquées utilisant Ethernet, Modbus Plus, Fipway, Fipio, Modbus, Uni-Telway, etc., dans une architecture unique sont possibles. Toutefois, pour faciliter la maintenance et la formation de l'utilisateur, ainsi que pour réduire les coûts d'exploitation, il est recommandé de viser une homogénéité maximum entre les types de réseaux et de bus utilisés.

Dans les exemples d'architecture suivants, nous présentons les solutions les plus adaptées en fonction des équipements connectés.

10.2. Architectures de réseau :

Diverses architectures de réseau sont disponibles. La gamme de produits Schneider permet de créer des mono réseaux Ethernet ainsi que des architectures multi réseau transparentes (Ethernet/Fipway/Modbus Plus). Les exemples suivants d'architectures de réseau illustrent les diverses solutions optimales proposées par les produits Schneider.

La sélection d'une architecture avec le réseau Modbus Plus ou Fipway est fortement liée à l'utilisation d'équipements Quantum ou Premium :

- Modbus Plus pour automates Quantum et Premium
- Fipway pour automates Premium

Dans les illustrations suivantes, les flèches indiquent les différentes possibilités de communication.

Nous avons essayé de présenter tous les scénarios disponibles.

Les types de communication illustrés dans les réseaux Ethernet homogènes sont également possibles lorsque ces réseaux sont étendus à l'aide de segments Modbus Plus ou Fipway.

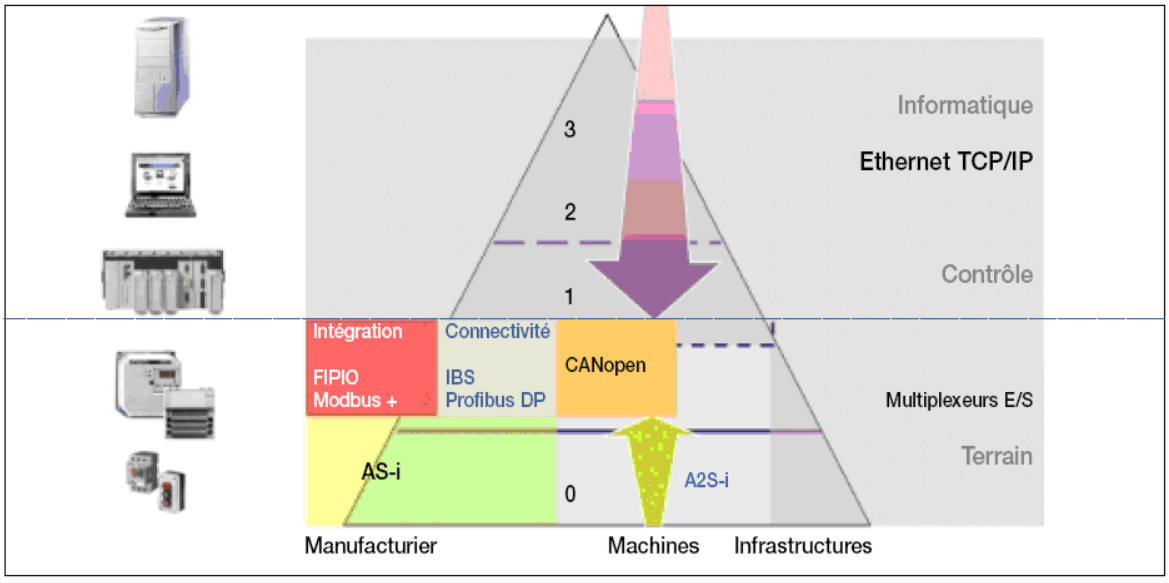


Figure I.22 : Le choix de Schneider Electric en Matière de Réseaux

10.3. Architecture Ethernet :

On distingue deux types d'architecture Ethernet :

- Architecture Mono Réseau.
- Architecture Multi Réseau.

Le schéma ci-dessous illustre ces deux types d'architecture Ethernet ;

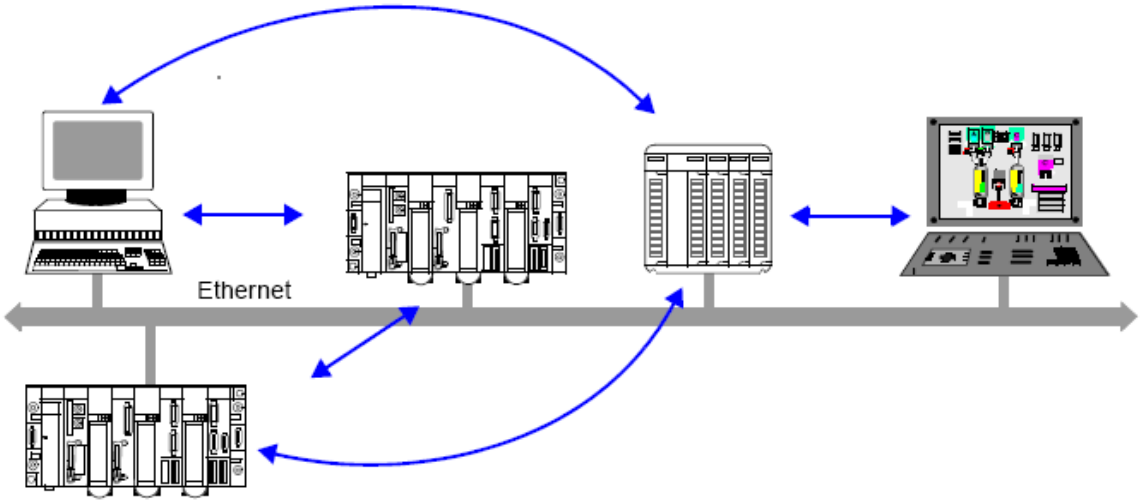


Figure I.23 : architecture Mono Réseau.

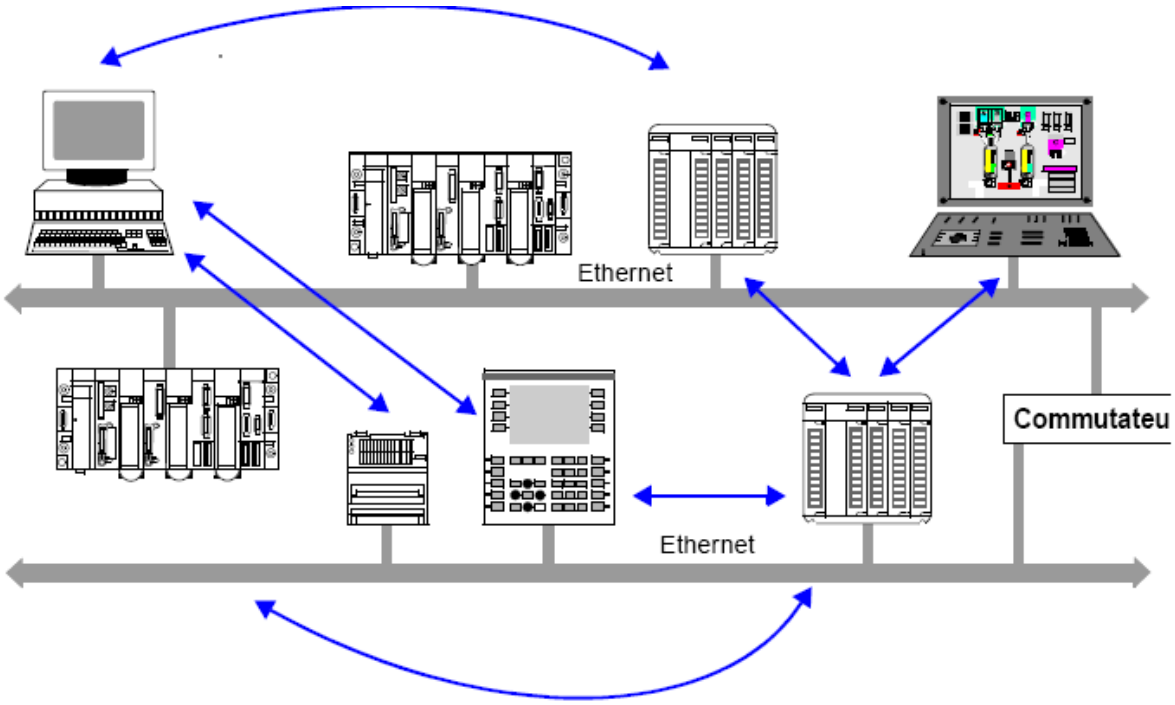


Figure I.24 : Architecture Ethernet multi réseau

10.4. Architecture Ethernet/Modbus multi réseau :

Le schéma ci-dessous illustre une architecture Ethernet/Modbus multi réseau :

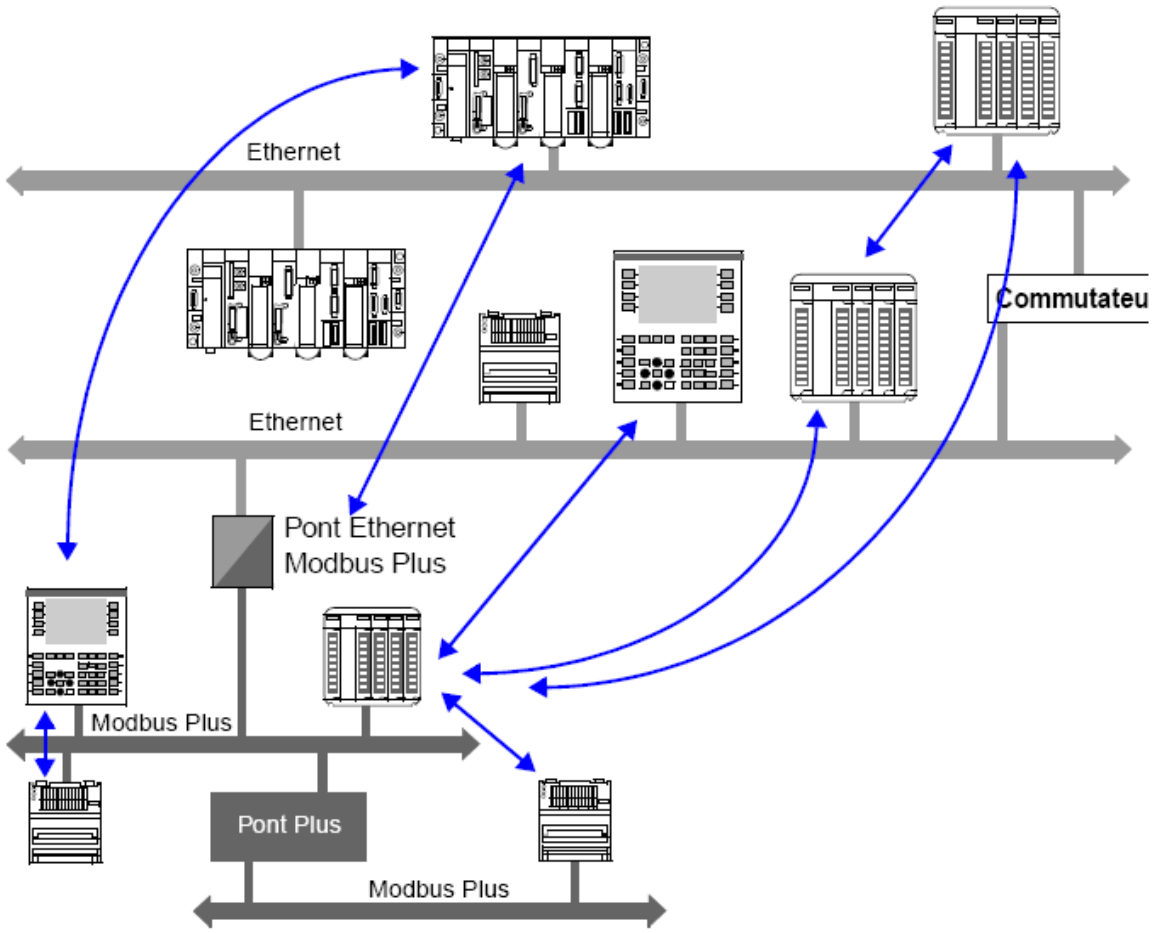


Figure I.25 : Architecture Ethernet/ Modbus Multi Réseau

L'accès est possible à partir d'équipements sur le réseau Modbus Plus via des ponts Ethernet/Modbus Plus. Par contraste, les équipements sur le second réseau Modbus Plus ne sont pas accessibles par un équipement Ethernet via le pont Plus.

Conclusion :

La Gestion Technique de Bâtiment (GTB) est un système informatique généralement installé dans de grands bâtiments ou dans des installations industrielles afin de superviser l'ensemble des équipements qui y sont installés.

CHAPITRE II

UNITY PRO

Présentation :

Le logiciel Unity Pro est un atelier logiciel destiné à programmer les automates Télémécanique Modicon Premium, Modicon Quantum et Modicon Atrium.

Il reprend toutes les valeurs d'usage reconnues des logiciels PL7 et Concept et propose un ensemble complet de nouvelles fonctionnalités pour plus de productivité et d'ouverture vers les autres logiciels.

Les cinq langages IEC6113-3 sont supportés en standard dans Unity Pro avec toutes les fonctions de mise au point, sur le simulateur ou directement en ligne avec l'automate. Il permet de structurer une application pour les plates-formes Atrium, Premium et Quantum en modules fonctionnels composés de :

- Sections (code programme).
- Tables d'animation.
- Ecrans d'exploitation.

1. 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).

2. Fonctions d'Unity Pro [6] :

2.1. Plates-formes matérielles :

- Unity Pro prend en charge les plates-formes matérielles suivantes :
- Modicon M340
- Premium
- Atrium
- Quantum

2.2. 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

2.3. Différentes étapes du processus utilisant Unity Pro :

Le logigramme ci-dessous présente les différentes étapes à suivre pour créer l'application. On doit respecter un ordre chronologique afin de définir correctement tous les éléments de l'application.

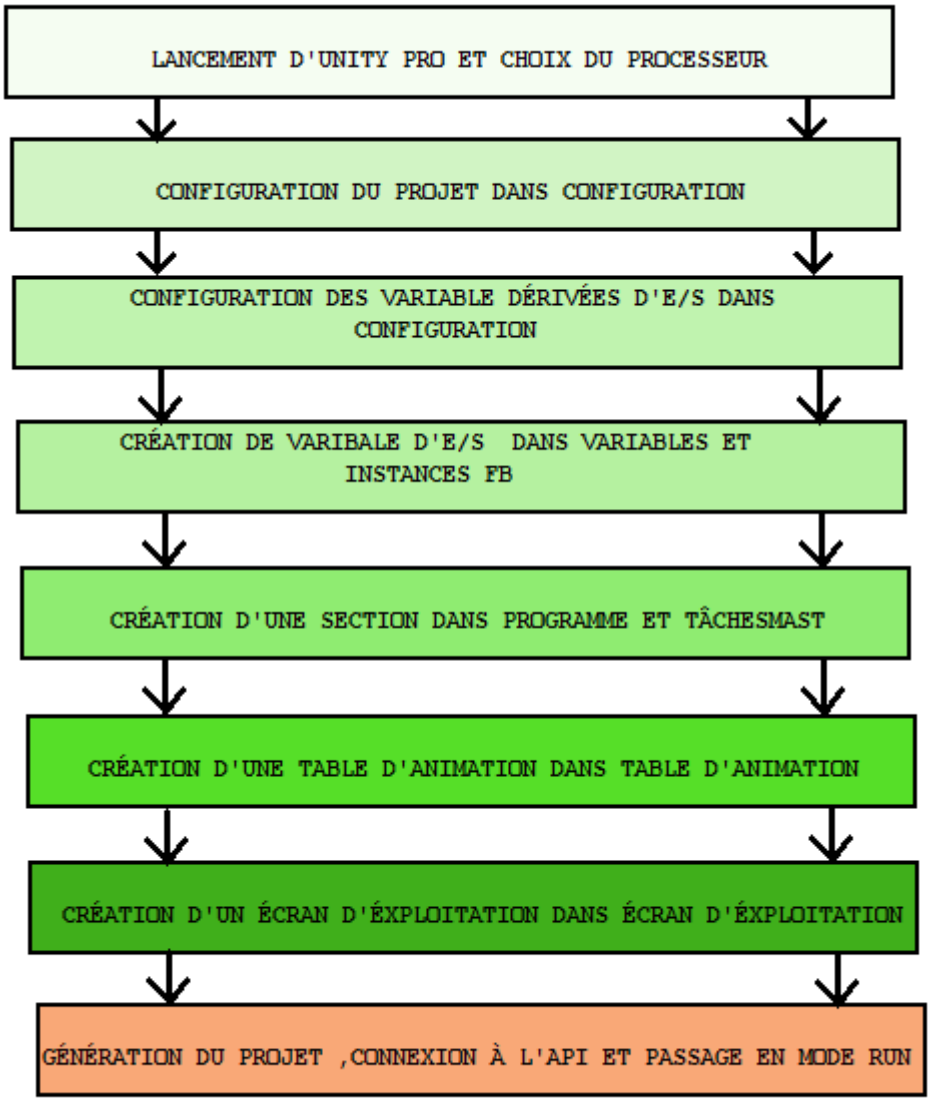


Figure II.1: Différentes étapes pour la création d'un projet sous Unity Pro

3. Interface utilisateur :

Le logiciel Unity Pro, via son écran d'accueil, donne l'accès à l'ensemble des outils proposés selon une ergonomie totalement repensée, profitant ainsi des retours d'expérience des logiciels de conception d'applications Concept et PL7 Junior/Pro.

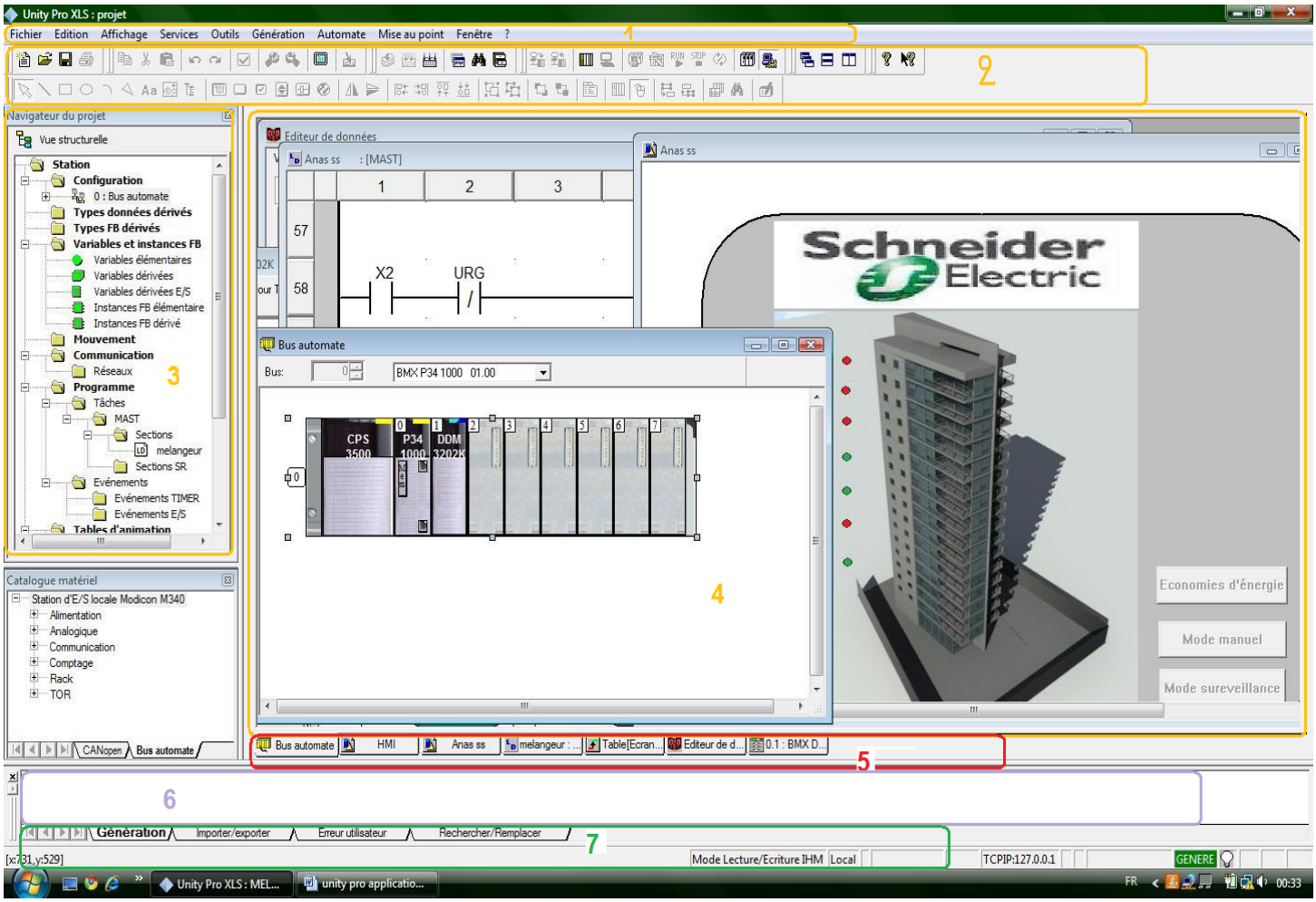


Figure II.2 : Ecran d'éxploitation

Cet écran d'accueil présente une vue générale composée de plusieurs fenêtres et de barres d'outils librement disposables sur la surface de l'écran :

1- Barre de menus, permet l'accès à toutes les fonctions.

2- Barre d'outils composée d'icônes, destinée à l'accès aux fonctions les plus utilisées.

3- Navigateur application, permet de parcourir l'application à partir d'une vue traditionnelle et/ou d'une vue fonctionnelle.

4- Zone fenêtres éditeur, permet de visualiser simultanément plusieurs éditeurs (éditeur de configuration, éditeurs langages à contact, littéral…, éditeur de données).

5- Onglets d'accès direct aux fenêtres éditeur.

6- Fenêtre d'informations liée à des onglets (erreurs utilisateur, import/export, recherche/remplacement…).

7- Ligne d'état.

4. Configuration matérielle :

4.1. Choix du processeur :

Il peut être simple ou double format, il occupe respectivement une ou deux adresses. Il gère l'ensemble d'une station automate constituée de :

- Modules d'entrées/de sorties TOR ;
- Modules d'entrées/de sorties analogiques ;
- Modules métiers (comptage, commande d'axes, commande pas à pas, communication...) qui peuvent être répartis sur un ou plusieurs racks connectés au bus X.

Le nombre d'entrées/sorties pouvant être gérés diffère d'un processeur à un autre, donc il faut savoir le nombre d'entrées/sorties dont on a besoin avant de choisir notre processeur **[Annexe B].**

Le choix du processeur est la première étape à réaliser pour créer une application, pour ce faire il faut :

- Choisir la plate-forme : Premium, Modicom ou Quantum (non interchangeable)
- Choisir le type de processeur

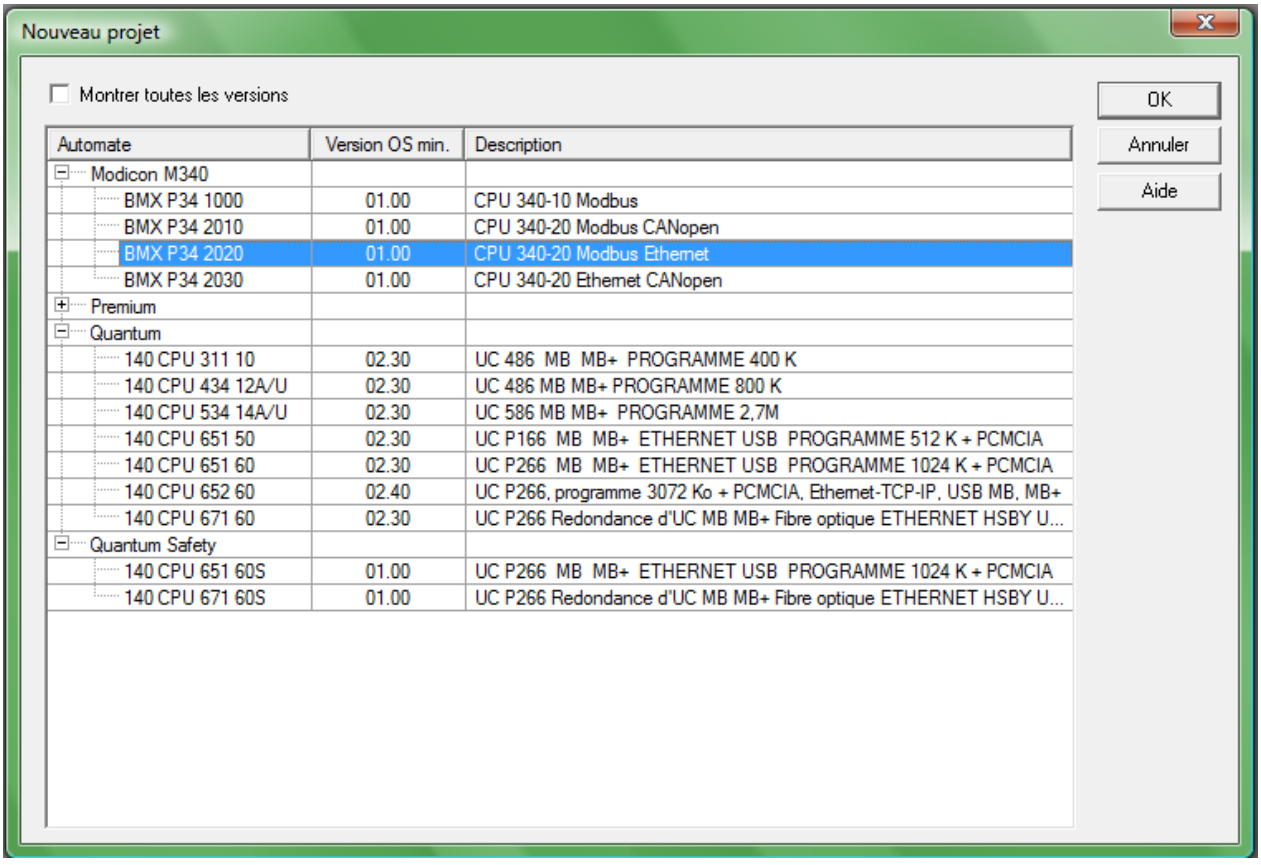


Figure II.3: Choix du processeur

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 II.1 : Exemple d'un automate premium avec les principales caractéristiques

4.2. Configuration d'un rack

Lors de la création d'un projet, un 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.

Ce rack contient le type de processeur sélectionné lors de la création du projet. Il est possible de remplacer ce processeur par un processeur compatible. Le nombre de racks gérés par le processeur diffère selon le processeur choisi, le tableau **[Annexe B]** montre le nombre de rack gérés pour différents processeurs.

On peut ajouter un rack en faisant glisser un des racks proposés dans le catalogue matériel (navigateur de l'éditeur de bus) vers les zones « 1 » ou « 2 », ou bien en double cliquant sur une de ces zones, comme montré sur la figure suivante:

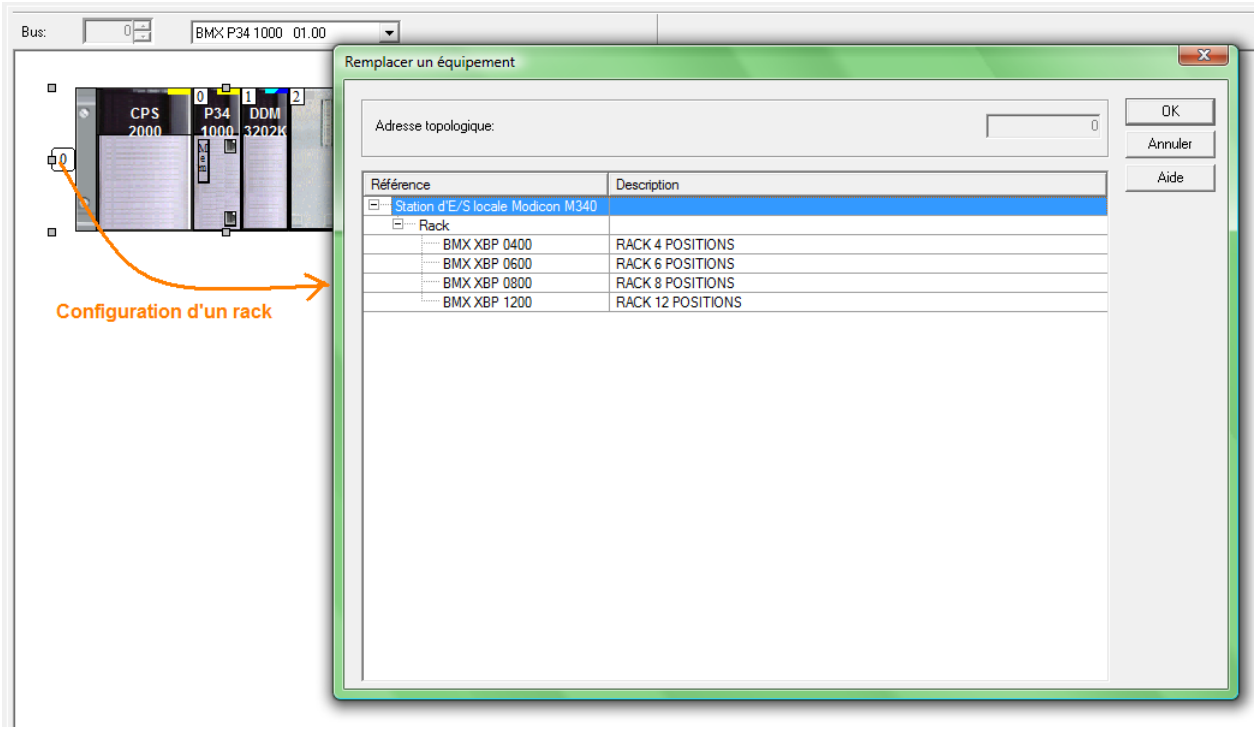


Figure II.4: Ajout d'un rack.

4.3. 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

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

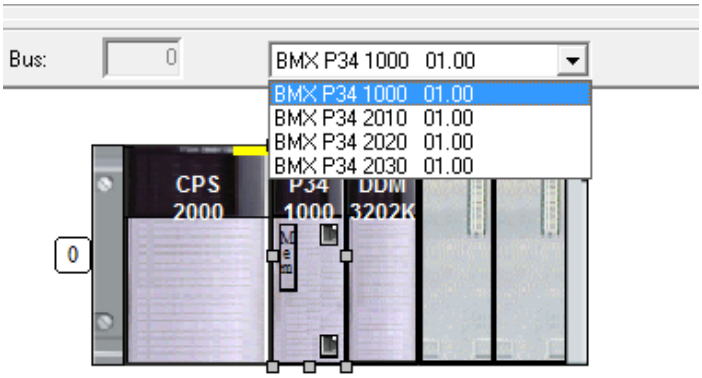


Figure II.5 : Liste des processeurs pour un remplacement.

4.5. Configuration des différents modules :

4.5.1. Configuration des modules d'E/S :

4.5.1.1. Positionnement des modules :

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

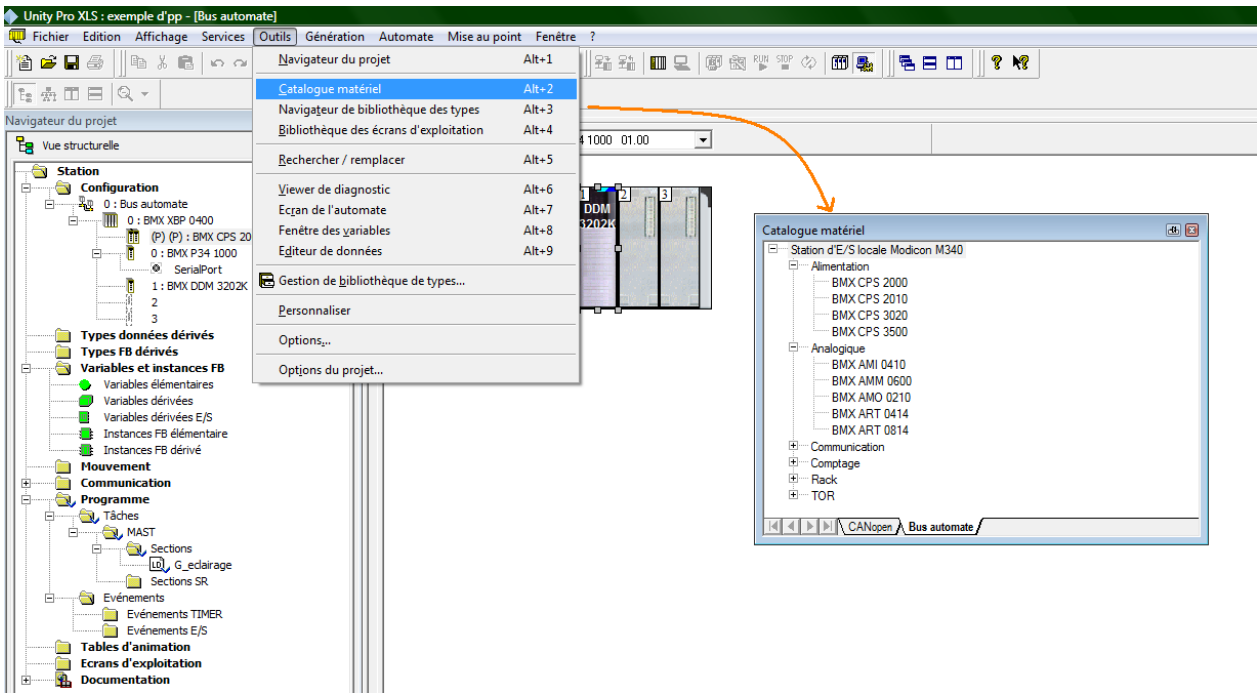


Figure II.6 : Les différents modules de Modicon M340

4.5.1.2. Bilan de consommation :

Un bilan de consommation est établi sur :

- Le module d'alimentation d'un rack
- Chaque module (processeur, modules d'E/S), dépendent du module d'alimentation du rack.

Ce bilan est présenté sous forme de barographe ou chaque couleur à une signification particulière, elle signale pour la tension correspondante:

- le débit de courant en cours: couleur verte,
- la quantité de courant encore disponible: couleur blanche,
- une surcharge de courant: couleur rouge, lors du dépassement un message est affiché.
- et la puissance totale (même code de couleur),

Le bilan de consommation *du module d'alimentation* montre la quantité de courant débitée par l'alimentation pour chaque tension qu'elle fournit, ainsi que la puissance totale.
Lorsqu'on ajoute ou retire un module, le bilan est ajusté à l'ouverture de la fenêtre Bilan de l'Alimentation et des E/S.

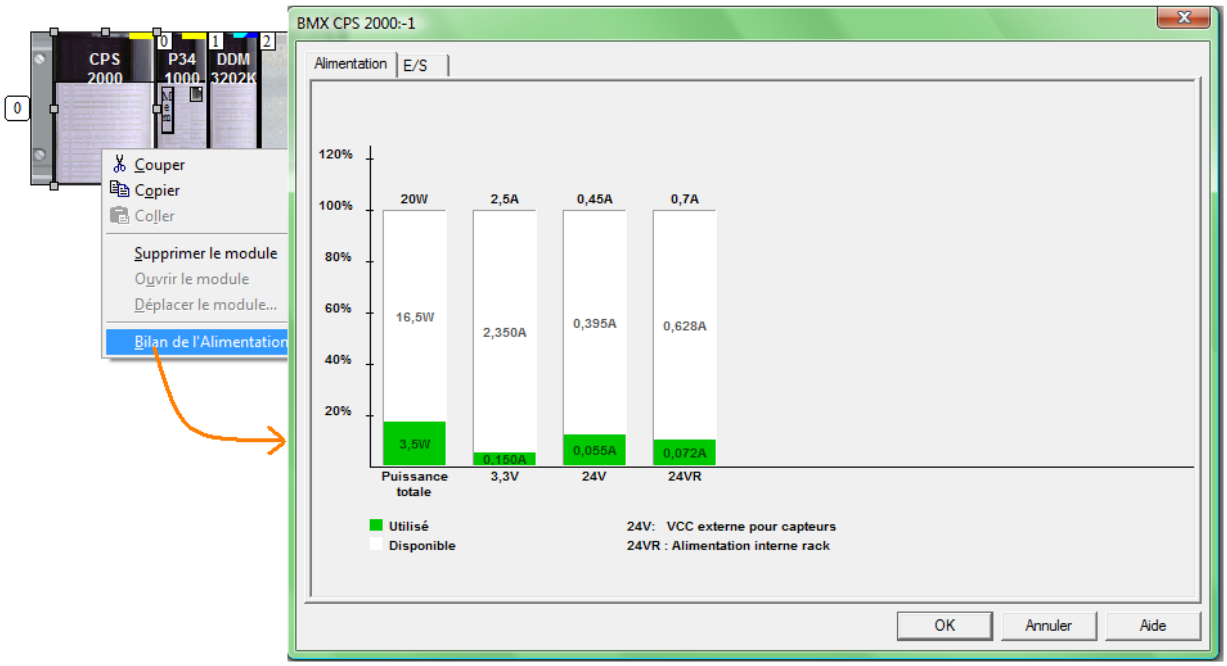


Figure II.7 : Bilan de consommation de l'alimentation.

On accède au bilan de consommation par le menu contextuel en sélectionnant le module d'alimentation puis en choisissant l'onglet *Bilan d'Alimentation.*

Remarque :
Le bilan de consommation pour les autres modules montre la quantité de courant débitée dans le module pour chaque tension qu'il utilise, ainsi que la puissance totale.

4.5.1.3. Les autres modules :

Le catalogue matériel offre la possibilité de choisir un module et de le faire glisser dans l'un des racks de l'éditeur de bus.

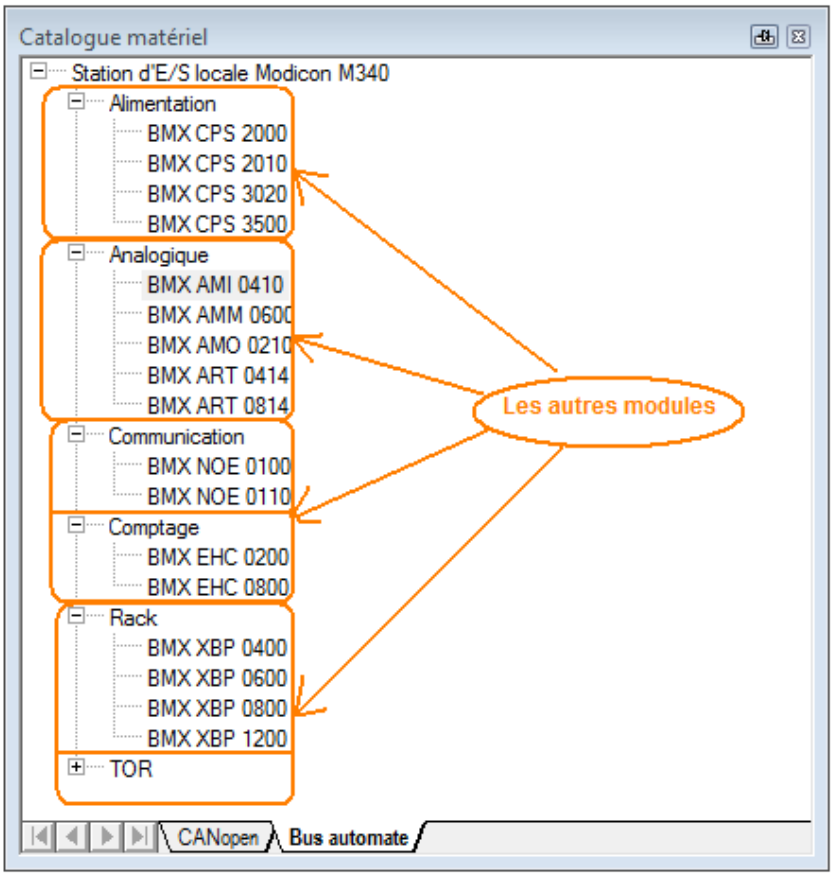


Figure II.8 : Modules proposés par le catalogue matériel

En double-cliquant sur le module choisi, on obtient une fenêtre qui nous donne la description du module ainsi que l'onglet « Objet d'E/S ». Cet onglet permet de gérer les objets d'entrée et de sortie d'un module, d'un équipement sur un bus de terrain ou encore les objets mémoire et système de l'automate.

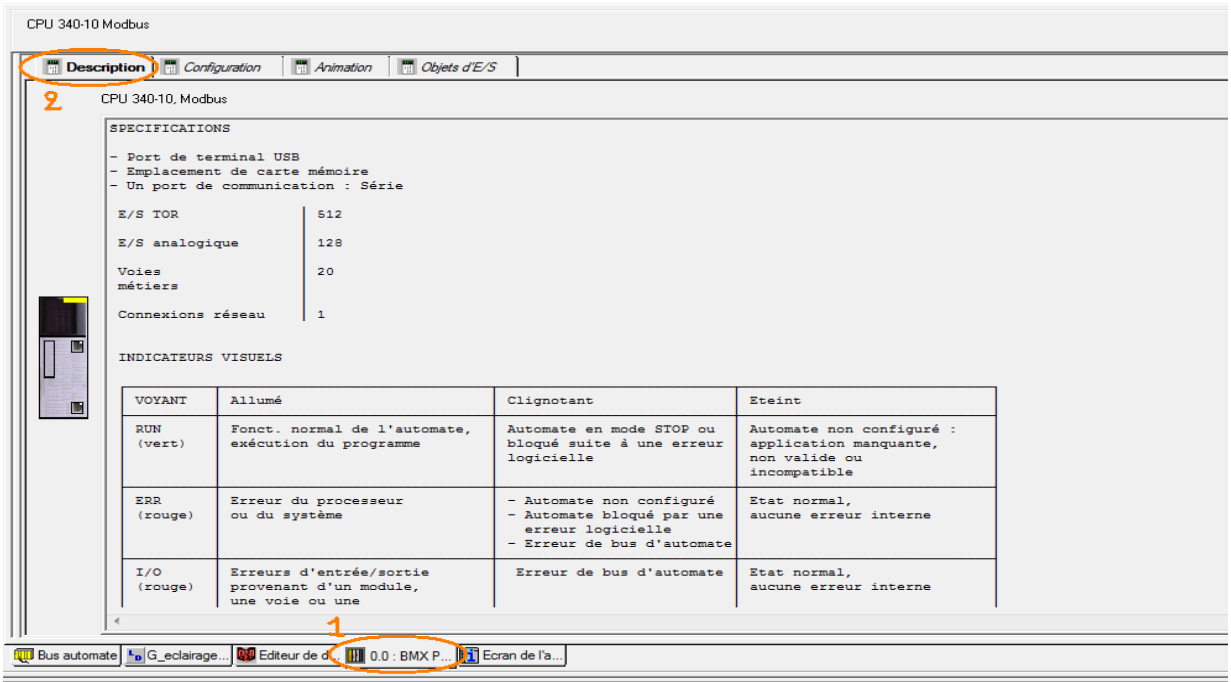


Figure II.9 : Description

4.5.1.4. 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 :

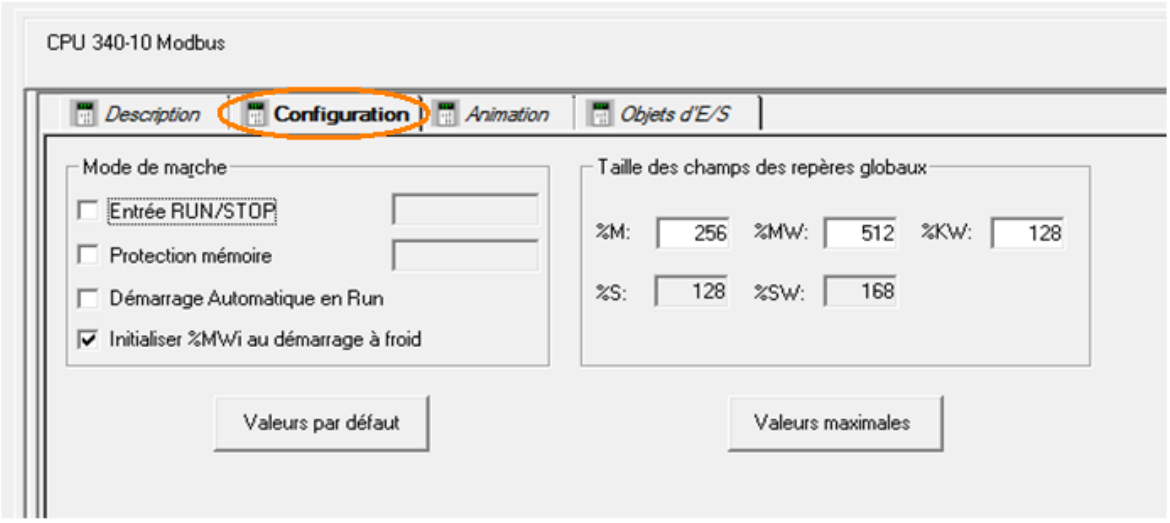


Figure II.10 : Configuration du processeur

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.

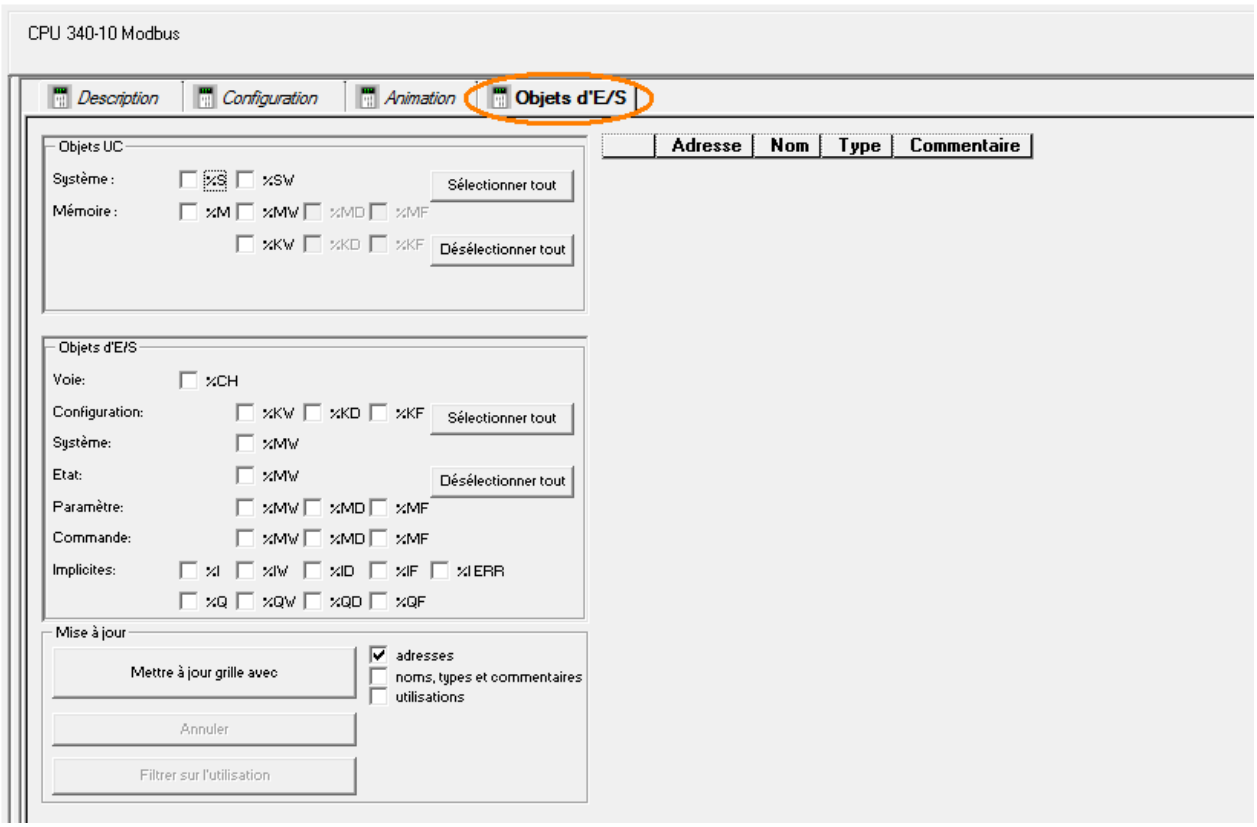


Figure II.11 : Objet d'E/S

5. Création et configuration des réseaux logiques de communication :

Ajouter un nouveau réseau (clic droit sur le dossier Réseaux du navigateur d'application)

- Choisir le type de réseau à créer (Ethernet, Modbus+..) et saisir son nom
- Saisir un commentaire (si nécessaire).

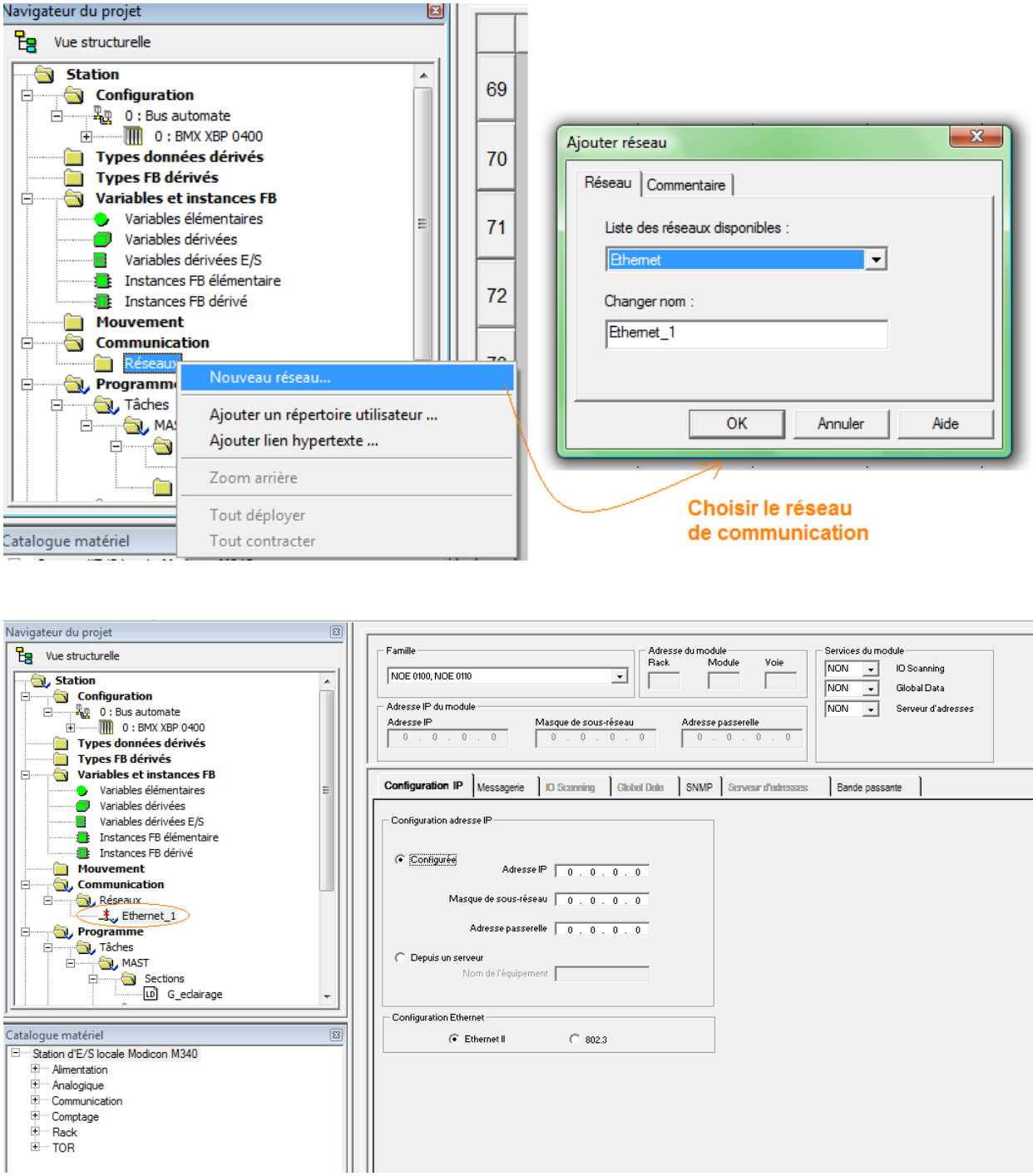


Figure II.12 : Création d'un réseau

- Activer le réseau logique à configurer
- Configurer le réseau logique : global data, I/O scanning, …

6. Définition du module de communication :

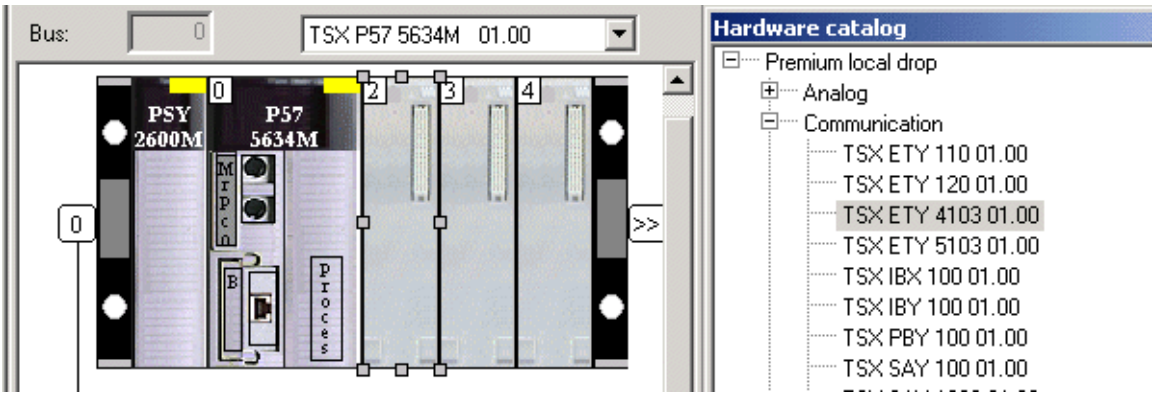


Figure II.13 : Choix du module de communication

- Définir le module de communication (glisser-déplacer depuis le catalogue matériel)
- Ou définir la carte PCMCIA (double clic sur la position de la carte et ajout du sous-module)

6.1. Associer le module ou la carte PCMCIA au réseau logique.

- Ouvrir le module de communication.
- Sélectionner la voie.
- Associer le module au réseau logique.

6.2. Configuration du module de communication :

Ces modules peuvent être simples ou doubles, en choisissant le module «Communication», on a la liste des modules compatibles avec le processeur utilisé. Un module de communication non compatible ne pouvant pas être sélectionné.
Après avoir ajouté le module, on double click sur lui et on aura la fenêtre « configuration » affichée.

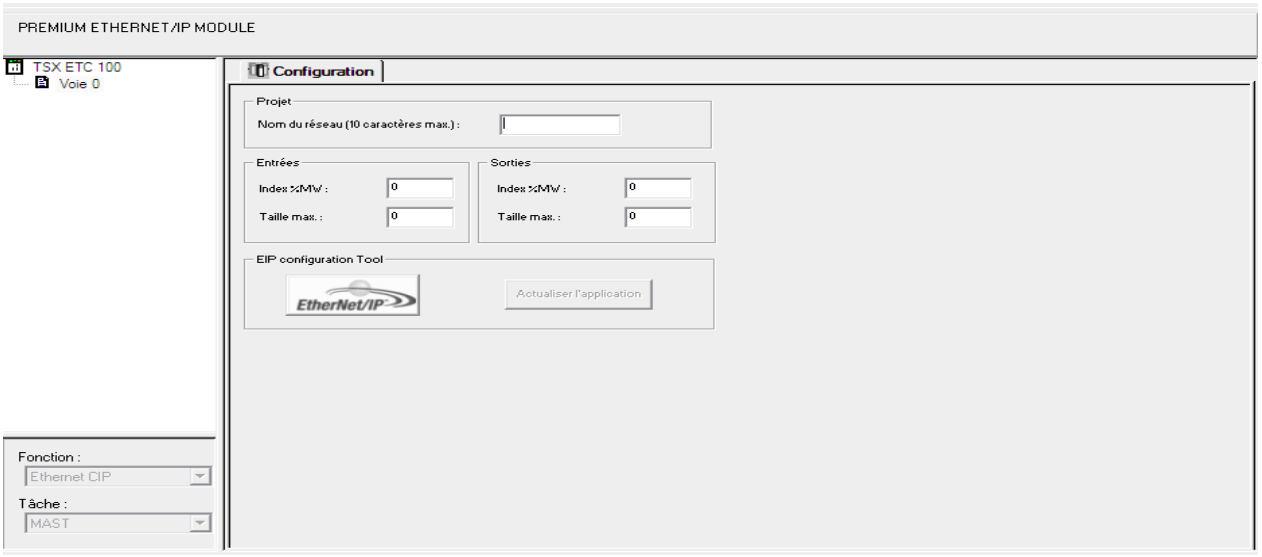


Figure II.14 : Configuration du module de communication

7. La Programmation :

Un programme s'exécute en trois étapes, à savoir, lecture de données puis tratement du programme et en fin écriture des sorties. Il peut être ***cyclique*** ou ***periodique***, dans le premier cas il entame le prochain cycle juste après la mise à jour des sortie, alors que pour le deuxième il ne l'entame qu'après un temps fixe (periode) même si la mise à jour des sorties a été faite.

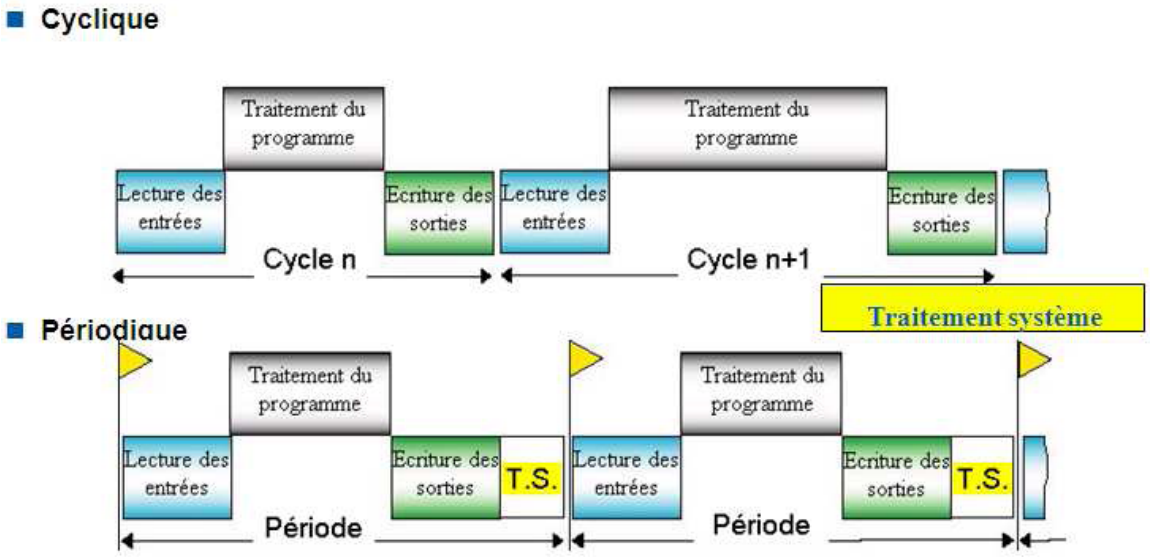


Figure II.15 : Etapes d'exécution d'un programme (cyclique et périodique)

Un programme peut être :

• Mono-tâche : Ne comprenant que la tâche principale (MAST)
• Multi-tâche : Comprenant les tâches MAST + tâches rapides (FAST) + tâches événementielles (EVT) + tâches auxiliaires (AUX).

Ces tâches diffèrent de part leur priorité d'exécution, la figure suivante montre l'ordre de priorité.

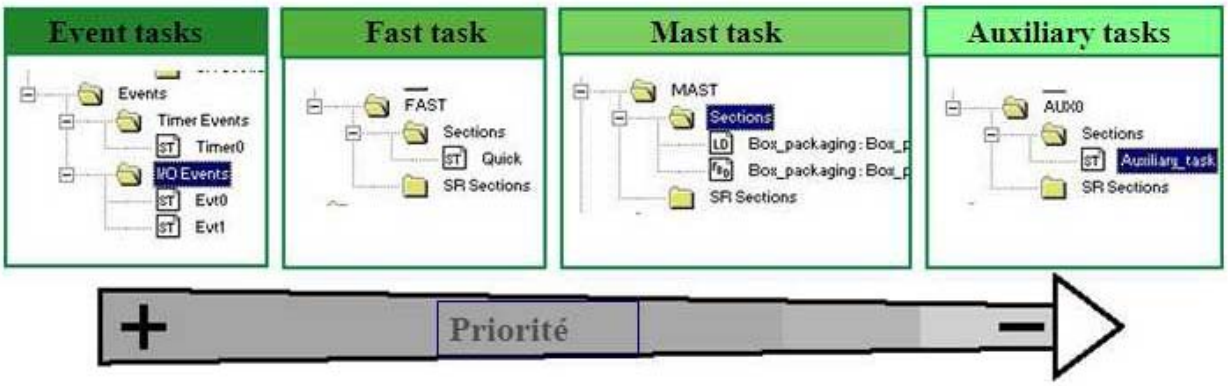


Figure II.16 : Ordre de priorité des différentes tâches

7.1. Les différentes tâches :

7.1.2. La tâche MAST :

La tâche « maître » représente le programme principal. Elle est obligatoire quelle que soit la structure adoptée mono-tâche ou multitâche donc elle est crée par défaut.
Le programme de la tâche maître est constitué de plusieurs modules de programmes appelés sections et sous-programmes, qui sont liées à une tâche. Une même section ou sous-programme ne peut pas appartenir simultanément à plusieurs tâches.

Les appels aux sous-programmes s'effectuent dans les sections ou depuis un autre sous programme (8 niveaux d'imbrications maximum).

L'exécution de la tâche maître peut être choisie (en configuration) : cyclique ou périodique.

7.1.3. Tâche FAST :

C'est une tâche plus prioritaire que la tâche maître MAST et périodique avec une période configurable de 1 à 255 ms. Comme pour la tâche MAST, le programme associé se compose de sections et de sous-programmes.

Le contrôle de la tâche ainsi que la visualisation des temps d'exécution se fait avec des bits systèmes associés.

Exemple : le bit système %S11 est positionné à 1 et l'application est déclarée en défaut bloquant pour l'automate en cas de débordement (chien de garde).

7.1.4. Création d'une nouvelle tâche :

Dans le cas d'un programme multitâches on peut ajouter une tâche depuis le navigateur de projet, on a le choix entre une tâche FAST et AUX (disponible que dans quelques processeurs).

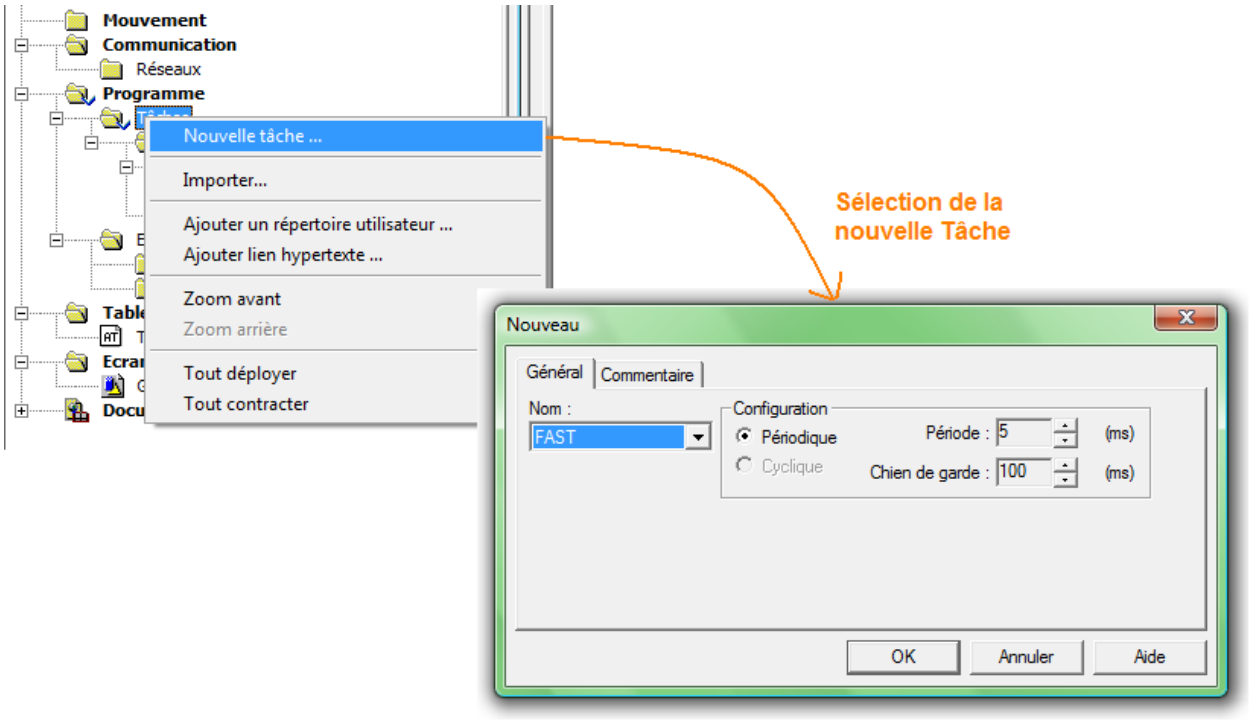


Figure II.17 : Création d'une nouvelle tâche

7.1.5. Tâches auxiliaires AUX :

Une tâche auxiliaire est utilisée pour les tâches de traitement plus lentes. On peut programmer jusqu'à 4 tâches auxiliaires maximum (AUX0 à AUX3) sur Premium TSX P57

Elle est composée de sections et des sous-programmes programmés en LD, FBD, IL, ST**.** L'exécution est périodique (de 10 ms à 2,55 s)

7.1.6. Tâche événementielle « EVT E/S» et « TIMER »:

Elle est utilisée pour réduire le temps de réponse du programme aux événements des modules d'entrée/sortie et aux temporisateurs d'événement, elle contient une seule section programmée en LD, FBD, IL, ST.

EVTi : événements provenant des modules d'entrée/sortie
TIMERi : événements provenant des temporisateurs d'événements (fonction ITCNTRL).

7.2. Choix du langage de programmation :

Dans le navigateur de projet on fait un clic droit sur MAST et on choisit « nouvelle section », on a la possibilité de choisir l'un des cinq langages de programmation

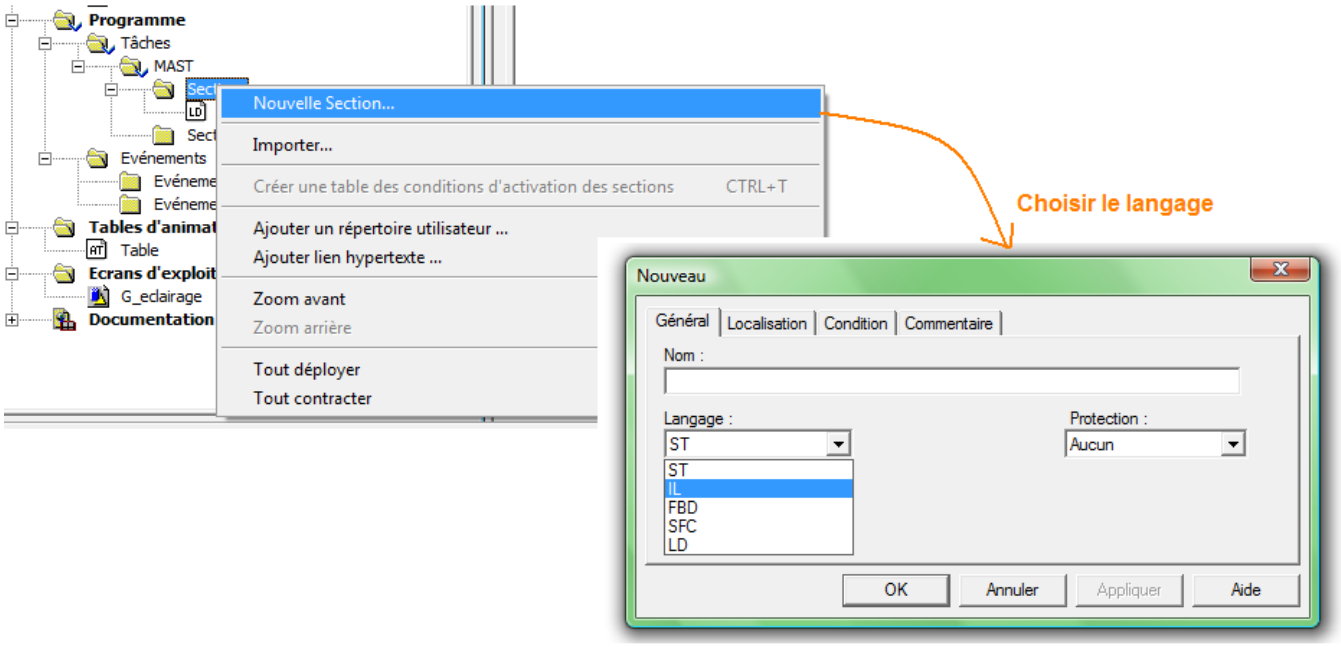


Figure II.18 : choix du langage

7.2.1. Instruction List « IL » :

Instruction List (*IL*) est un langage booléen proche du langage machine, il est compatible avec la norme IEC 61131-3. Il est difficilement lisible pour un utilisateur.

Figure II.21 : Programme en IL

Chaque instruction commence dans une nouvelle ligne et est composée de :
Commentaire (optionnel) : Informations supplémentaires sur la ligne
Opérateur : définit la nature de l'instruction. Les différents opérateurs sont :

- Assignment (*LD, ST, S, R*)
- Logique (*AND, OR, XOR, NOT*)
- Arithmétique (*ADD, SUB, MUL, DIV, MOD*)
- Comparaison (*GT, GE, EQ, NE, LE, LT*)
- Structure (*JMP, RET, «) »*) 54
- Appel de fonction (*CAL function_name*)

Modificateur : influence l'exécution de l'opérateur, les différents modificateurs sont :

- N : invertit la valeur de l'opérande bit par bit.
- C : l'instruction associée n'est exécuté que si le résultat est vrai.
- CN : l'instruction associée n'est exécuté que si le résultat est faux.
- ( ) : sont utilisées pour pouvoir combiner les différentes instructions.

Opérande : l'objet sur lequel agit l'opérateur, il peut être une adresse directe, une valeur, une variable, un DDT, un élément d'une DDT, une sortie d'une EFB ou bien un appel d'une EFB/DFB.

Etiquette (optionnel) : localise une séquence dans le programme.

7.2.2. LADDER DIAGRAM:

Le Ladder est équivalent à un schéma à relais.Il est composé de contacts, fonctions et bobines reliés à 2 barres verticales. Chaque réseau est composé d'objets graphiques correspondant aux contacts, liaisons, bobines, blocs opérations, blocs fonctions EFs/EFBS/DFBs, saut, appel de sous programmes.

Coté gauche on trouve une barre de potentiel qui correspond à la phase (conducteur L), seuls les objets LD (contacts, bobines) connectés à cette source d'alimentation sont exécutés par le programme LD. La barre de potentiel droite correspond au conducteur neutre n'est pas visible. Cependant toutes les sorties de bobines et FFB sont logiquement connectées.

Le langage de programmation LD est orienté cellules, c'est-à-dire un seul objet est placé dans chaque cellule**.** Une grille divise la fenêtre en lignes et colonnes.

L'ordre de traitement des objets individuels d'une section LD est déterminé par le flux de données d'une section. Les réseaux associés à la barre d'alimentation gauche sont traités du haut vers le bas (connexion à la barre d'alimentation gauche). Les réseaux de la section sont indépendants des autres et sont traités dans l'ordre de placement.

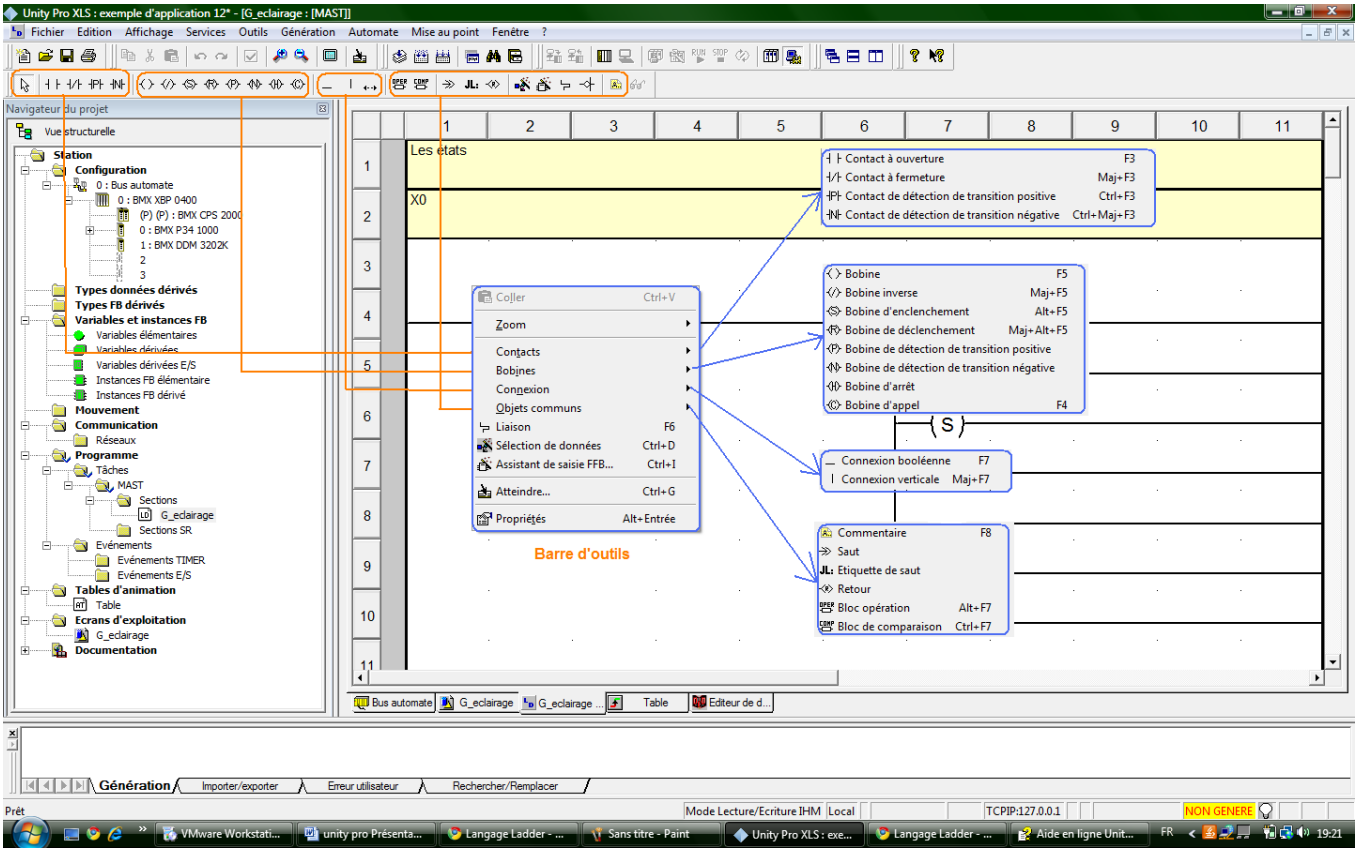


Figure II.19 : Editeur de programme LADDER

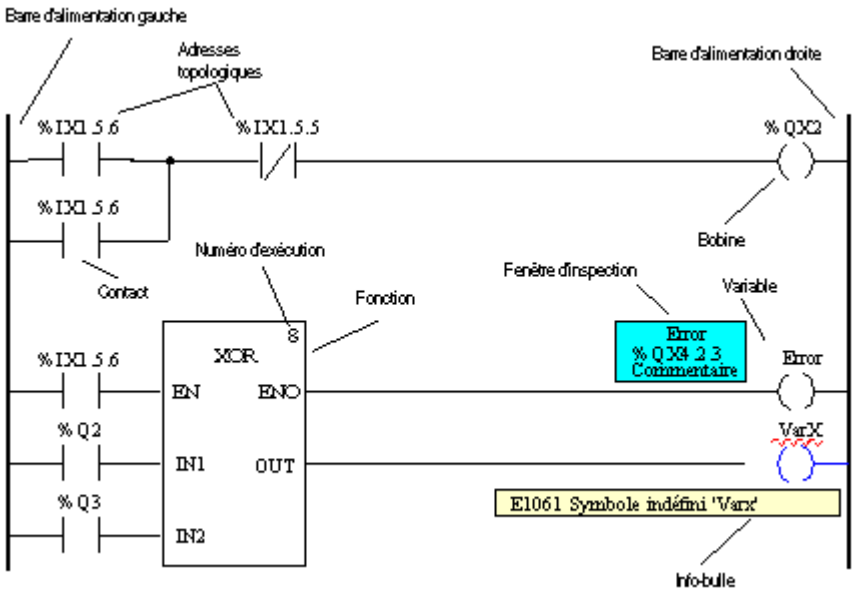


Figure II.20: représentation d'une section LD

7.2.3. Functional Block Diagram « FBD »:

Le « Function Block Diagram » est un éditeur graphique orienté « flux de données », il est conforme au standard IEC 61131-3.Ce langage est particulièrement adapté aux applications de commande de processus, il est constitué de blocs élémentaires, fonctions dérivées, structures.…

Une section programmée en FBD comporte l'équivalent d'une grille présentant 30 colonnes de 23lignes.

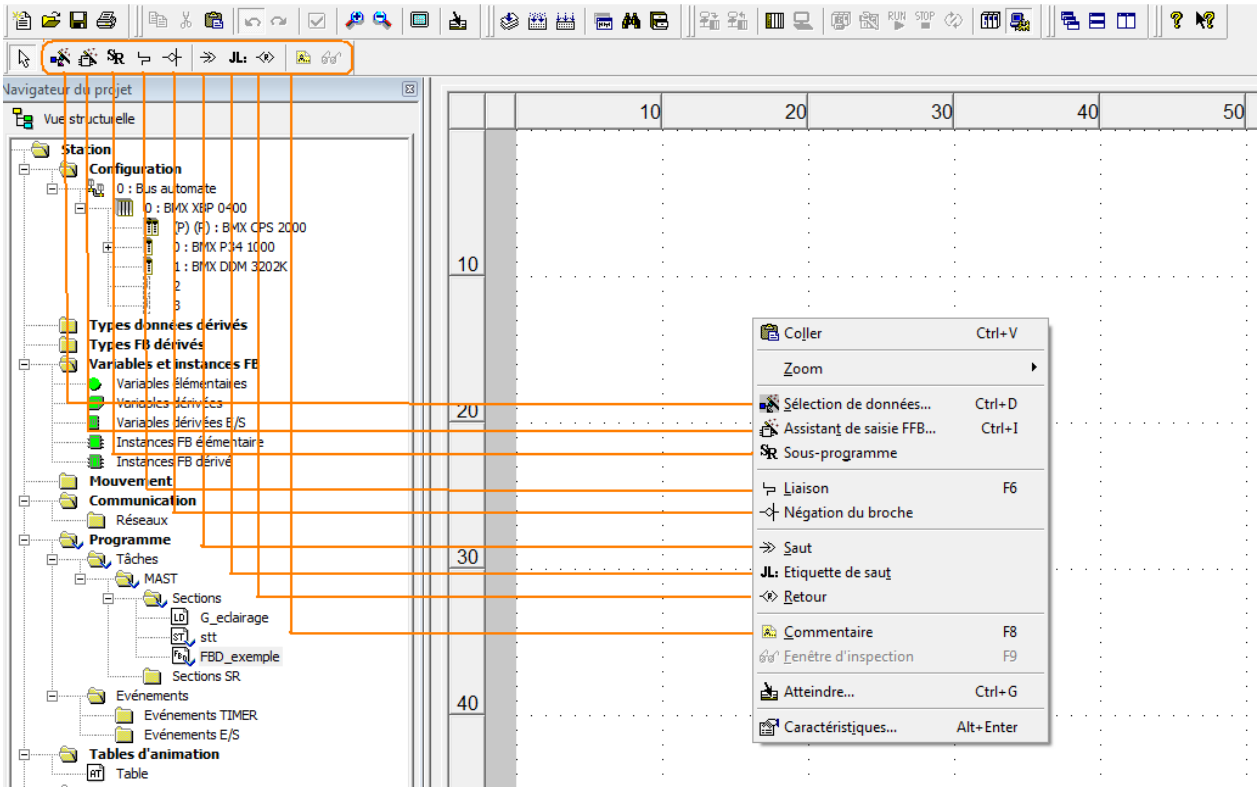


Figure II.21 : Editeur de Programme FBD

Comme on peut le voir sur la figure, l'éditeur DFB propose de nombreuses fonctionnalités accessibles soit dans la barre d'outils, ou bien avec un clic droit sur la fenêtre principale. On a :

1) Sélection de données : Pour choisir une variable déjà crée dans la bibliothèque ou bien définir une nouvelle variable.
2) Assistant saisie FFB : Pour entrer un nouveau bloc.
3) Sous programme : Pour créer un sous programme.
4) Liaison : Pour relier les différents blocs
5) Négation de branche
6) Saut :
7) Etiquette de saut
8) Retour :
9) Commentaire.

Les objets du langage de programmation FBD offrent des aides permettant de structurer une section en un ensemble de :

- Blocs élémentaires EFs, Les blocs fonctions élémentaires EFBs,
- Blocs fonctions dérivés DFBs
- Procédures
- Eléments de contrôle

Ces objets regroupés sous l'abréviation générique FFB peuvent être liés entre eux par :

- Des liaisons
- Des paramètres réels
- Des commentaires peuvent être ajoutés pour insérer une logique étendue sous forme de macros.

Les Différentes fonctions FFB peuvent être importées grâce au gestionnaire de bibliothèque.

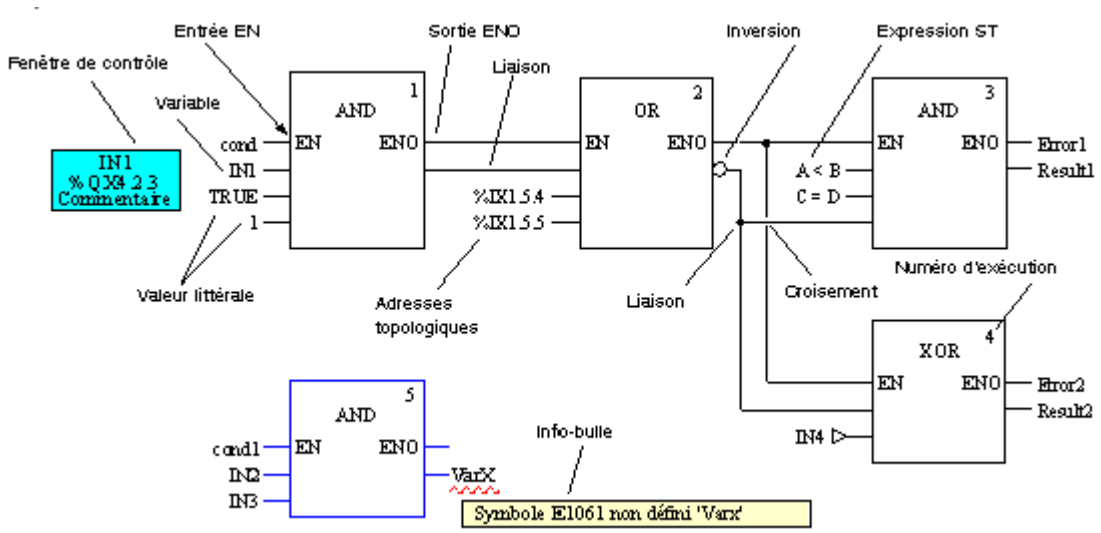


Figure II.22 : Exemple d'un schéma FBD

Comme le montre la figure précédente, les entrées des différents blocs peuvent être des constantes booléennes et analogiques, des variables, et des expressions logiques programmées en « ST ».

7.2.4. LITTERAL STRUCTURE (ST) :

Le langage littéral structuré (ST) est un langage évolué de type algorithmique particulièrement adapté à la programmation des fonctions arithmétiques complexes, manipulations de données. Il permet la réalisation de programmes par écriture de lignes de programmation, constituées de caractères alphanumériques.

Une section de programme littéral est organisée en phrases. Une phrase littérale est l'équivalent d'un réseau de contacts en langage contacts.

On a accès directement aux boucles conditionnelles à travers la barre d'outils, on clique sur l'une d'elles et elle est automatiquement ajoutée dans notre programme. Il nous suffit de remplir les vides avec des instructions se terminant par un « ; ».

On peut aussi importer des fonctions (EF) et des blocs de fonctions (EFB, DFB) grâce à l'assistant FFB.

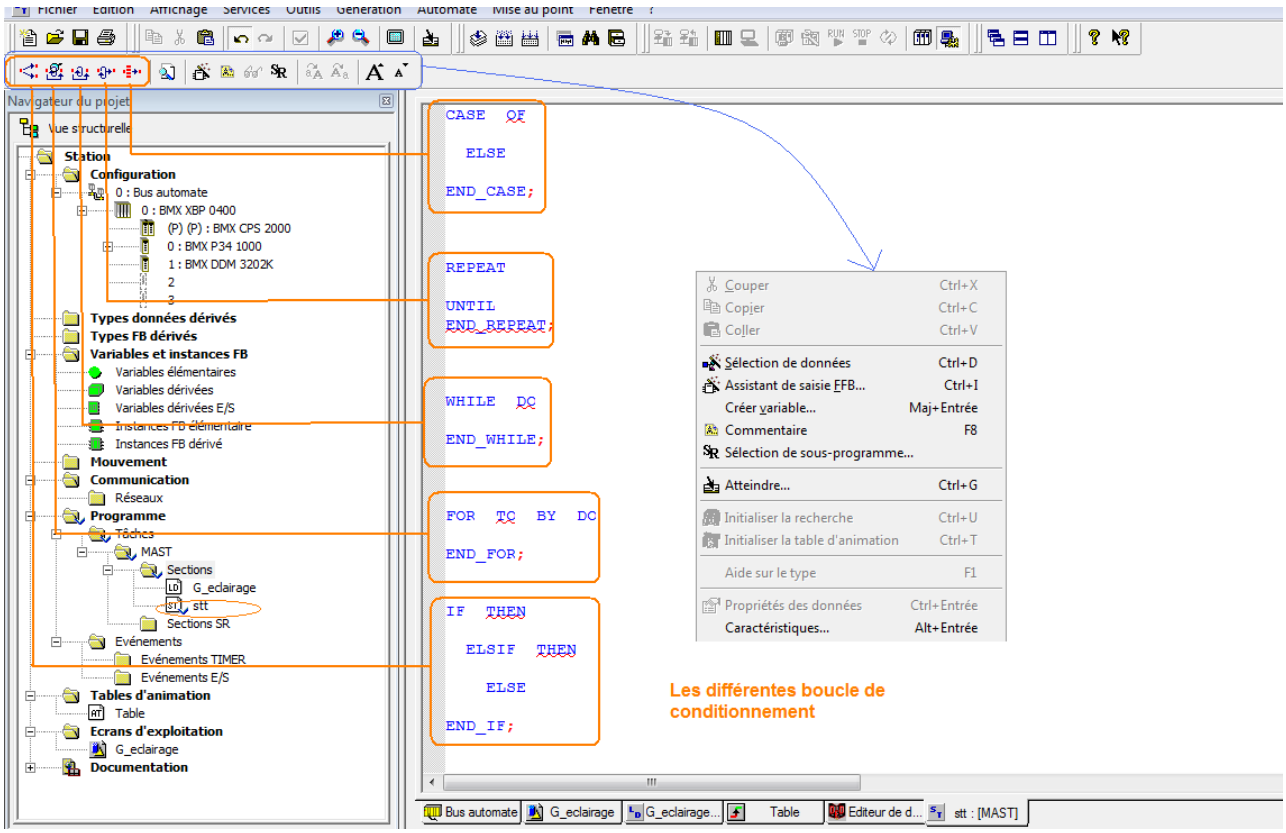


Figure II.23: Editeur de programme ST

Les instructions doivent être terminées par des points-virgules. Une ligne peut contenir plusieurs instructions (séparées par des points-virgules).Un seul point-virgule représente une instruction vide. Une ligne est limitée à 300 caractères. Il est possible d'utiliser des sauts de ligne dans les instructions (instructions d'affectation à plusieurs lignes).Des étiquettes, symboles et commentaires peuvent être librement placés dans la section. (Les commentaires peuvent être saisis à tout endroit où les espaces sont autorisés).

Une vérification de la syntaxe et de la sémantique a lieu directement après la saisie des instructions d'affectation. Le résultat de la vérification est indiqué par différentes couleurs de texte.

Les sections comportant des erreurs de syntaxe ou de sémantique peuvent également être enregistrées.

7.2.5. Sequential Functional Chart (SFC):

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'actions
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.

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

La logique de la section peut être commentée par des objets texte. Une section SFC est une "machine d'états", c.-à-d. que l'état est déterminé par les étapes actives et les transitions renvoient le comportement de commutation/modification entre les états. Les étapes et transitions sont reliées les unes aux autres par des liaisons dirigées. Deux étapes consécutives ne peuvent jamais être directement reliées ; elles sont toujours séparées par une transition. Les évolutions des états actifs de signaux se déroulent le long des liaisons dirigées, et sont déclenchées par la commutation d'une transition. Le déroulement d'une séquence va dans le sens des liaisons dirigées et se déroule de la partie inférieure de l'étape précédente à la partie supérieure de l'étape suivante. Les divergences sont traitées de gauche à droite.

Chaque étape peut compter zéro ou plusieurs actions. A chaque transition est associée une condition de transition.

La dernière transition de la séquence est toujours reliée à une autre étape de la séquence (par une liaison graphique ou un symbole de "saut") de manière à obtenir une boucle fermée.
Les séquences d'étapes se déroulent donc de façon cyclique.

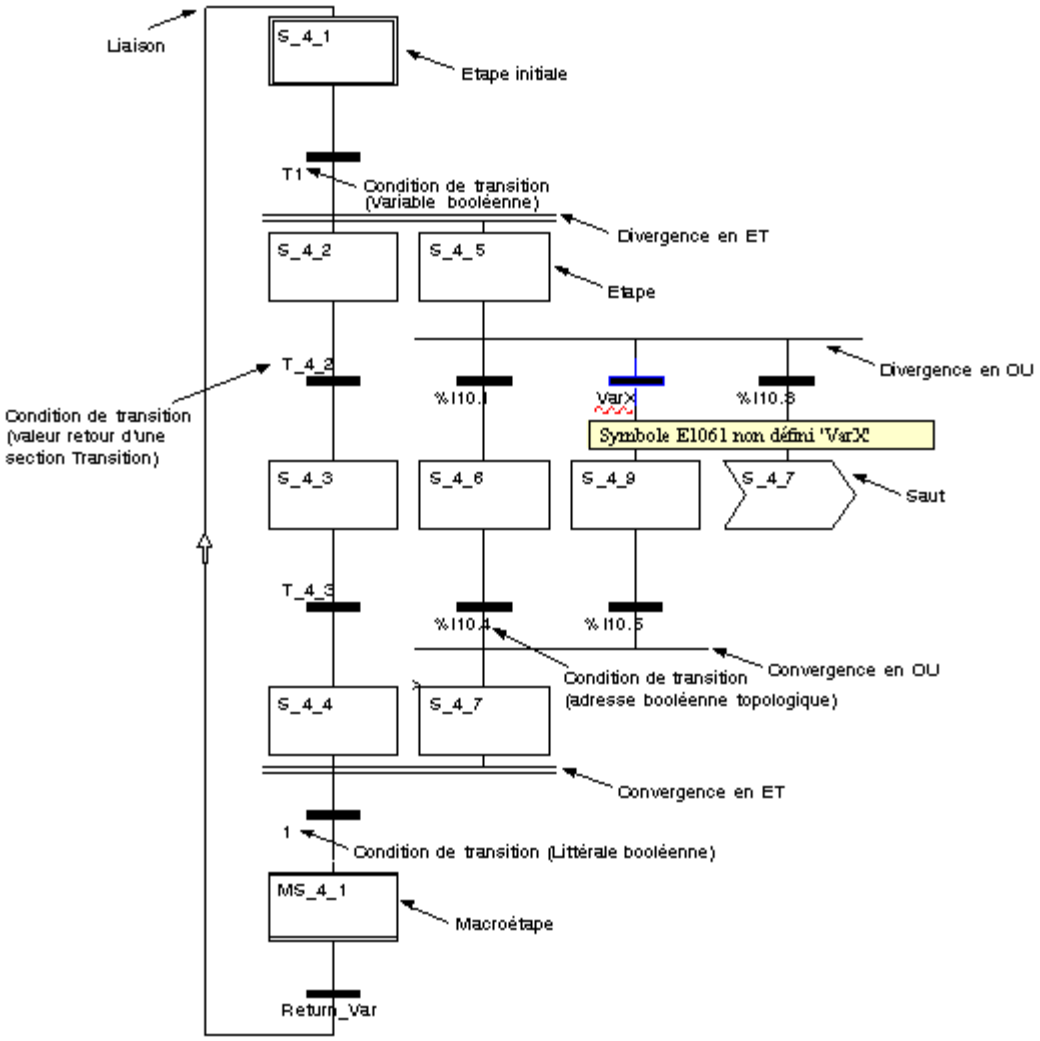


Figure II.24: Représentation d'une section SFC

Le comportement d'un réseau SFC dépend largement du nombre de jetons choisis, c.-à-d. du nombre d'étapes actives.

Un comportement univoque est possible en utilisant un seul jeton (single token). Les divergences en ET comportant un jeton actif (étape) par branche sont considérées comme des jetons uniques. Ceci correspond à une séquence d'étapes selon la norme CEI 61131-3.

Une séquence d'étapes comportant un maximum d'étapes actives (jetons multiples) définies par l'utilisateur augmente le niveau de liberté. Les limitations relatives à l'obligation d'unicité et du non-blocage sont à cet effet levées et doivent être assurées par l'utilisateur. Les séquences d'étape à jetons multiples ne sont pas conformes à la norme CEI 61131-3.

7.3. Variables :

7.3.1. Définitions :

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 Acces_interdit est associée au mot mémoire %MW102. Acces_interdit 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.

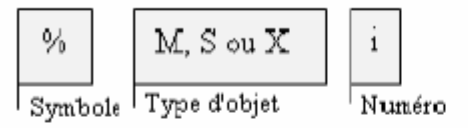
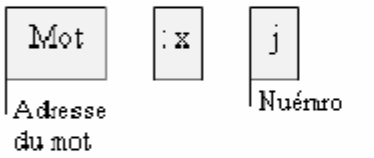
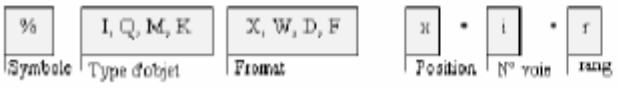
Une **variable privée** est une variable utilisée par certains blocs fonctions. Ces variables ne sont pas accessibles par le programme d'application

I/ODDT est l'abréviation de « 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.

7.3.2. Adressage :

L’adressage diffère selon que notre variable est un bit ou un mot, se trouve dans un module ou bien est interne.

Règles d'adressage	**Syntaxe**
Objets bits : 	**Type d'objet :** M : bits internes S : bits système X : étapes (Grafcet) **Numéro :** dépend de la configuration
Bits extrait de mots : 	Numéro : 0 à 15 rang du bit dans le mot.
Objets de modules d'entrées/sorties du TSX37 : 	**Type d'objet :** I : image de l'entrée Q : image de la sortie M :variable interne K : constante interne **Format :** X : booléen W : simple longueur 16bits D : double longueur 32bits F : flottant 32bits

	Position module : 0 à 10 (TSX 37-22) **N° voie :** 0 à 31 ou MOD **Rang :** 0 à 127 ou rang
Objets de modules d'entrées/sorties du rack : 	**Type d'objet :** I : image de l'entrée Q : image de la sortie M : variable interne K : constante interne **Format :** X : booléen W : simple longueur 16bits D : double longueur 32bits F : flottant 32bits **Rack :** 0 à 7 **Position module :** 0 à 14 (position dans le rack) **N° voie :** 0 à 127 ou MOD **Rang :** 0 à 127 ou rang

Tableau II.2 : Adressage.

7.3.3. Editeur de données :

On y accède depuis le navigateur de projet en cliquant sur « Variables et instances FB »

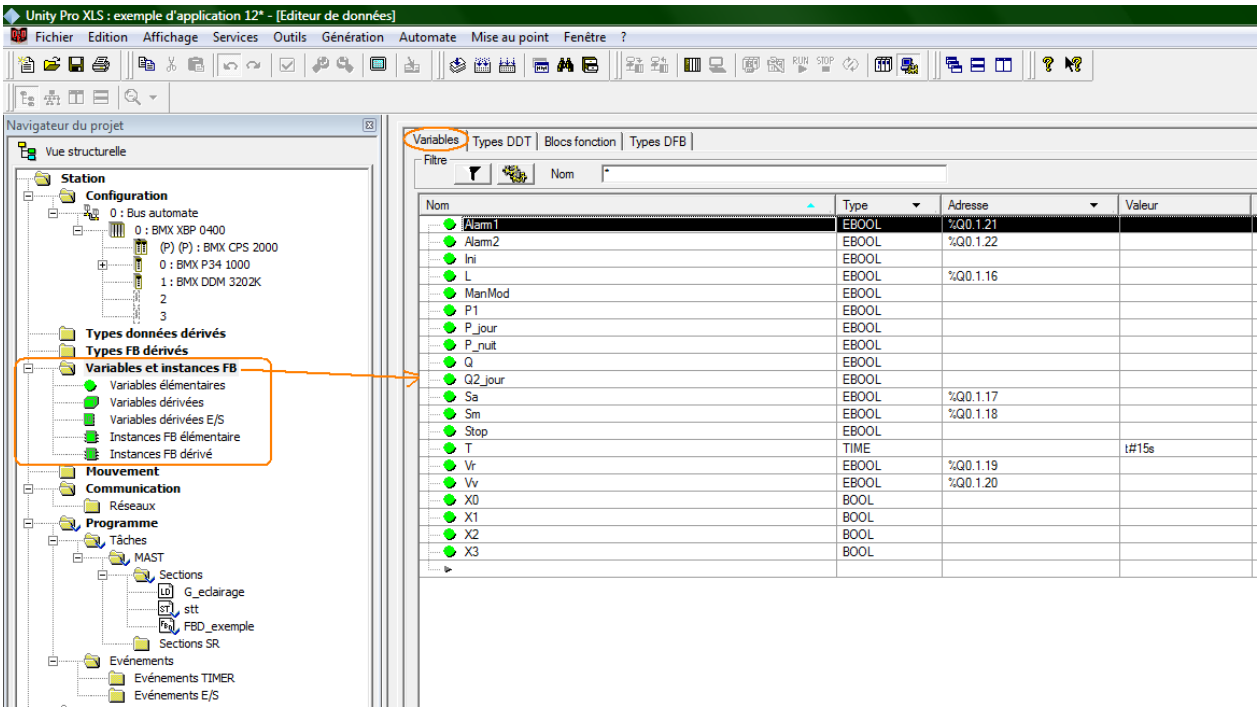


Figure II.25 : Editeur de données.

7.4. Les onglet :

7.4.1. Onglet « Variables » :

L'éditeur comprend quatre onglets, l'onglet « variables » permet de définir les différentes variables utilisés dans le programme, leurs adresses si elles sont affectées, leurs types, des noms et des commentaires.

Les variables peuvent être de différents types :

- **BOOL** : doit être activé (ON) (1) ou désactivé (OFF) (0)

- **WORD** : représente une "chaîne de 16 bits", ce qui signifie que la longueur des données est de 16 bits

- **INT** : représente une valeur d'entier. La plage des valeurs s'étend de -32 768 à 32 767

- **UINT** : représente une valeur d'entier non signé. La plage des valeurs s'étend de 0 à 65535

- **REAL** : représente une valeur à virgule flottante. La plage des valeurs s'étend de 8,43e-37 à 3,36e+38

7.4.2. Onglet « types DDT » :

Un type de données dérivé (DDT) correspond : soit à un tableau, soit à une structure concernant les données d'entrées/sorties. Dans ce cas, le type n'est pas créé par l'utilisateur mais fourni par le constructeur (IODDT),

7.4.3. Onglet « Bloc fonctions » :

Permet de créer des instances de types « Blocs de Fonction Elémentaires (EFB) », les EFB sont des blocs prédéfinis qu'on peut trouver dans la bibliothèque (par exemple : les compteurs, les temporisations…). Cet onglet permet de choisir un de ces blocs pour le programme et définir ses variables d'entrée/sortie.

7.4.5. Onglet « types DFB » :

Cet onglet permet de créer des instances de bloc de fonctions dérivées, ce sont des blocs crées par l'utilisateur. On définit les entrées/sorties qui peuvent êtres locales ou globales et on définit son rôle en la programmant on double cliquant sur « section », on peut utiliser les 4 langages de programmations (IL, LD, FBD et ST).

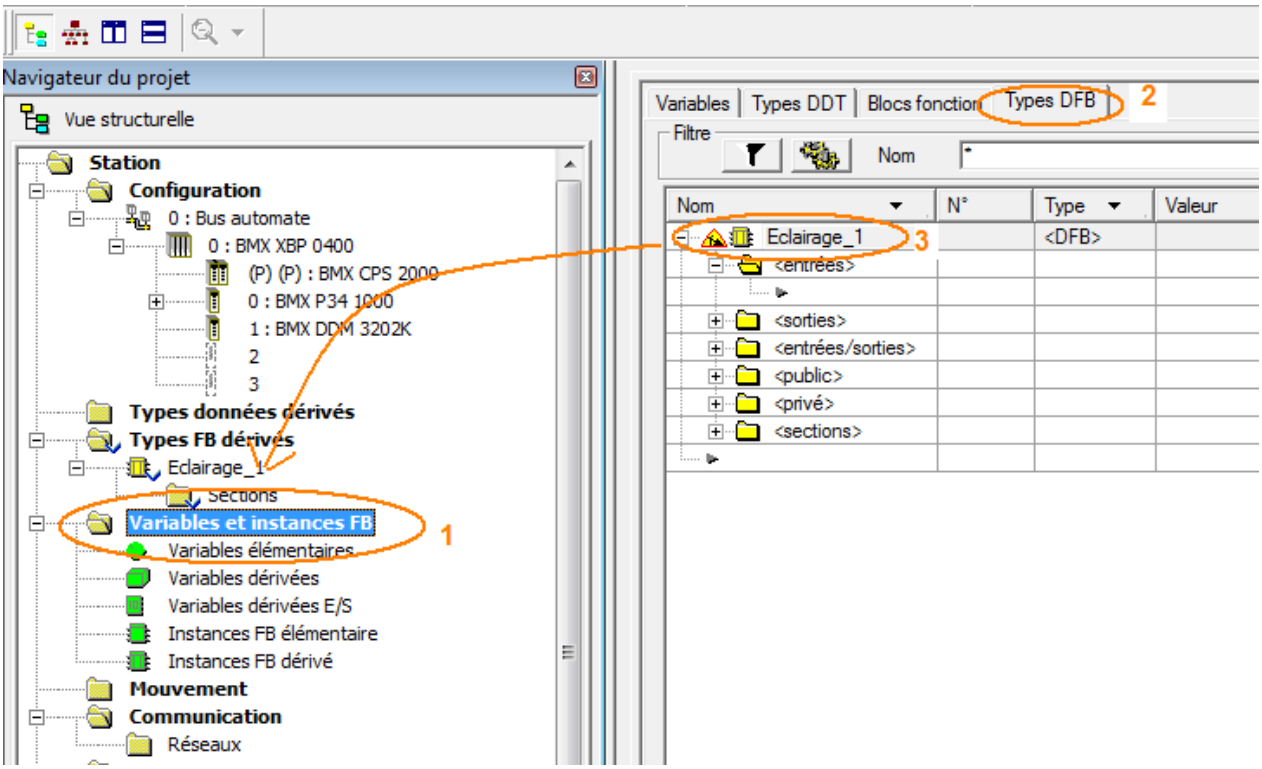


Figure II.26: Création d'un bloc DFB.

La DFB crée ne peut être utilisée que dans le projet courant. Pour qu'elle puisse être utilisé dans n'importe quel projet sur UNITY PRO il suffit de la placer dans la bibliothèque comme le montre la figure suivante :

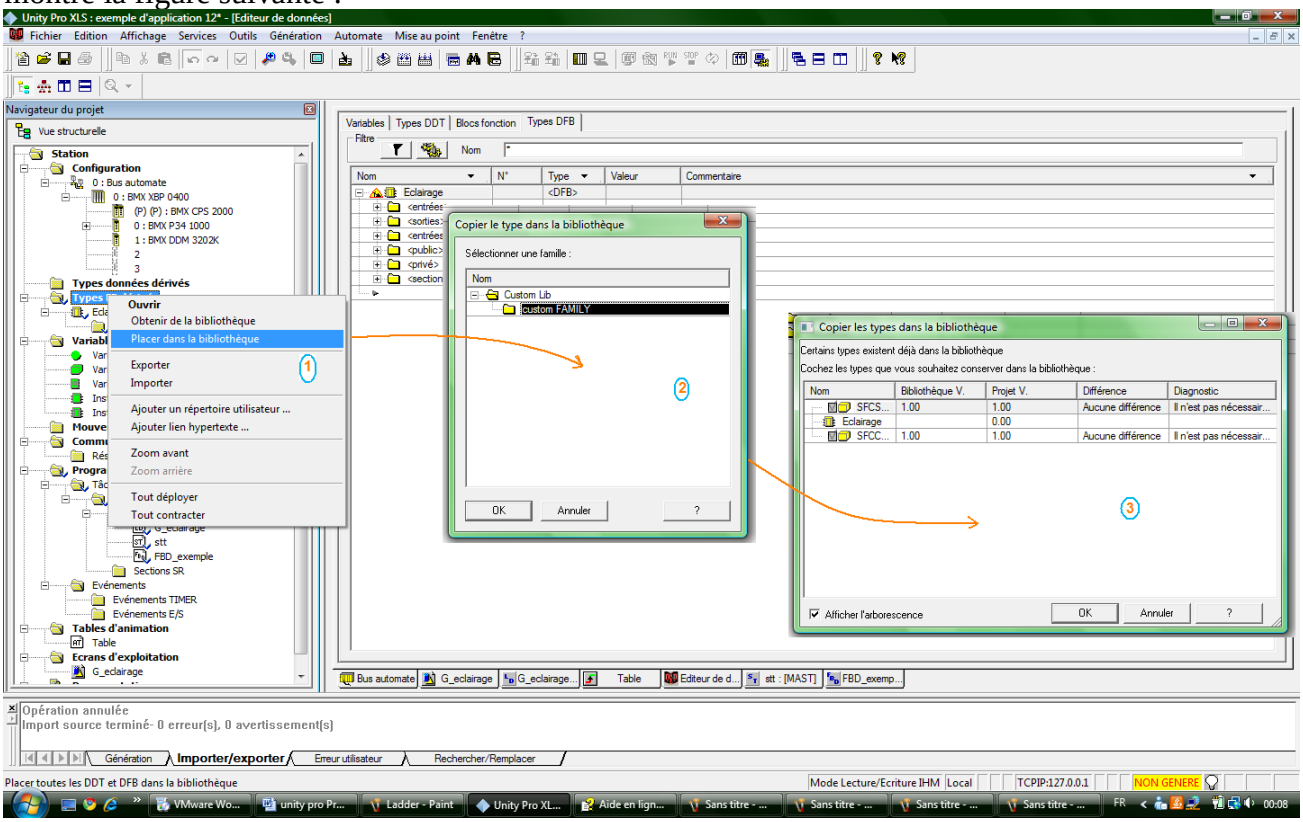


Figure II.27: Ajout de la DFB dans une bibliothèque.

8. Gestionnaire de bibliothèque :

Il contient tous les objets disponibles pour développer un projet d'automatisation:

- EFs (Fonctions Elémentaires) :
- EFBs (Blocs fonctions Elémentaires)
- DFBs (Blocs fonctions dérivés)
- DDTs (Types de Données dérivés)

Il fournit un ensemble de fonctionnalités permettant de modifier le contenu de la bibliothèque. Il exécute les transferts entre la bibliothèque et le projet.

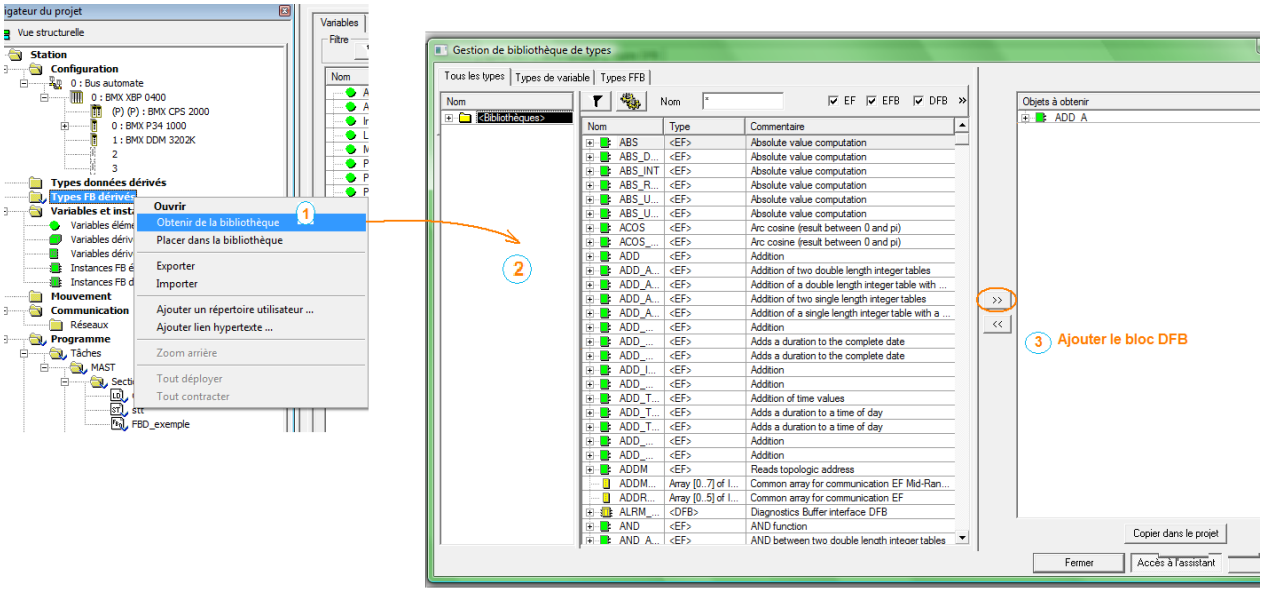


Figure II.28: gestionnaire des bibliothèques.

On a plusieurs blocs prédéfinis qui forme une bibliothèque de base**, avec** les groupes Tableaux, CLC_INT, Comparaison, Date et heure, Logique, Mathématiques, Statistiques, Chaînes, Temporisateurs et compteurs, Type à Type.

- **Communication** : avec le groupe Etendu contenant des FFB de communication

- **CONT_CTL** : avec les groupes Conditionnement, Automate, Mathématiques, Mesures, Traitement de sortie, Gestion des consignes

- **Diagnostics** : avec le groupe Diagnostics contenant 14 FFB de diagnostic

- **Gestion E/S** : avec les groupes Configuration E/S analogique, Mise à l'échelle E/S analogique, Echange explicite, E/S immédiate, Configuration E/S Quantum

- **Mouvement** : avec les groupes Commande d'axe, CAM, MMF Start

- **Bib Obsolète** : avec les groupes CLC, CLC_PRO, Extensions/Compatibilité

- **Système** : avec les groupes Evénements, Gestion SFC, Horloge Système

9. Assistant FFB:

Il permet d'ajouter un FFB dans un programme quelque soit le langage de programmation utilisé, il n'y a que la représentation de ce bloc diffère. On y accède depuis la barre d'outils de chaque éditeur de programme ou bien en faisant un clic droit au milieu de cet éditeur.

On peut donc ajouter des Bloc EFB ou des fonctions EF qui sont prédéfinies et disponibles dans la bibliothèque, dans ce cas il faut juste affecter des variables aux entrées/sorties. On peut aussi créer un bloc DFB en définissant nos entrées/sorties et en programmant la fonction de ce bloc.

En choisissant un bloc, il est automatiquement ajouté dans le programme, il reste à définir les variables d'entrées/sorties. Cette affectation diffère d'un langage à un autre.

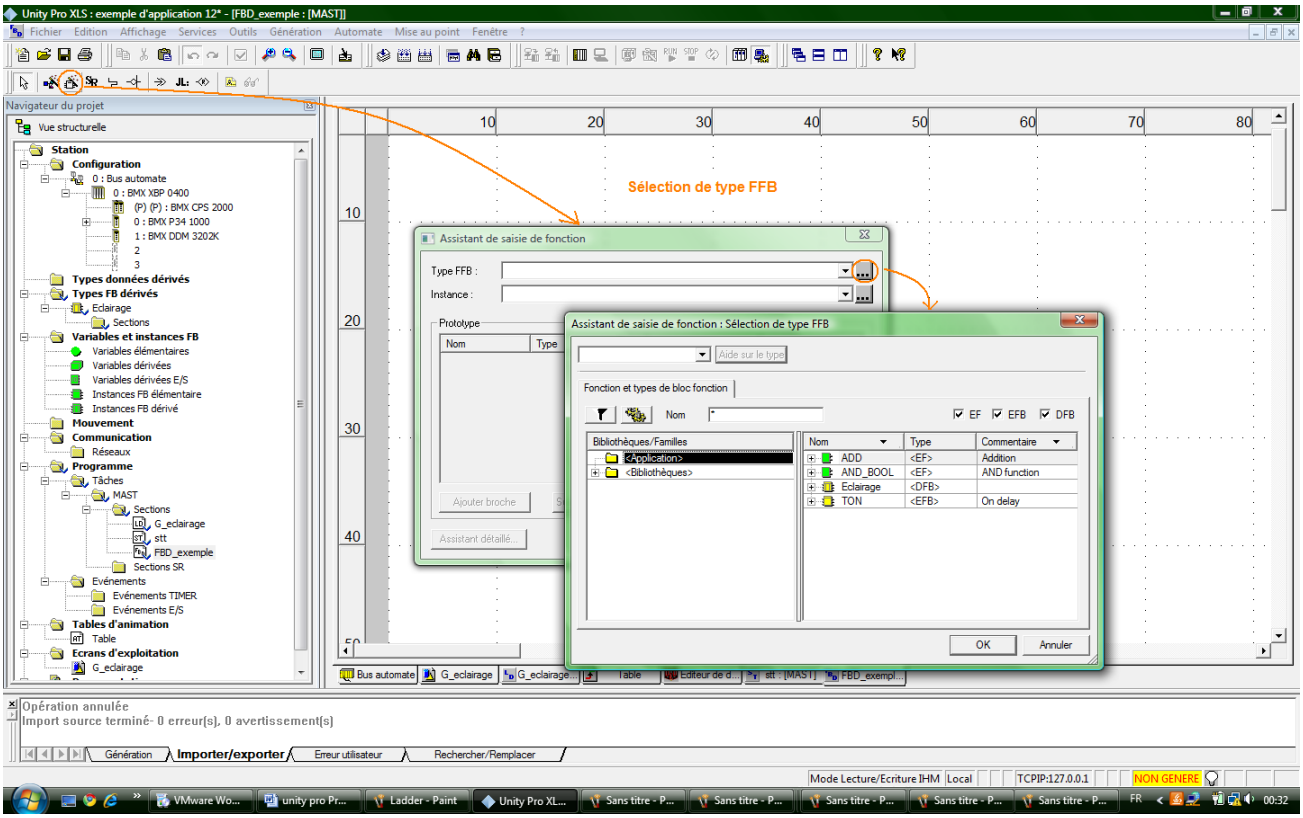


Figure II.29 : Assistant FFB.

10. Simulation avec Unity Pro :

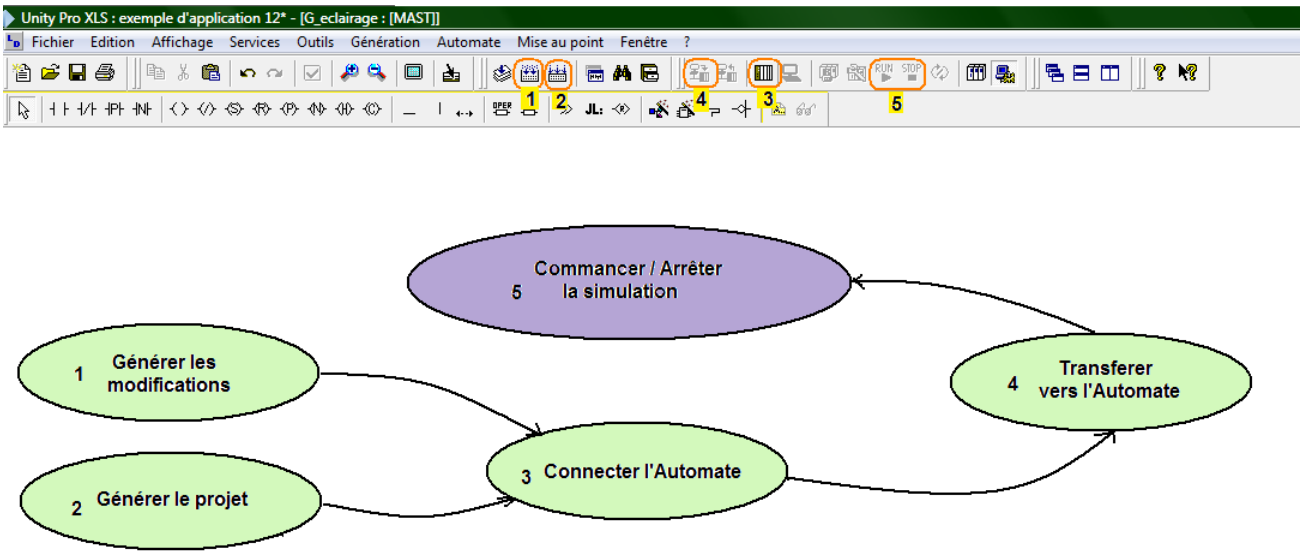


Figure II.30: Simulation du programme.

Pour exécuter le programme il faut :

1) Générer le projet ou bien générer les modifications.
2) Connecter l'automate.
3) S'il n'y a pas d'erreurs détectées, charger le programme
4) Exécuter le programme.

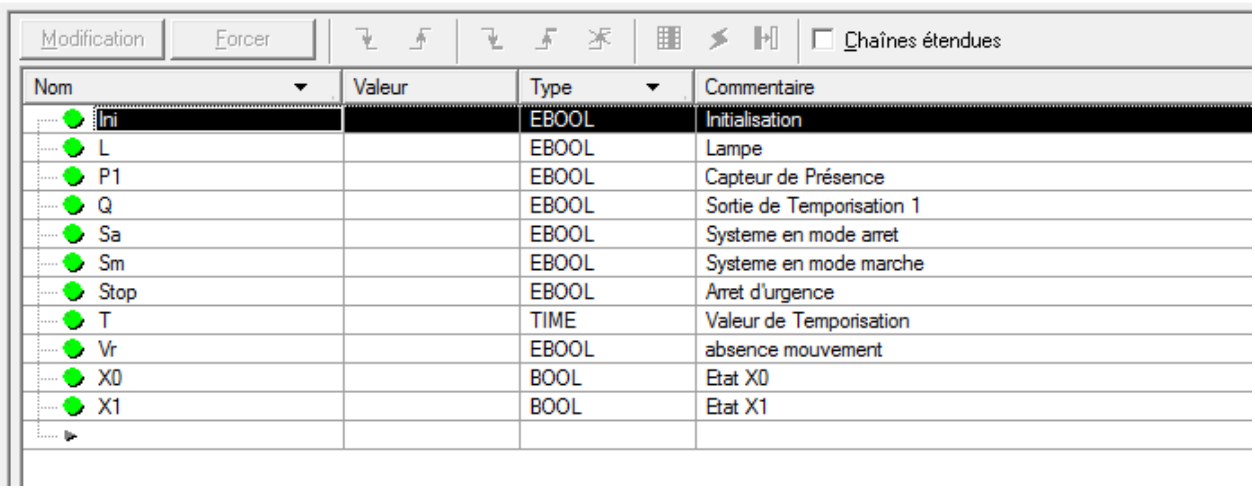
S'il n'y a pas d'automate connecté on peut visualiser l'exécution soit avec la table d'animation, soit avec les écrans de visualisation.

10.1. Table d'animation :

On y accède depuis le navigateur de projet, elle permet de :

Modifier les variables internes grâce à l'onglet ***« modification »***, soit en en inscrivant la valeur dans le champ correspondant soit en utilisant la mise à 1/0 (pour les booléens)

Forcer l'état d'une variable affectée à un module d'entrées grâce à l'onglet ***« Forcer »***.et ce en inscrivant la valeur ou en utilisant aussi la mise à 1 ou mise à 0 comme le montre la figure suivante :



Nom	Valeur	Type	Commentaire
Ini		EBOOL	Initialisation
L		EBOOL	Lampe
P1		EBOOL	Capteur de Présence
Q		EBOOL	Sortie de Temporisation 1
Sa		EBOOL	Systeme en mode arret
Sm		EBOOL	Systeme en mode marche
Stop		EBOOL	Arret d'urgence
T		TIME	Valeur de Temporisation
Vr		EBOOL	absence mouvement
X0		BOOL	Etat X0
X1		BOOL	Etat X1

Figure II.31 : Table d'animation

10.2. Ecran d'exploitation :

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

• Le navigateur projet qui permet de naviguer dans les écrans et lancer les différents outils (l'éditeur graphique, l'éditeur de variables, l'éditeur de messages, ...),

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

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

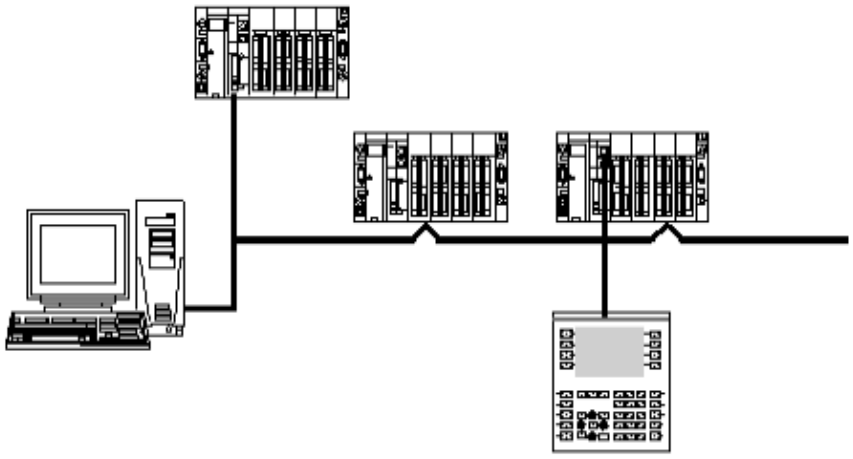


Figure II.32 : Exemple de structure d'automatisme qui utilise des écrans d'exploitation

Dans cette structure, on trouve :

• L'automate qui contient le projet d'automatisme qui gère le procédé.

• Le terminal qui contient le projet d'automatisme avec les écrans d'exploitation. Il est connecté aux automates par la liaison console ou par un réseau.

• Les écrans d'exploitation visualisent le procédé et peuvent être commandés par le clavier du terminal, la souris ou un pupitre de commande connecté aux automates.

10.2.1. Création d'un écran d'exploitation :

Pour accéder aux écrans d'exploitations, il faut Visualiser le projet selon la vue structurelle (Affichage →Vue Structurelle), puis déployer le dossier écran d'exploitation. Si on veut ouvrir un nouvel écran il faut choisir « Nouvel écran » dans le menu contextuel du dossier écran d'exploitation.

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

• les objets standards : ligne, rectangle, ellipse, courbe, poly-ligne, texte. Ils peuvent être statiques ou dynamiques (ayant une variable associée modifiant leur affichage).

• les images : fichiers bitmap avec l'extension BMP ou JPG,

• les objets de pilotage (ou de commande) : bouton, case à cocher, champ de saisie, compteur, curseur, objet d'échange explicite, bouton de navigation écran…Ils sont activés

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

• les objets composés : ensemble d'objets des 3 types précédents, créé par l'utilisateur ou en provenance de la bibliothèque d'objets.

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

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

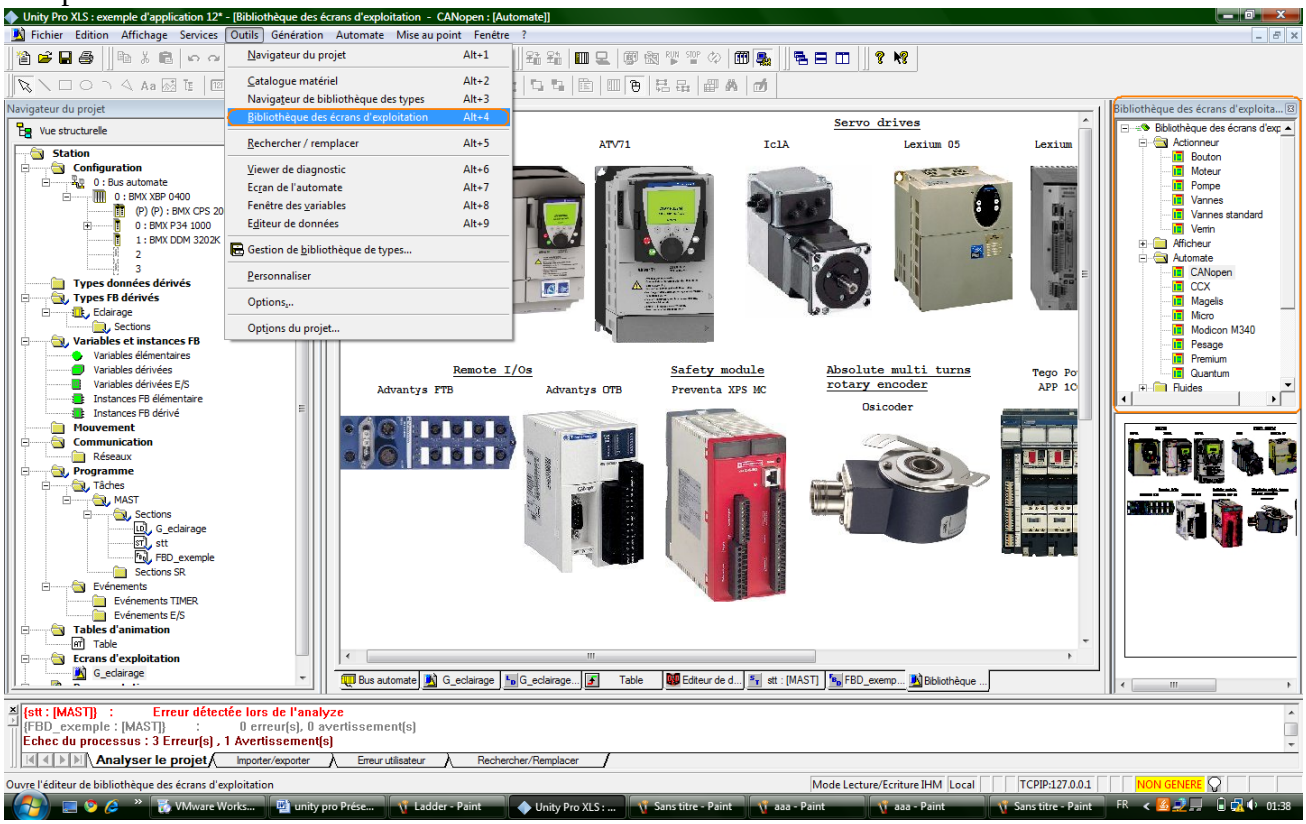


Figure II.33 : Bibliothèque des écrans d'exploitations

11. Exemples d'applications [8] :

11.1. Gestion d'éclairage et contrôle d'accès (en LADDER) :

11.1.1. Cahier de charge :

Le système est défini par une salle équipée de capteur qui vont détecter la présence ou pas d'un individu ou de lumière au cours de la journée ou bien de la nuit
Pour cela on a réalisé le scénario suivant :

Durant la journée,

- En cas de mouvement la lampe s'allume et reste allumée tant que il y'a un individu dans la salle, quand la personne quitte la salle, une temporisation est lancée juste après pour éteindre la lampe
- Si la personne revient avant que la temporisation soit écoulée, la lampe reste allumée et la tempo est réinitialisée.
- Si on a un accès interdit, une temporisation de T=10s est lancée et la lampe s'allume, si la personne quitte la salle avant que cette tempo s'écoule, on réinitialise la tempo et la lampe s'éteint après T=15s, Si la personne reste dans la salle, une alarme se déclenche.

Pendant la nuit,

- si le capteur détecte un accès à n'importe quelle salle, l'alarme est lancée.

L'écran d'exploitation :

Afin de simplifier l'utilisation du programme on a crée l'écran suivant en utilisant des images externe et d'autre prédéfini dans la bibliothèque d'Unity Pro.

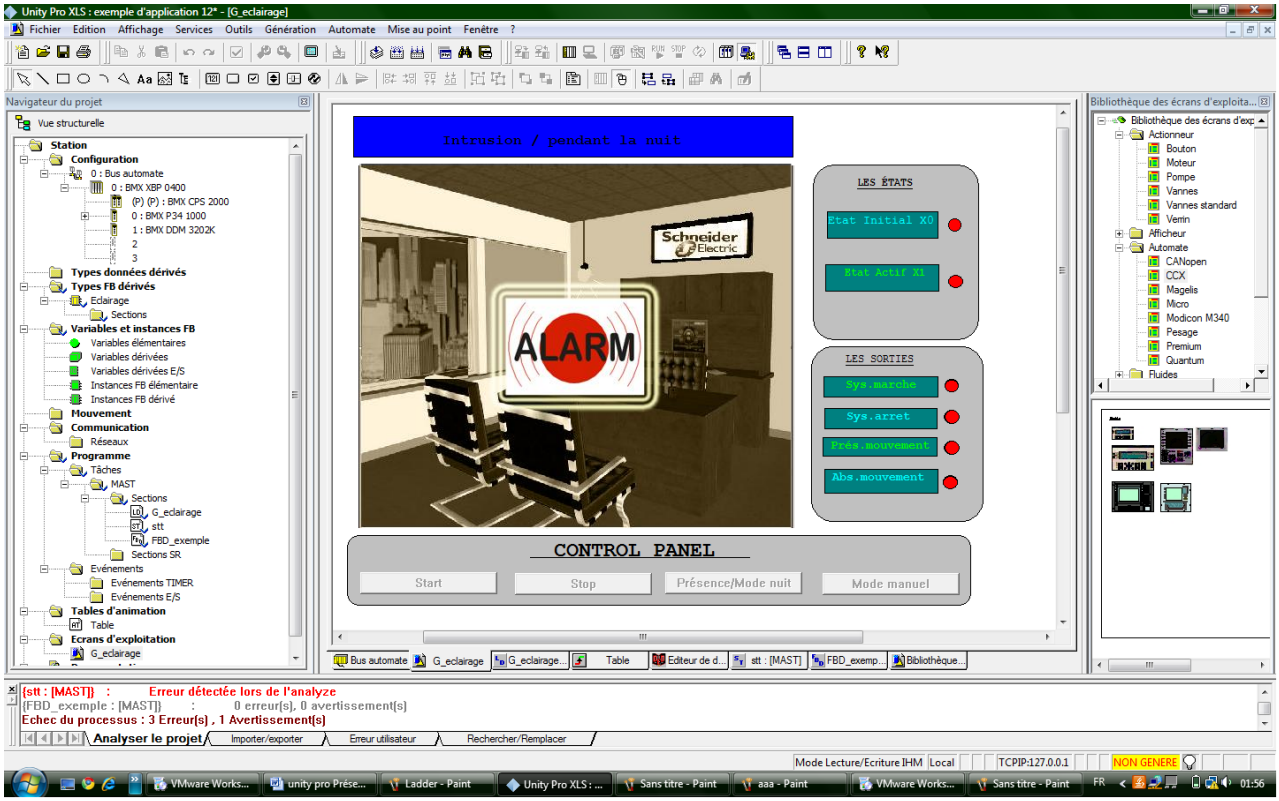


Figure II.34 : Ecran d'exploitation

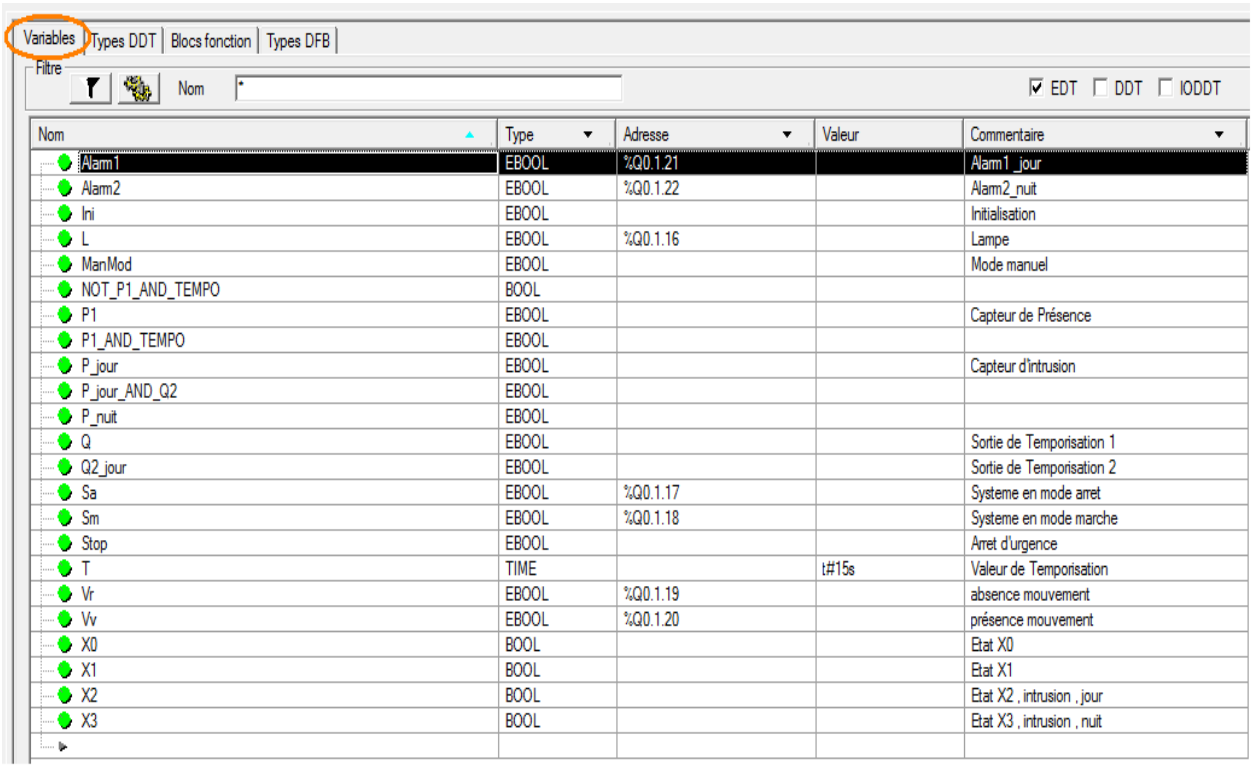
L'écran comporte :

- Un bouton Star pour initialiser le cycle

- Un bouton Stop pour désactiver l'alarme et réinitialiser le cycle
- Un bouton Présence/Mode_Nuit pour forcer la variable P1
- Un bouton Mode_Manuel pour passé du mode Auto→Manuel

11.1.2. Les Variable :

Le tableau suivant montre les variables utilisées dans la programmation et la mise en marche de ce système sous Unity Pro.



Nom	Type	Adresse	Valeur	Commentaire
Alarm1	EBOOL	%Q0.1.21		Alarm1 _jour
Alarm2	EBOOL	%Q0.1.22		Alarm2_nuit
Ini	EBOOL			Initialisation
L	EBOOL	%Q0.1.16		Lampe
ManMod	EBOOL			Mode manuel
NOT_P1_AND_TEMPO	BOOL			
P1	EBOOL			Capteur de Présence
P1_AND_TEMPO	EBOOL			
P_jour	EBOOL			Capteur d'intrusion
P_jour_AND_Q2	EBOOL			
P_nuit	EBOOL			
Q	EBOOL			Sortie de Temporisation 1
Q2_jour	EBOOL			Sortie de Temporisation 2
Sa	EBOOL	%Q0.1.17		Systeme en mode arret
Sm	EBOOL	%Q0.1.18		Systeme en mode marche
Stop	EBOOL			Arret d'urgence
T	TIME		t#15s	Valeur de Temporisation
Vr	EBOOL	%Q0.1.19		absence mouvement
Vv	EBOOL	%Q0.1.20		présence mouvement
X0	BOOL			Etat X0
X1	BOOL			Etat X1
X2	BOOL			Etat X2 , intrusion , jour
X3	BOOL			Etat X3 , intrusion , nuit

Figure II.35 : Déclarations des variables

11.1.3. Programmation en langage SFC :

La figure suivante montre le diagramme du programme en langage SFC.

Tous les détails concernant les étapes de programmations son en [**Annexe B].**

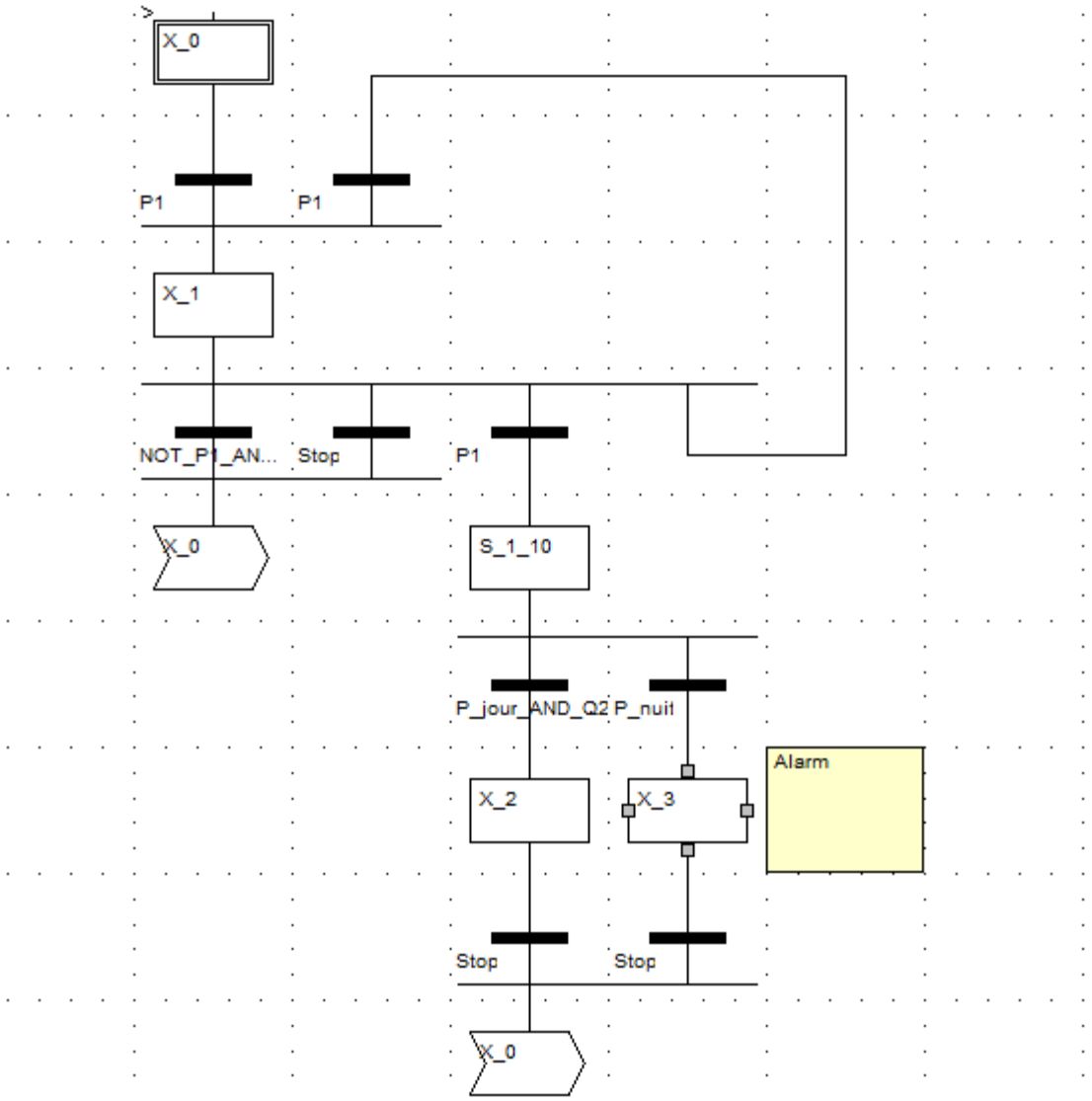


Figure II.36 : Grafcet (Programmé en SFC)

11.1.4. Programmation LADDER :

Il suffit de retranscrire la réalisation matérielle en langage (LD). Pour ce faire on a besoin des contacts, des bobines et d'un bloc de temporisation. On utilise des « mémento (Mi) » pour chaque « état (Xi) », ces mémentos permettent la sauvegarde de l'état.

On a aussi besoin d'un bloc fonction <<Tempo>> pour la temporisation pour chaque sortie, le programme

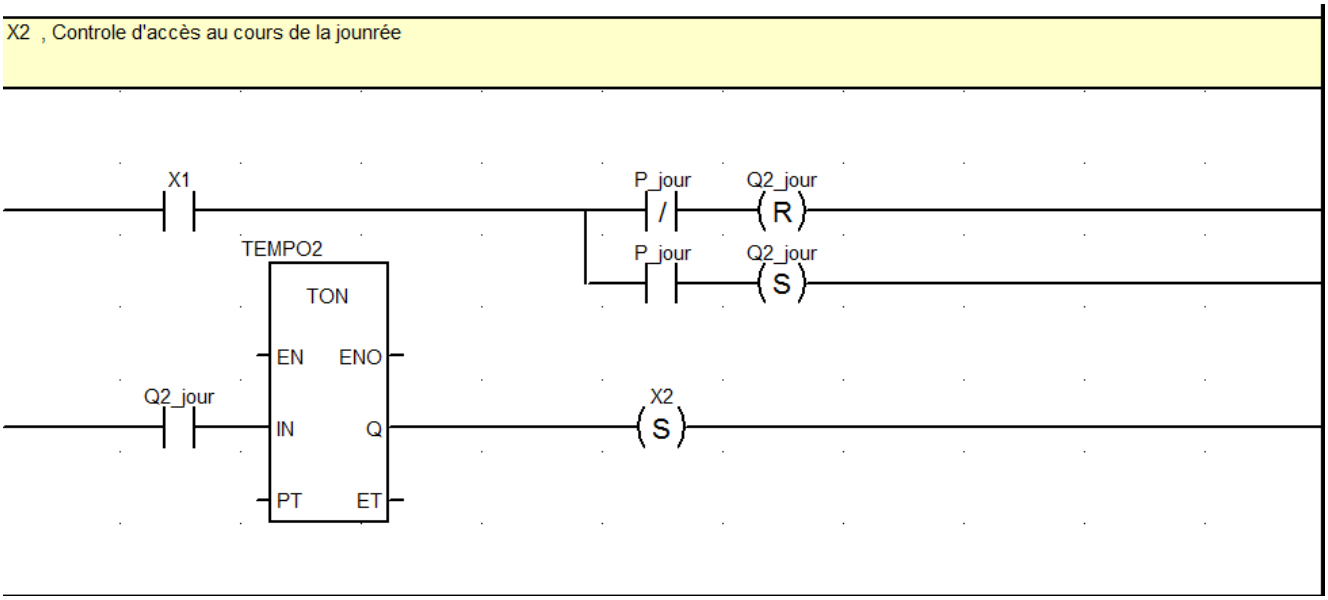


Figure II.37: Programmation avec le LADDER

11.1.5. La simulation des états :

Les figures suivantes les différents écrans suivant les états actifs du système

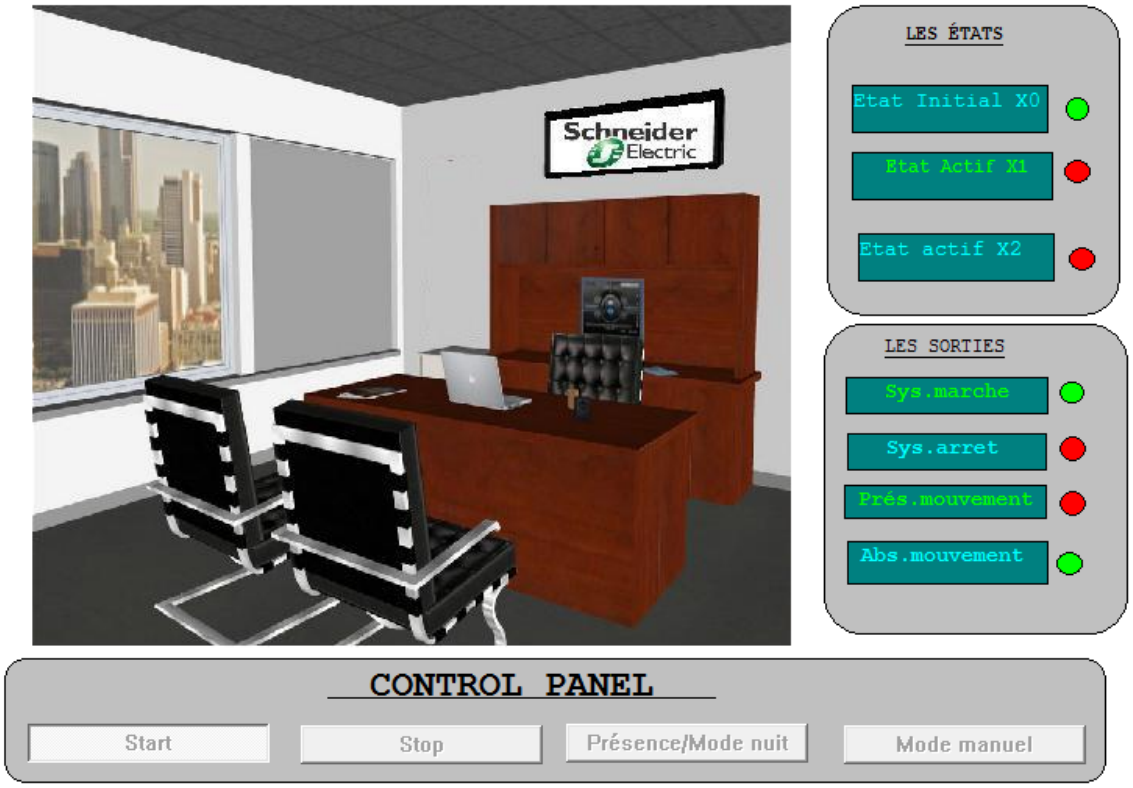


Figure II.38: Etat Initial X0.

Figure II.39: Etat X1 (Nuit, ou Présence)

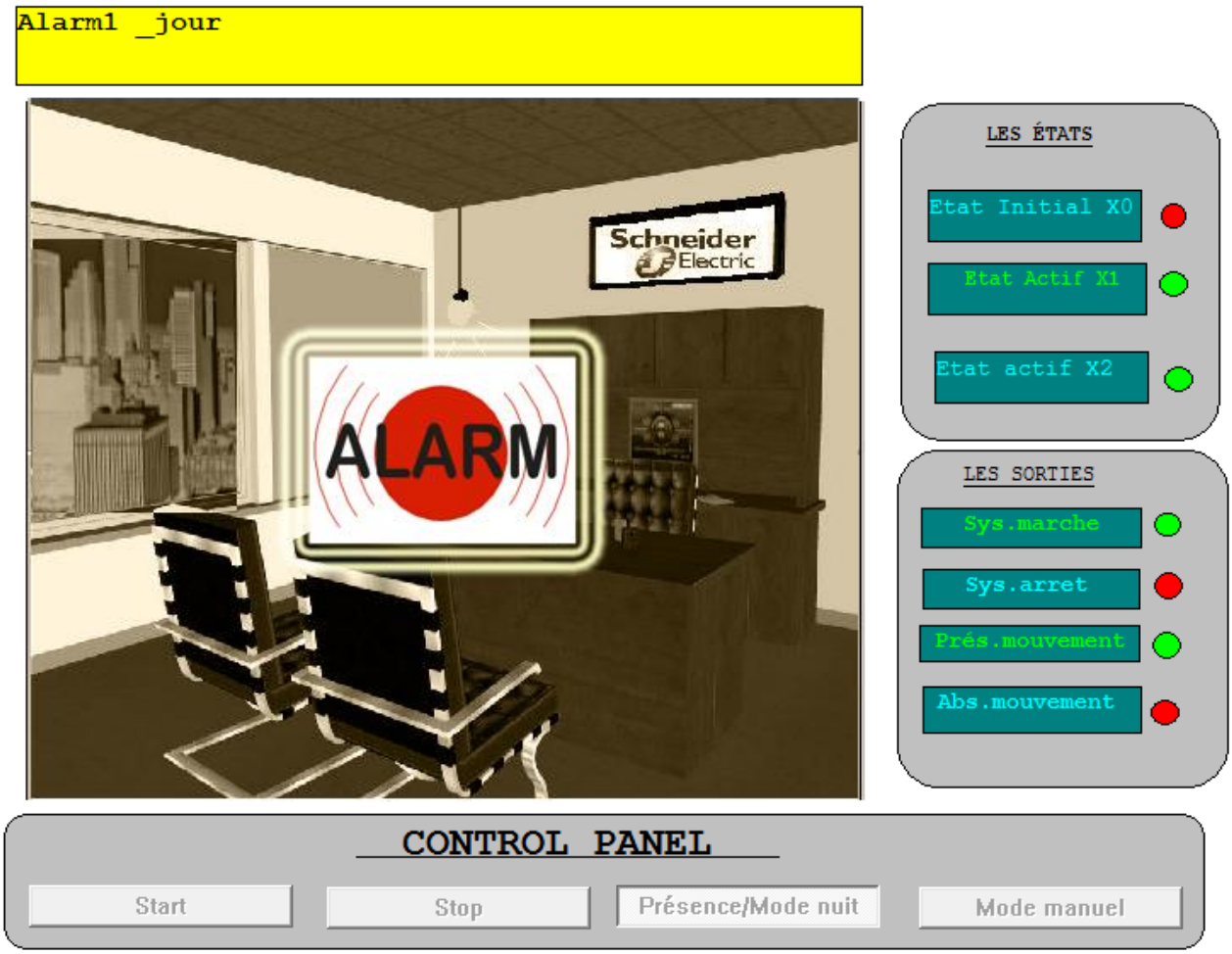


Figure II.40: X2 /X3: Etat d'alarme

11.2. Exemple de programmation du FBD en ST:

On veut commander la lumière automatiquement ou manuellement dans salle équipée de *Trois types de capteurs :*

- De présence
- De lumière
- Fin de course Rideau

Deux boutons :

- Mode manuel ou automatique
- Un interrupteur intelligent qui peut êtres commandé par l'automate ou par l'utilisateur.

Cahier de charge :

Pour cela on va créer deux blocs fonction programmé en langage ST :

- **Bloc Erreur :**

Le bloc Erreur est programmé afin de détecter les différentes pannes du système et de les traduire en un signaler de sortie affiché sous forme de message sur l'écran du superviseur, qui peut par la suite identifier et localiser cette panne avec précision.

- **Bloc Eclairage :**

Ce bloc est programmé pour lire les données provenant des capteurs et du Bloc d'Erreur, et de commander l'interrupteur pour allumer ou éteindre la lampe suivant un scénario bien définit
Il doit aussi calculer le temps d'allumage de lampe pour pouvoir le traduire en Energie totale consommée

11.2.1 Création du bloc DFB :

On doit tout d'abord créer un bloc DFB (Eclairage par exemple) ensuite en on va le programmer en choisissant le langage ST.

Création du Bloc DFB Eclairage :

La figure suivante montre comment créer le bloc « Eclairage » et comment déclarer ses variables Entrées/Sorties.

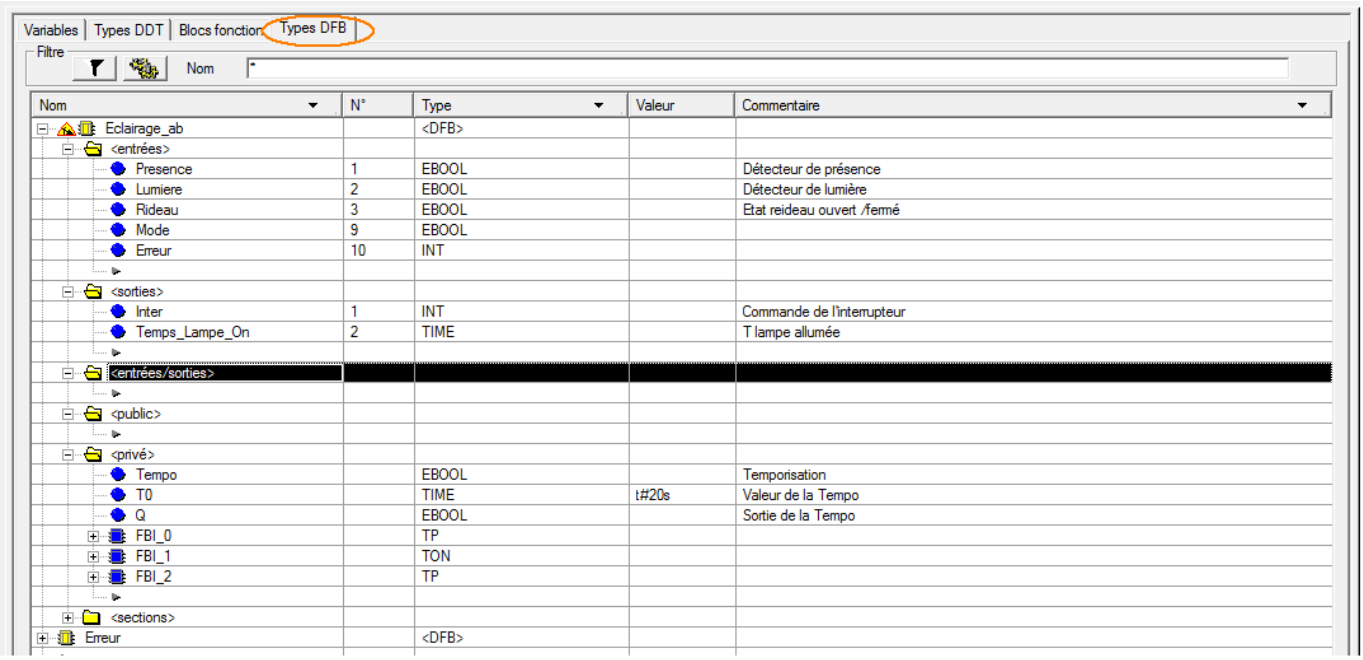


Figure II.41: Création du bloc DFB Eclairage

11.2.2 Programmation de la Section en ST :

Afin de programmer le bloc Erreur on doit créer une nouvelle section et choisir le ST comme langage de programmation ensuite double clique sur cette section pour pouvoir l'éditer.

La figure suivante décrit ces deux étapes.

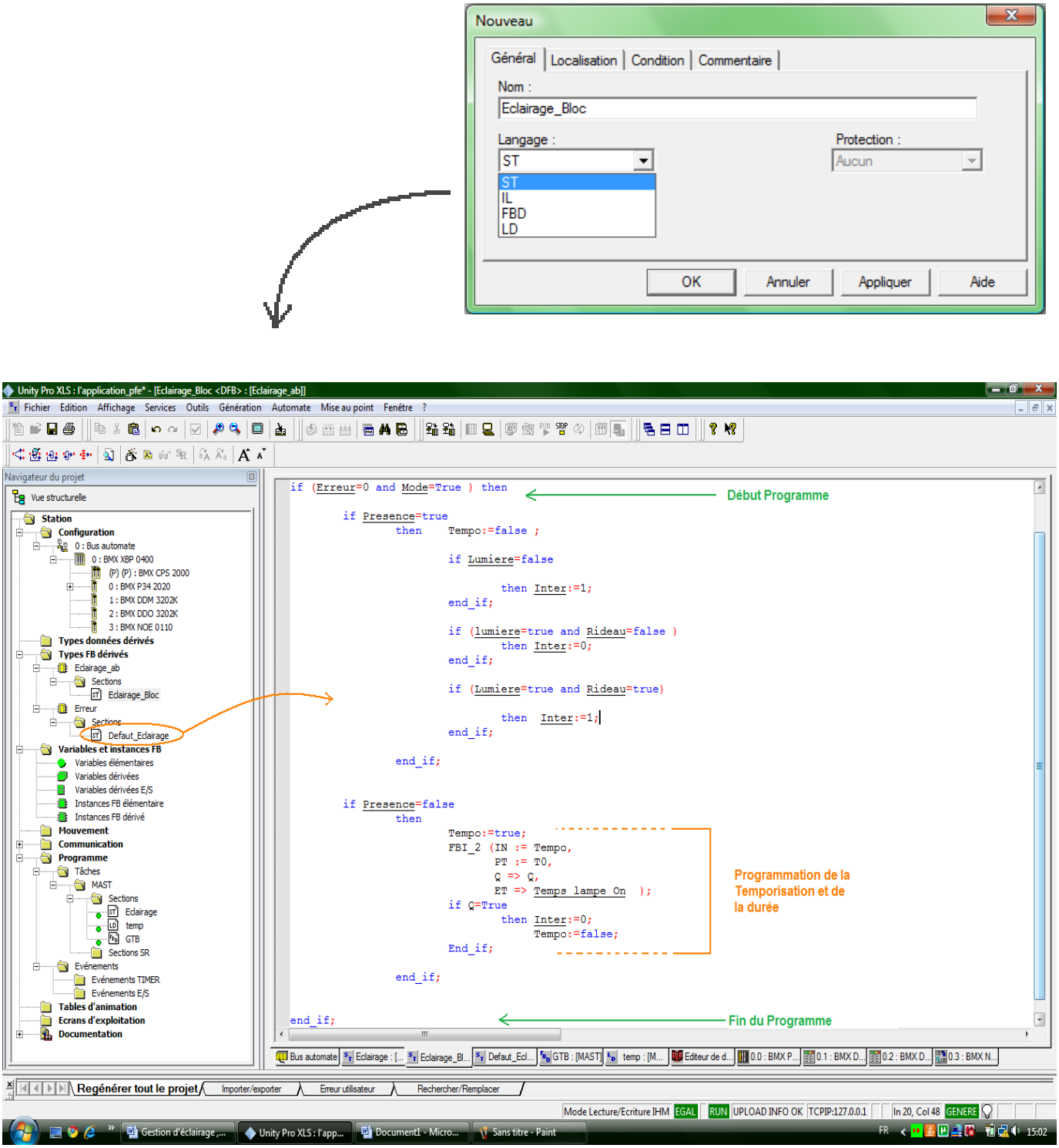


Figure II.42: Création et programmation de la section ST

Le programme détaillé se trouve dans **[Annexe B].**

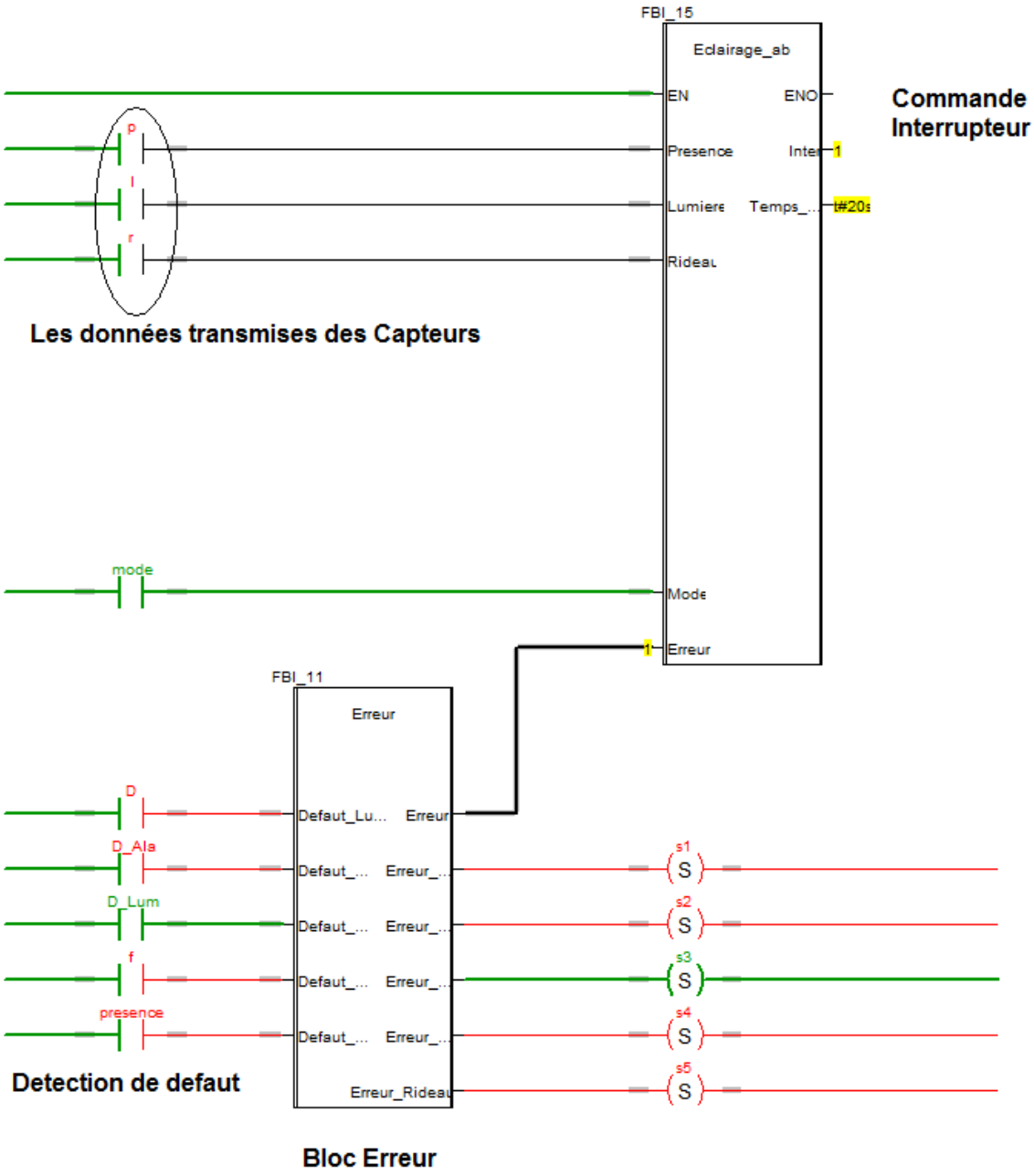


Figure II.43 : Bloc FBD, simulé en LADDER

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. La possibilité de créer des blocs de fonctions utilisateurs « DFB » et de les intégrer à la bibliothèque est un avantage de taille. On peut ainsi élaborer nos propres fonctions et les utiliser dans n'importe quel projet

CHAPITRE III

VIJEO CITECT

Introduction :

Dans un projet industriel, il est toujours nécessaire de faciliter le contrôle des différentes entités de l'usine (ou bâtiment) 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 Citect.

1. Description générale :

Vijeo Citect est un système HMI performant qui est utilisé sous Microsoft Windows 2000 et Windows XP. HMI signifie "Human Machine Interface", il s'agit donc de l'interface entre l'homme (l'utilisateur) et la machine (le process). Le contrôle proprement dit du process est assuré par les automates programmables (API). Une communication s'établit donc entre Vijeo Citect et l'opérateur d'une part et entre Vijeo citect et les automates programmables d'autre part.

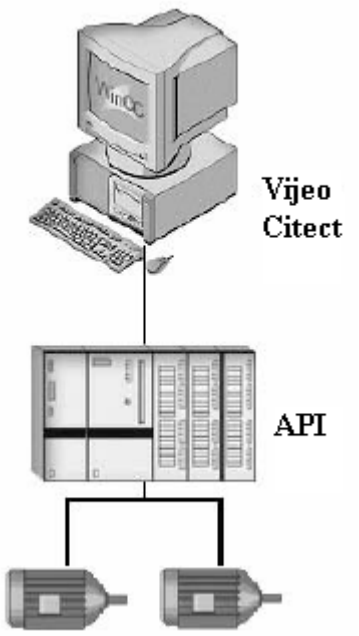


Figure III.1 : Schéma de communication entre Vijeo Citect et l'API.

Vijeo Citect permet de visualiser le process et de concevoir l'interface utilisateur graphique destinée à l'opérateur.

- WinCC permet à l'opérateur de surveiller le process. Pour ce faire, le process est visualisé par un graphisme à l'écran. Dès qu'un état du process évolue, l'affichage est mis à jour.
- WinCC permet à l'opérateur de commander le process. A partir de l'interface utilisateur graphique ; il peut par exemple entrer une valeur de consigne ou ouvrir une vanne.
- Lorsqu'un état de process devient critique, une alarme est déclenchée automatiquement. L'écran affiche une alarme en cas de franchissement d'un seuil défini par exemple.

- Les alarmes et valeurs de process peuvent être imprimées et archivées sur support électronique par WinCC. Ceci vous permet de documenter la marche du process et d'avoir accès ultérieurement aux données de production du passé.

2. Création d'un nouveau projet [9]:

Cette section décrit la marche à suivre afin de créer un projet pour la première fois.

Pour démarrer Vijeo Citect, dans le menu Démarrer de Windows on pointe la souris sur programmes, Schneider Electric, Vijeo Citect, Vijeo Citect 7.0, Explorateur Vijeo Citect.

Dans la barre d'outils cliquer sur l'icône « Nouveau » 

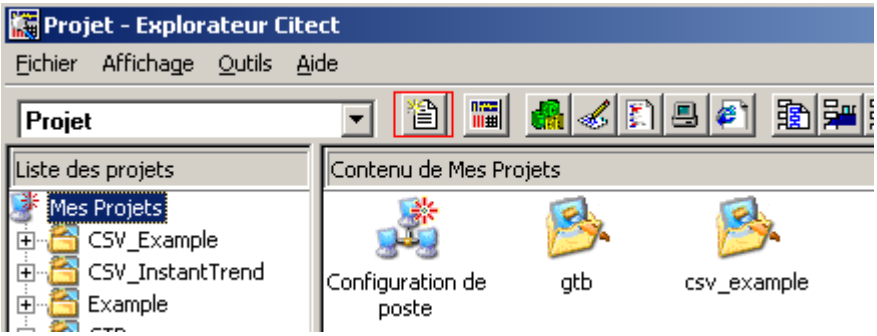
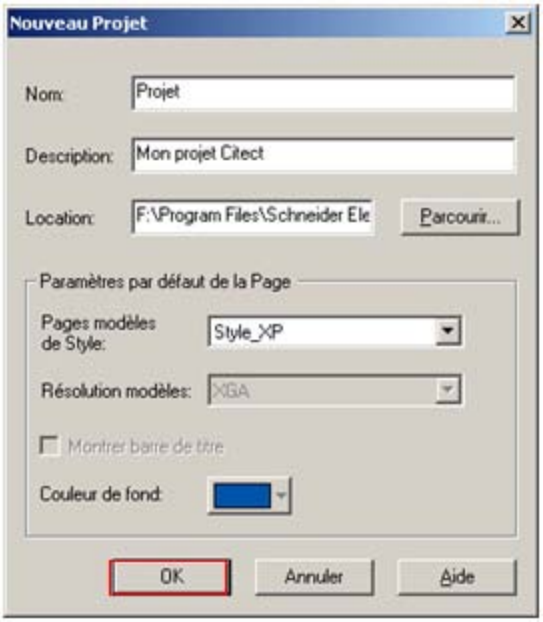


Figure III.2 : Création d'un nouveau projet sous Vijeo Citect

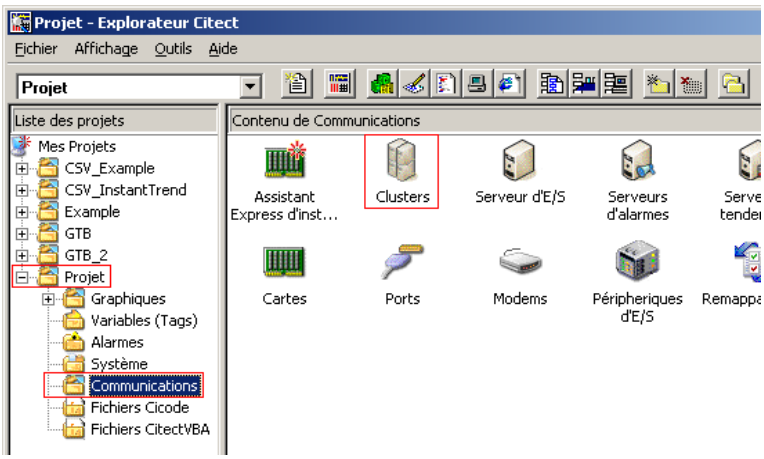
Dans la boite de dialogue « Nouveau Projet », mettre le nom du projet, puis une description et cliquer sur OK. On peut aussi changer le style de la fenêtre si on le désire.



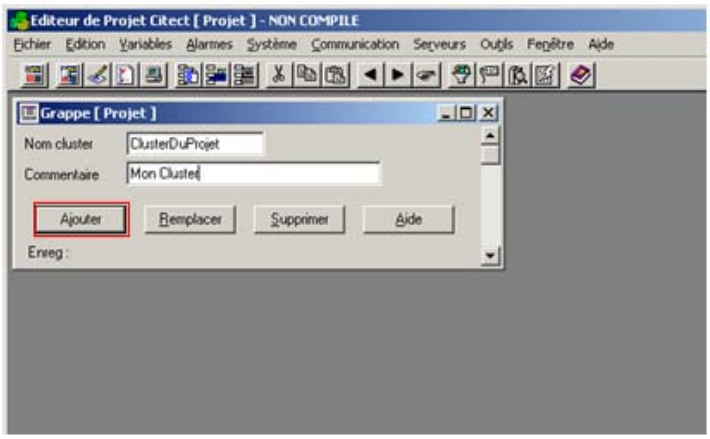
2.1 Configuration des Clusters :

Le cluster est un groupe de serveurs d'alarmes, de tendances, de rapports et d'E/S. En général, un cluster contient également des clients de visualisation Vijeo Citect locaux. Si l'installation comprend plusieurs sections ou systèmes, on peut utiliser plusieurs clusters à raison d'un par section. On appelle aussi les clusters, Grappes, en Français.

Etendre l'arborescence du projet à la gauche de l'Explorateur Citect et sélectionner Communications. Double cliquer maintenant sur l'icône Clusters.

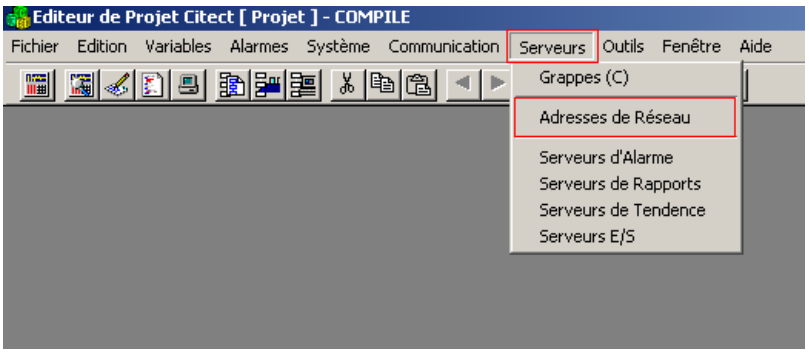


L'éditeur de Projet Citect devrait apparaître. Là donner le nom du Cluster et lui donner un commentaire approprié puis cliquer sur Ajouter.



Fermer la fenêtre Grappe.

Cliquer sur le menu Serveurs en haut de la fenêtre Editeur de Projet Citect, puis sur « Adresses de Réseau ».



Là ajouter le nom de l'adresse, l'adresse (127.0.0.1) pour une simulation interne car c'est l'adresse réseau interne de l'ordinateur, et ajouter un commentaire approprié. Puis cliquer sur Ajouter, et fermer la fenêtre.

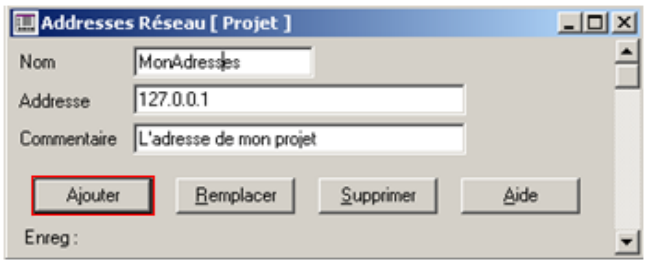


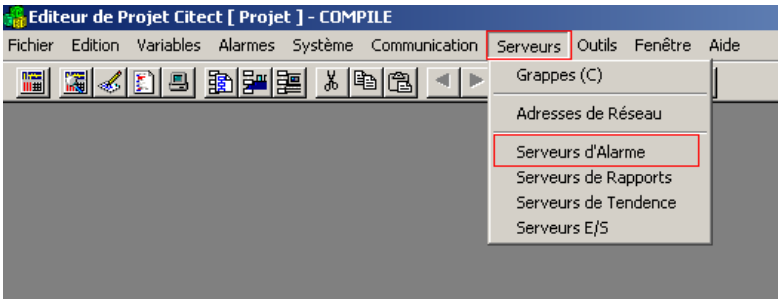
Figure III.3 : Ajout d'un réseau sous Vijeo Citect

On a mis l'adresse interne de l'ordinateur car on va dans cet exemple relier notre projet au simulateur d'automate de Unity Pro qui a comme adresse l'adresse interne de l'ordinateur, si on veut le relier à un véritable automate on doit donner l'adresse réelle de l'automate.

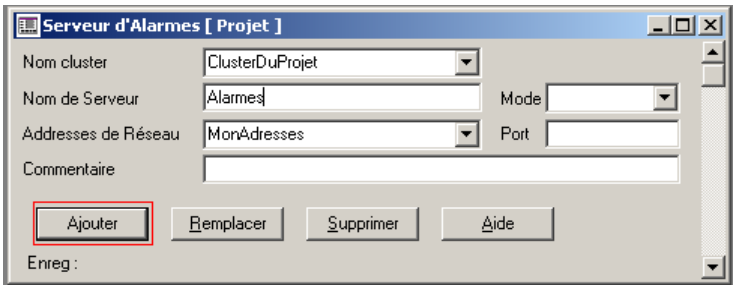
Cela assigne l'adresse TCP/IP locale au système, qui est approprié aux systèmes autonomes seulement.

Maintenant assigner le cluster et l'adresse réseau aux divers rôles du serveur requis dans le système Citect. Cela inclut les communications (entrées/sorties), les alarmes, les rapports et les tendances.

Depuis le menu serveurs, sélectionner « Serveursd'Alarmes ».



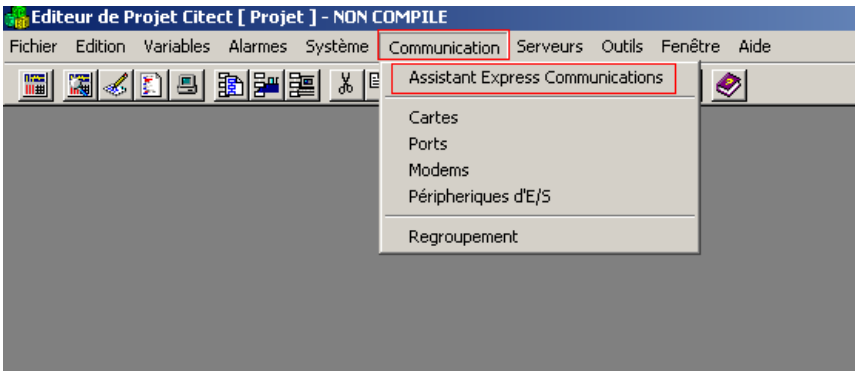
Puis faire dérouler les menus, « Nom cluster » et « Addresses de Réseau », pour choisir le cluster et le réseau, ajouter maintenant le nom de l'alarme et un.



On fait la même chose pour les serveurs de rapports, de Tendances et d'entrées sorties.

2.2. Configuration des périphériques d'entrée/sortie :

Toujours dans l'éditeur de projet Citect cliquer sur communication puis sur assistant express communications.



Cliquer sur suivant pour les trois premières étapes, puis à la quatrième choisir périphérique d'E/S disque ou externe selon le cas, si on veut travailler avec un système réel choisir externe sinon choisir disque, pour l'exemple choisir de travailler sur le disque.

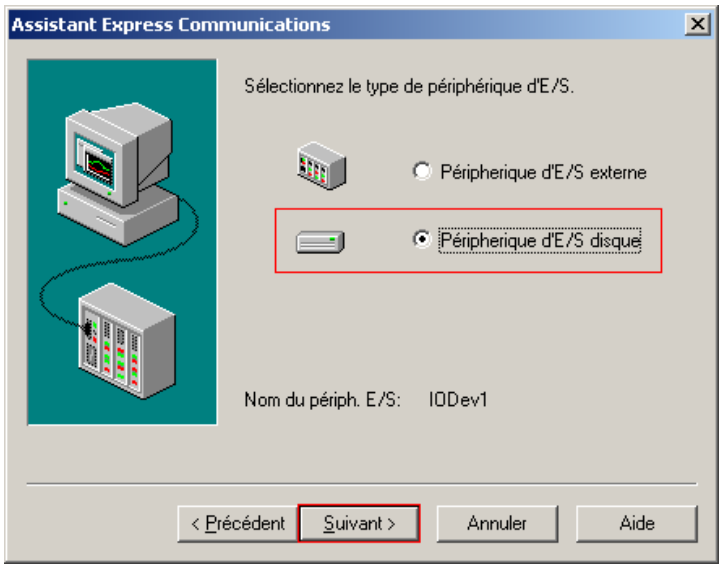
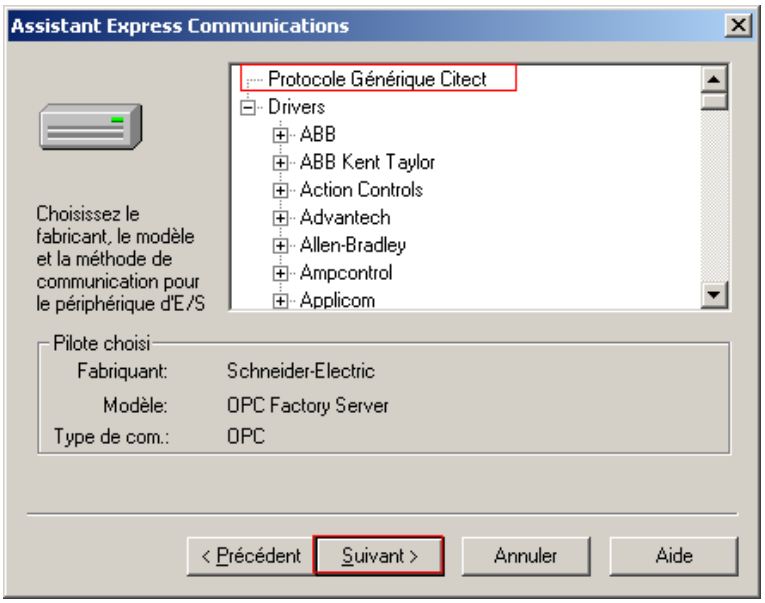


Figure III.3 : Assistant de configuration de periphériques d'E/S sous VijeoCitect

A la cinquième étape choisir le protocole de communication, pour l'exemple nous choisirons « Citect Generic Protocol » sinon on doit choisir le protocole à utiliser.



Puis pour les deux dernières fenêtres il ne reste qu'à cliquer sur suivant et terminer.

Le système a donc été configuré. On a besoin de configurer les tags de l'automate que le système va utiliser pour contrôler l'équipement. Pour l'exemple on contrôlera une pompe, son mode de fonctionnement et sa vitesse.

2.3. Configuration des Tags :

Les tags fournissent un lien entre l'opérateur et le monde réel. Pour l'exemple créer trois tags pour représenter une pompe, un tag de statut marche/arrêt, un tag de fonctionnement automatique/manuel, et un tag de contrôle de vitesse.

Cliquer sur « Variables » dans votre projet dans l'Editeur de Projet Citect. Puis double cliquer sur « Variables ».

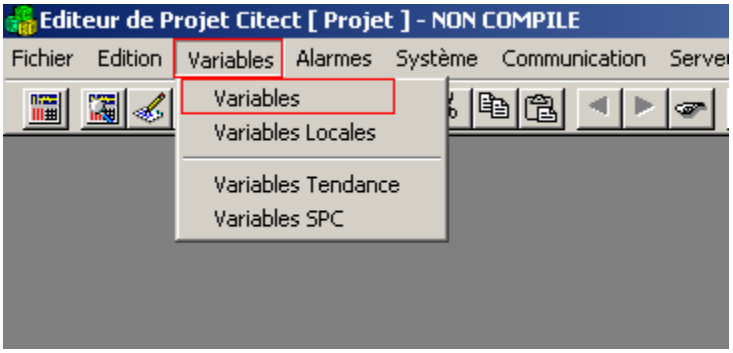
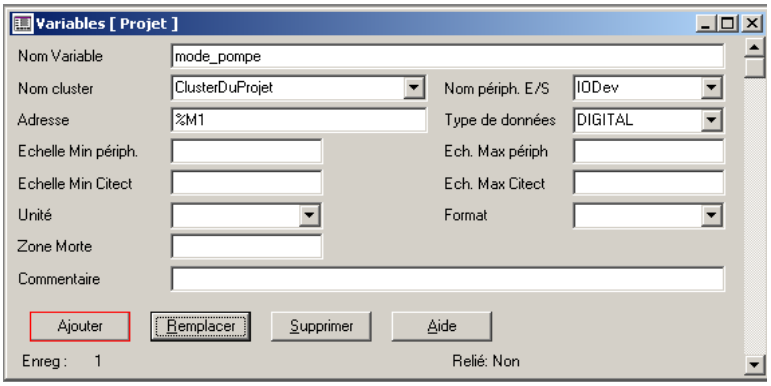
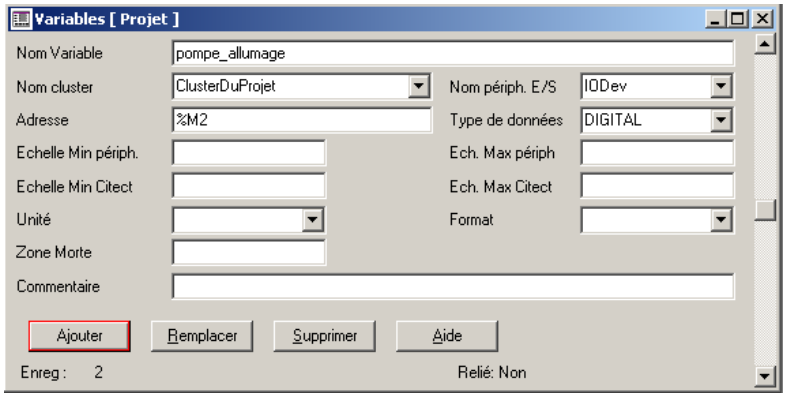
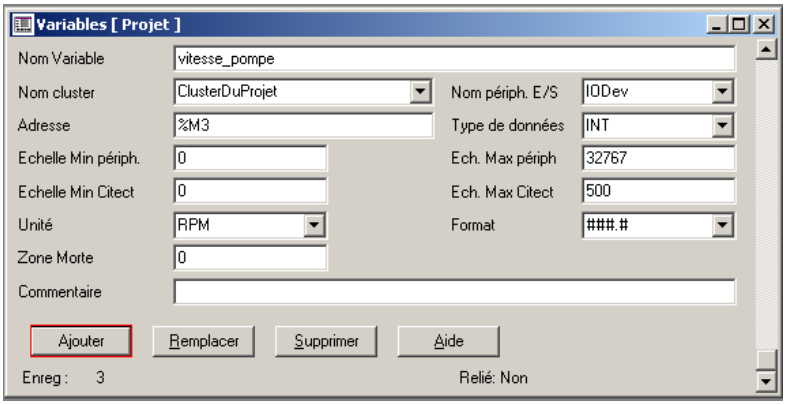


Figure IV.5 : Ajout de Variables sous Vijeo Citect

Ceci va ouvrir la boite de dialogue, « Variable Tags », qui sert à configurer des variables. Remplir le formulaire avec le nom du tag de la variable, le cluster utilisé, l'adresse, où il faut mettre l'adresse mémoire qu'elle a dans l'automate, le périphérique d'entrée sortie, le niveau vide et le niveau plein en mode brut et en mode mis à l'échelle, l'unité de la variable ainsi que son format et pour finir le type de la variable. Il est aussi possible d'ajouter un commentaire. Après la fin de la configuration des variables cliquer sur « Ajouter » et configurer les autres variables.







Créer un programme Unity Pro pour l'exemple contenant trois variables, la première « mode_pompe » qui va être égale à 1 quand la pompe sera en mode manuel et égale à 0 quand la pompe sera en mode automatique, mettre dans le memento %M1, la deuxième variable représente l'état de la pompe, si elle est allumée ou éteinte, à mettre dans le memento %M2 et enfin la troisième pompe représente la vitesse de la pompe, à mettre dans le memento %M3.

Après avoir fini fermer la fenêtre de configuration de variables et revenir à l'explorateur de Vijeo Citect.

2.4. Création des pages graphiques :

2.4.1. Présentation :

Pour cet exemple le but est de créer une page simple qui permet à l'opérateur d'allumer ou d'éteindre une pompe, de contrôler son mode de fonctionnement, manuel ou automatique, et un curseur qui permet de contrôler sa vitesse.

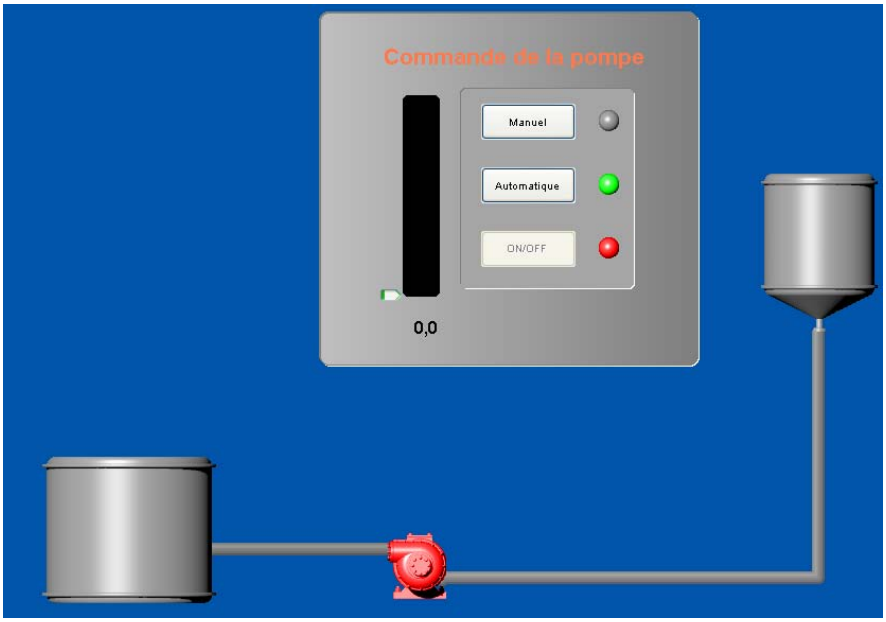
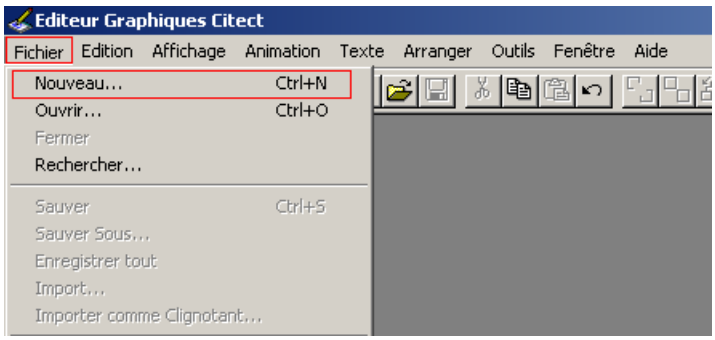


Figure III.6 : Fenêtre de projet sous Vijeo Citect

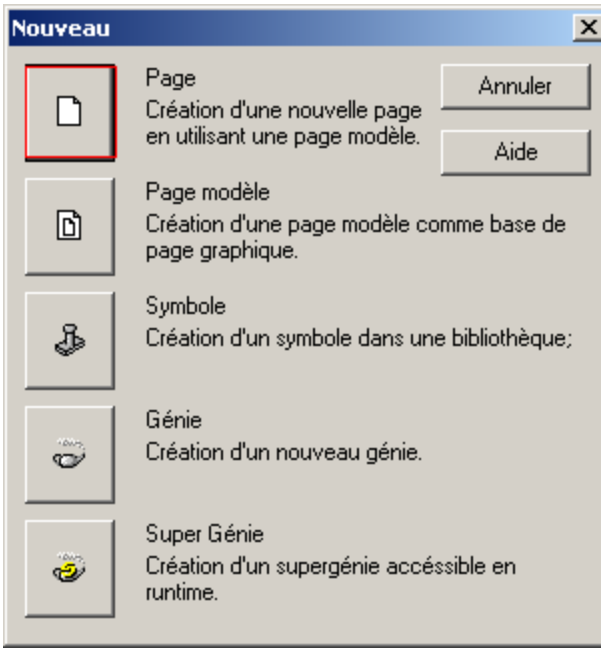
La pompe prendra la couleur verte quand elle fonctionnera et rouge lorsqu'elle sera à l'arrêt, des lampes indiqueront le mode de fonctionnement et enfin une jauge et un nombre indiqueront la vitesse de rotation de la pompe.

2.4.2. Création d'une nouvelle page :

Aller dans l'éditeur graphique Citect, cliquer sur l'icône Nouveau.



Un pop up apparaît qui permet de sélectionner le type de page à créer. Cliquer sur le bouton « Page ».



Un autre pop up apparaît qui permet d'utiliser une page modèle. Choisir le modèle voulu parmi tous les modèles dans tous les styles différents. Prendre par exemple le modèle « normal » dans « xp_style ». Dans ce modèle, il y'a déjà plusieurs boutons prédéfinis qui aident à la navigation pour commencer.

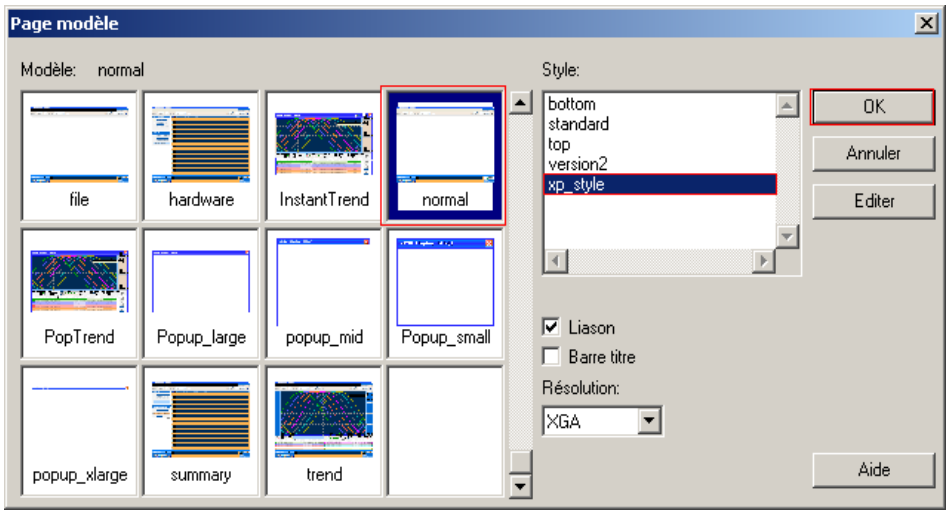


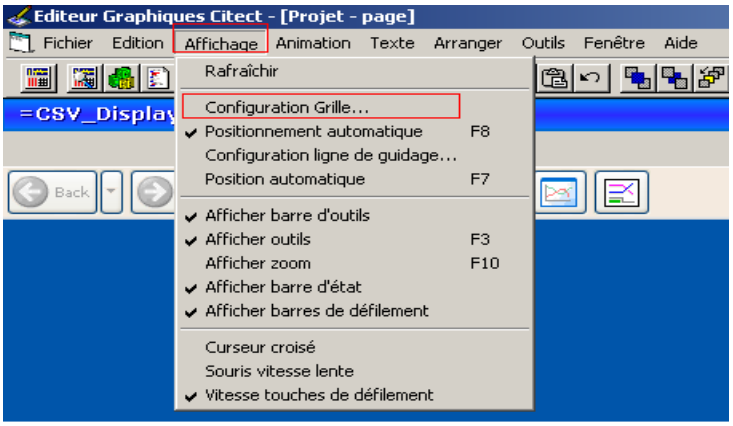
Figure III.7 : Pages modèles sous Vijeo Citect

Sauvegarder en utilisant le bouton enregistrer  . La première fois que la page est sauvegardée, il faut créer le fichier et donc lui donner un nom. Pour l'exemple prendre « Page » comme nom.

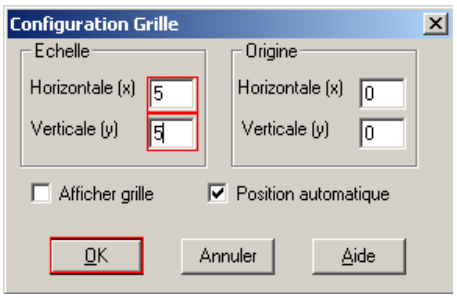
2.4.3. Configuration de la grille :

Il est possible de configurer une grille qui permet au curseur de se déplacer dans des positions uniformément positionnées, ce qui facilitera la tache pour positionner les objets de la même façon.

Sur la barre d'outils, cliquer sur affichage puis sélectionner « Configuration Grille… ».



Une boite de dialogue apparaît, changer l'échelle comme voulu, en général 5x5 est conseillé. Cocher « Position automatique » puis valider en cliquant sur Ok.



La page est à présent prête pour la création et le positionnement des objets graphiques.

2.4.4. Configuration des boutons :

Création des boutons pour commander la pompe.

Dans les outils, cliquer sur l'icône Bouton.

Cliquer et maintenir le bouton de la souris et former le bouton voulu, puis relâcher le bouton.

Quand le bouton de la souris est relâché, un pop up « propriétés de bouton » apparaît. Changer le nom du bouton dans le champ « Texte », appeler ce premier bouton « Manuel », qui sera le bouton qui mettra la pompe en fonctionnement manuel. Cliquer sur Entrée en haut du pop up pour configurer l'action qu'effectue ce bouton.

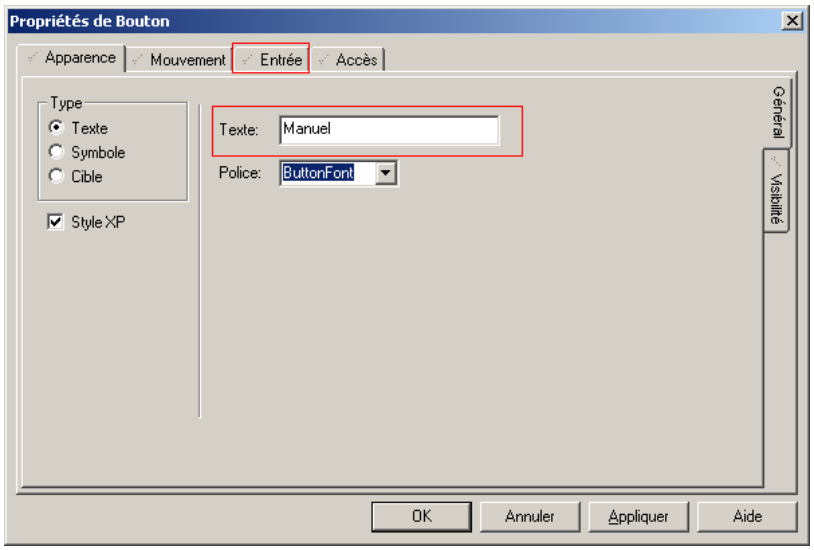
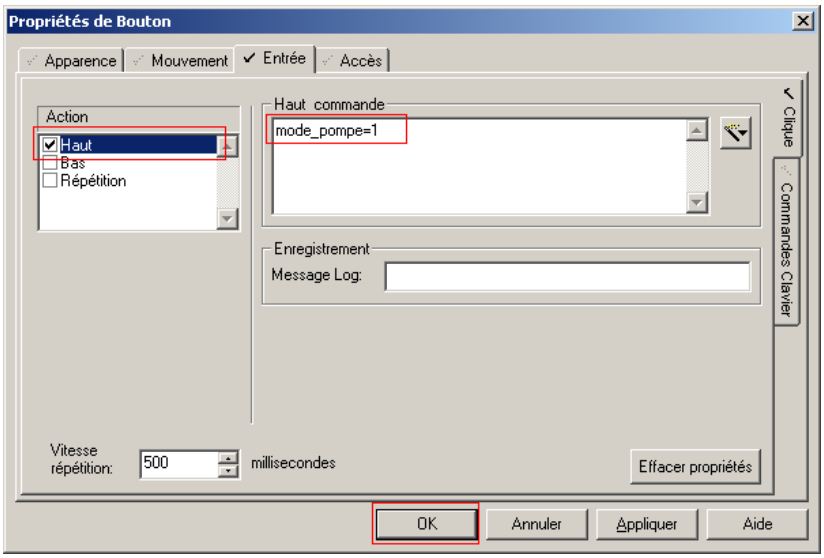


Figure III.8 : Configuration de boutons sous Vijeo Citect

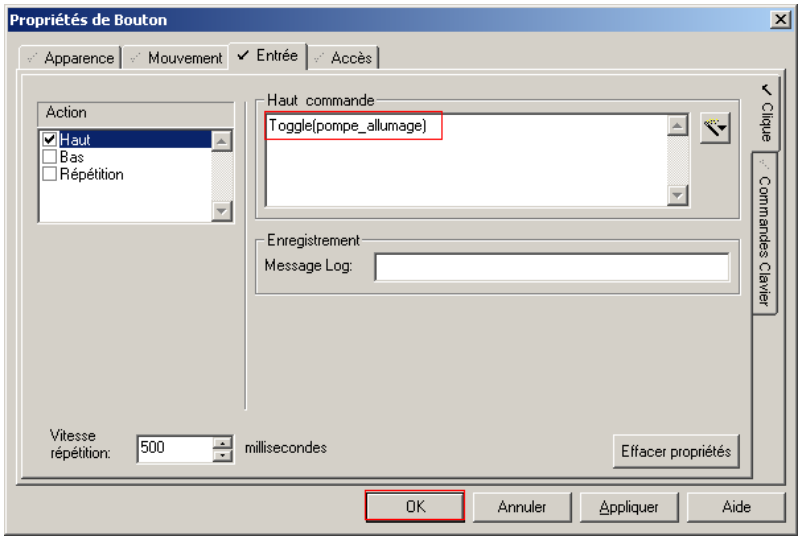
Dans le bouton insérer on peut insérer des variables ou des fonctions prédéfinies. Pour l'exemple prendre « mode_pompe », mettre que cette variable sera égale à 1 dans commande haut en cochant « Haut » dans actions et en mettant « mode_pompe=1 » dans le champ Haut commande puis cliquer sur OK.



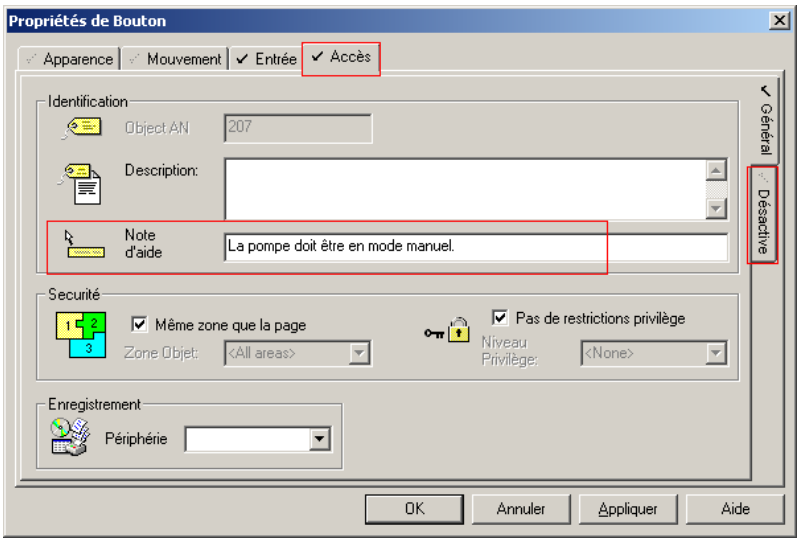
Dans ce cas en cliquant sur le bouton manuel, la pompe se mettra en mode de fonctionnement manuel, et même en recliquant plusieurs fois dessus, ce pour quoi il faut créer un deuxième bouton qui mettra la variable « mode_pompe » à 0 et qui mettra donc la pompe en fonctionnement automatique.

Pour cela il n'est pas impératif de répéter toutes les actions, il suffit de copier et coller le bouton précédent puis double cliquer sur le bouton et modifier seulement le nom et mettre la fonction inverse, « mode_pompe=0 ».

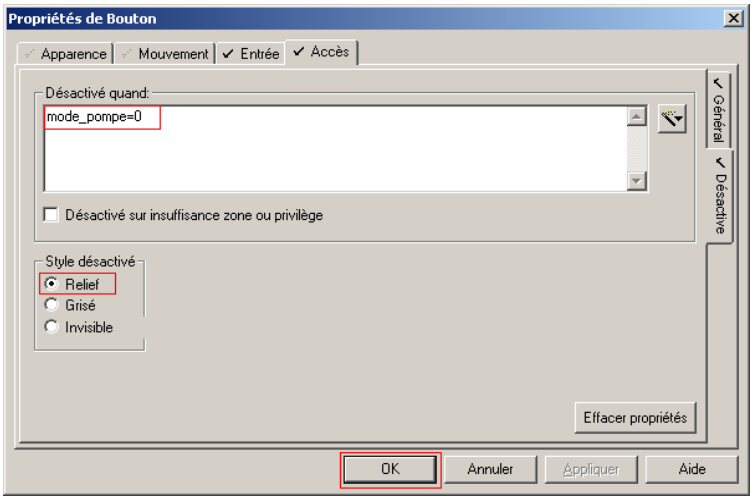
Pour créer un bouton qui allumera ou éteindra la pompe selon son état. Créer un bouton, mettre On/Off comme nom, et dans l'onglet Entrée insérer la variable « Pompe_allumage » et la mettre dans la fonction Toggle, que l'on peut trouver dans les fonctions prédéfinies et qui permet d'allumer la pompe si celle-ci est éteinte et vice-versa.



Ce bouton, ON/OFF, sera inhibé quand la pompe sera en mode automatique et un commentaire d'aide expliquera à l'opérateur que la pompe doit être en mode manuel pour qu'il puisse l'allumer ou l'éteindre pour cela, aller dans l'onglet Accès, dans la note d'aide écrire « La pompe doit être en mode manuel.».Puis cliquer sur « Désactive ».



Là mettre dans le champ « désactivé quand » « mode_pompe=0 » et on peut changer le style quand le bouton est désactivé. Pour l'exemple nous allons le laisser en relief.



2.4.5. Jeu de symboles :

Nous allons maintenant ajouter des symboles pour le renseignement sur l'état de la pompe.

Dans la barre d'outils cliquer sur le bouton « Jeu de symboles ».

Puis placer le symbole à l'emplacement voulu, dans ce cas le placer devant le bouton « manuel ». Dans le champ « Symbole VRAI quand » mettre l'expression voulue, dans ce cas mettre « mode_pompe=1 ». Cliquer maintenant sur le bouton config du symbole FAUX.

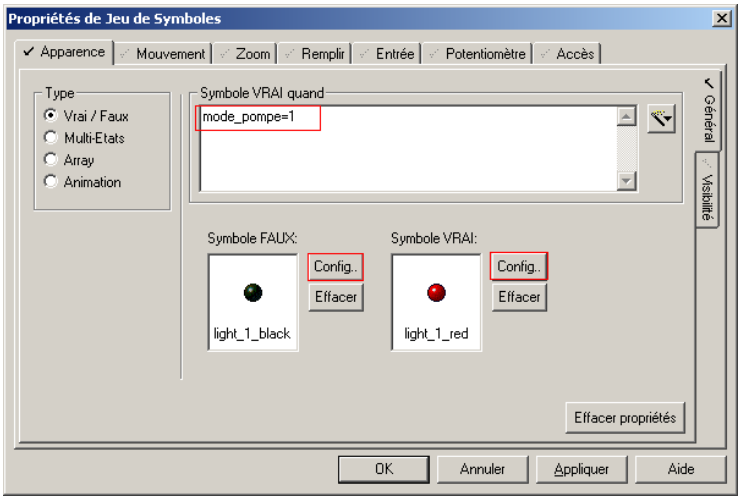
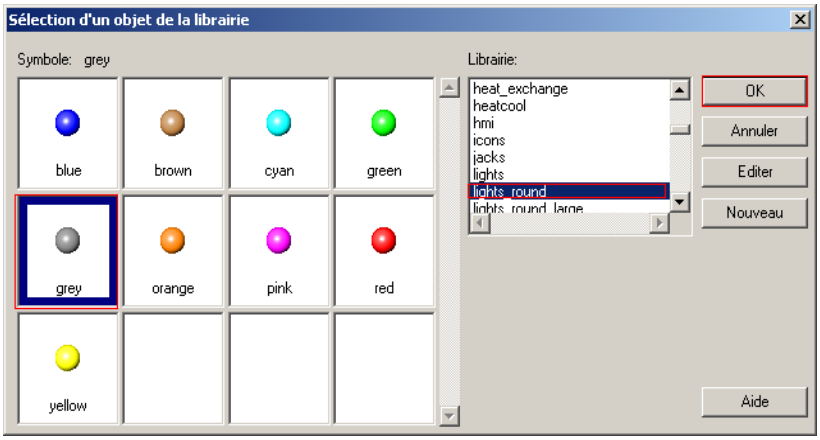


Figure III.9 : Configuration d'un symbole animé sous Vijeo Citect

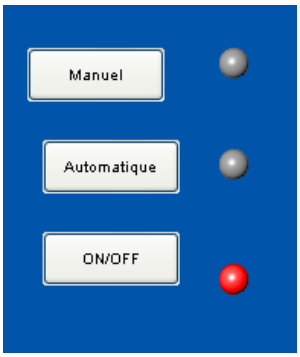
Configurer le symbole à afficher quand l'état est faux, choisir par exemple le symbole lampe grise dans la librairie « lights_round ».



Ceci remplacera la lampe noire qu'il y avait au début. Répéter la même chose pour le symbole VRAI mais cette fois choisir la lampe verte.

Copier le symbole et le mettre devant le bouton « auto » et modifier l'état vrai en mettant « mode_pompe=0 » et confirmer en cliquant sur ok.

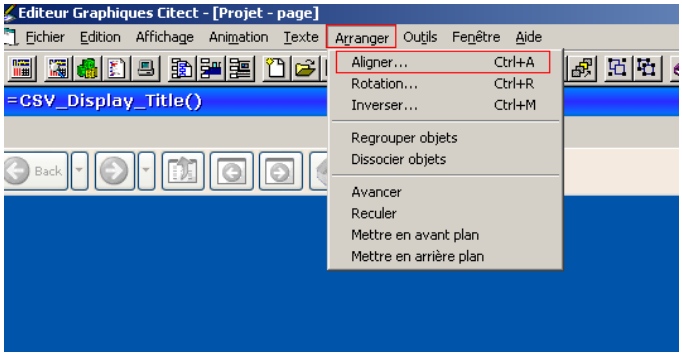
Faire une dernière copie et la mettre devant le bouton ON/OFF et changer l'état vrai par «Pompe_allumage=1 » et changer les symboles en mettant la lumière rouge pour l'état FAUX et une lumière verte pour l'état VRAI.



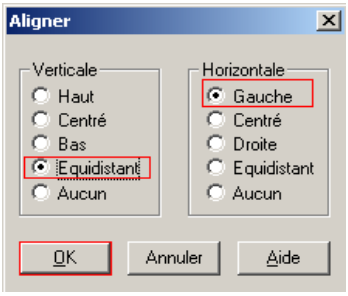
2.4.6. Alignement d'objets :

Comme il est difficile d'aligner les objets manuellement il existe un outil qui le fait automatiquement.

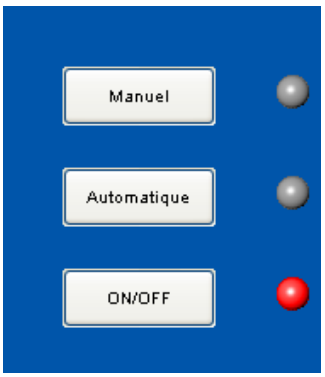
Sélectionner les trois boutons en cliquant sur les trois tout en maintenant la touche CTRL appuyée ou en maintenant le bouton de la souris appuyé et en passant par les trois boutons, puis cliquer dans la barre de menu sur arranger puis sur aligner, ou alors cliquer sur « CTRL+A ».



Un pop up apparaît, choisir le type d'alignement, mettre par exemple en alignement horizontal « Gauche » et en alignement vertical « Equidistant », comme cela les boutons seront alignés horizontalement sur la gauche et verticalement la même distance séparera chaque bouton et le suivant.



Répéter la même opération pour les symboles mais avant cela sélectionner chaque bouton et son symbole et les aligner verticalement sur le centre.



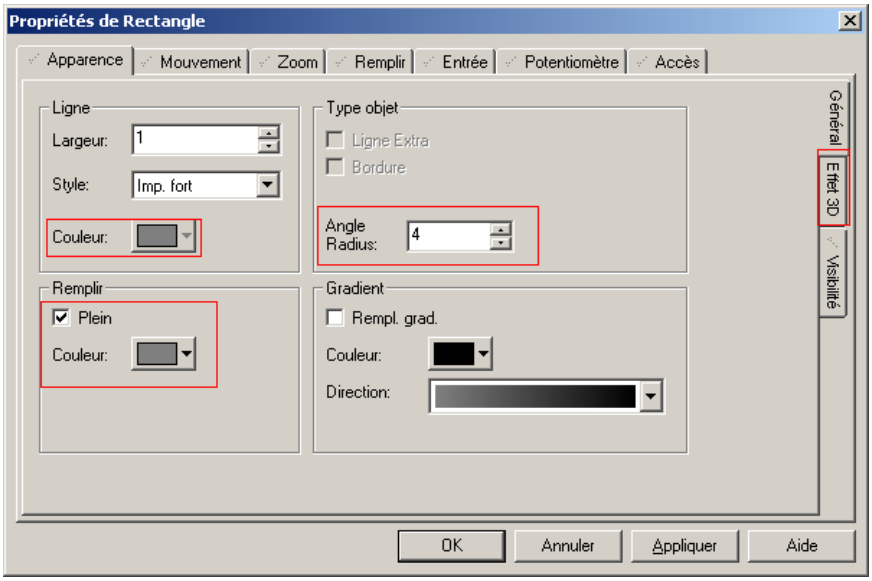
2.4.7. Dessiner les formes :

Pour dessiner un rectangle gris autour des boutons, afin de faire ressortir le panneau de contrôle.

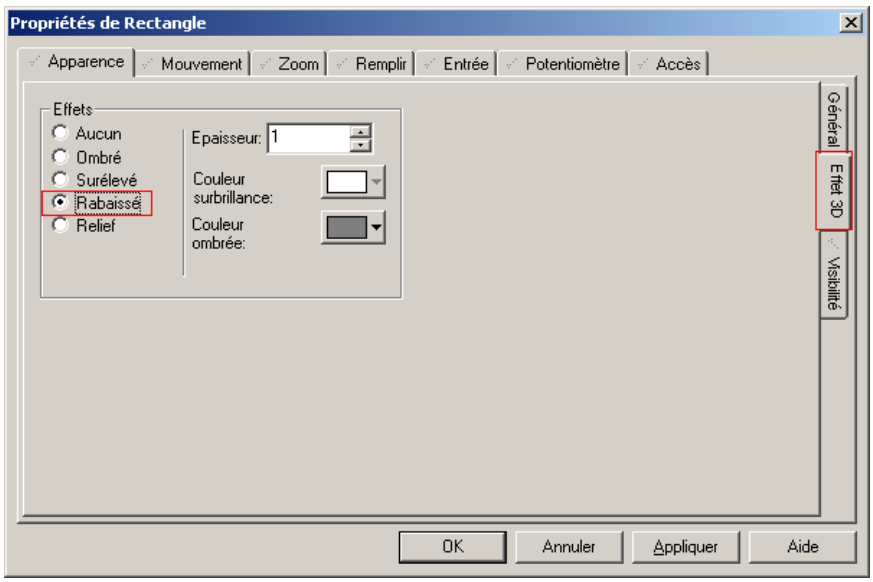
Cliquer sur l'outil rectangle  , puis cliquer sur le bouton de la souris sans le relâcher afin de former un rectangle autour de tous les éléments. Une fois que le rectangle est

positionné il sera toujours possible de l'agrandir en cliquant sur un coin et en maintenant la souris jusqu'à l'obtention de la taille voulue.

Quand on relâche le bouton de la souris, la page de propriétés du rectangle apparaît, il est possible de modifier les propriétés du rectangle dans cette page. Modifier la couleur et cocher la case « plein » et modifier la couleur du remplissage aussi, ajouter un angle de contour pour avoir un effet arrondi en modifiant l'angle radius à quatre par exemple.

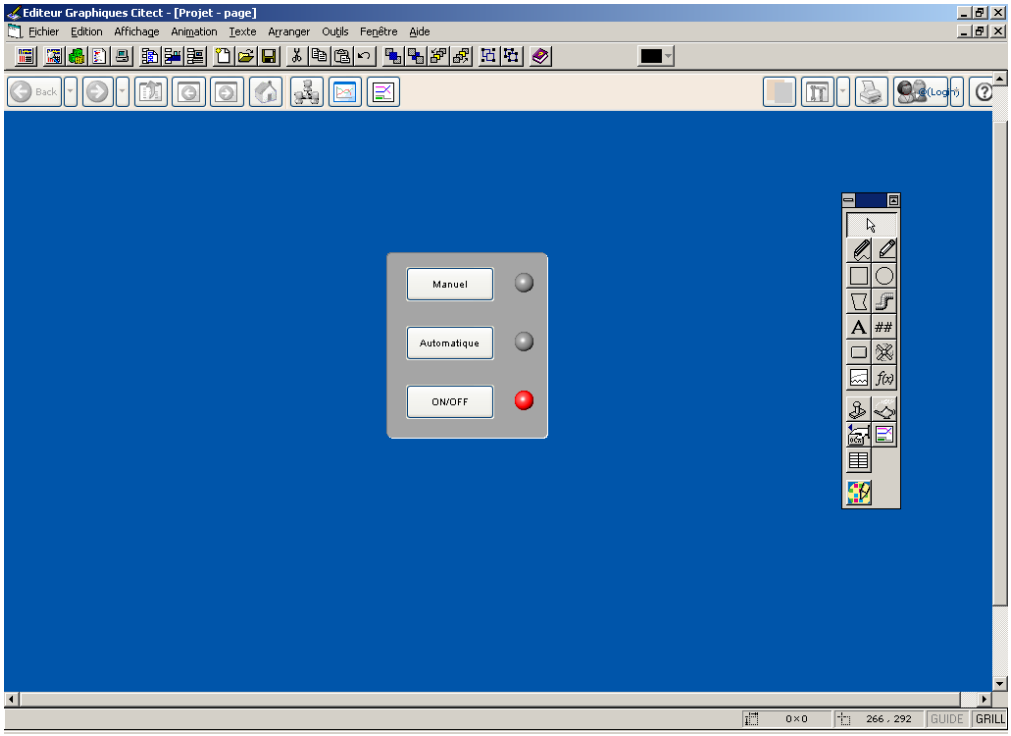


Cliquer sur effet 3d et choisir l'effet Rabaissé, puis cliquer sur ok pour valider.



Le rectangle devrait recouvrir les éléments maintenant, mettre le rectangle en arrière plan. Pour cela cliquer sur le bouton « Reculer »  .

Votre page devrait ressembler à cette figure.

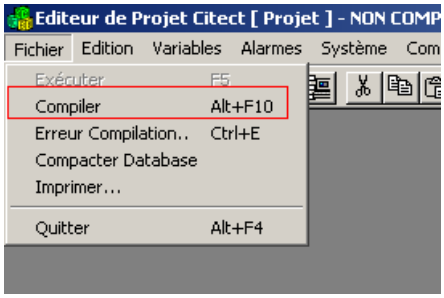


Sauvegarder la page.

2.4.8. Test de la page graphique :

Pour tester la page graphique afin de voir ce à quoi elle ressemble vraiment quand elle est lancée. Compiler tout d'abord le projet afin de voir s'il n'y a aucune erreur de configuration, et après cela lancer l'assistant de configuration de poste.

Aller à la fenêtre d'éditeur de projet Citect et cliquer sur fichier puis sur compiler.



Ou alors cliquer directement sur le bouton compiler  dans la barre d'outils.

Une barre de progression de compilation apparaît suivie d'un pop up de réussite de compilation.

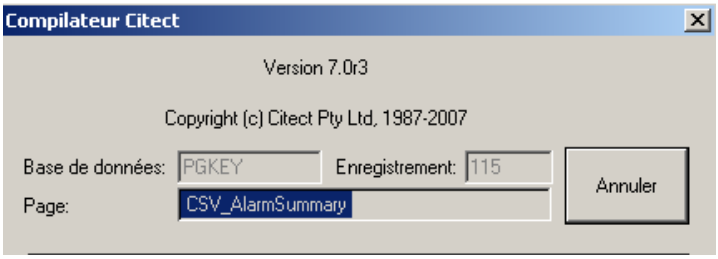


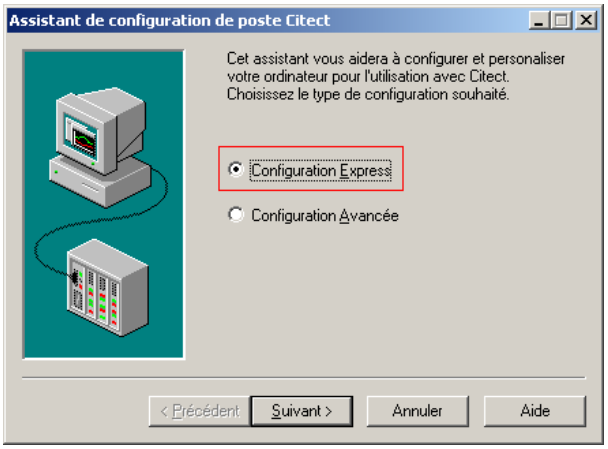
Figure III.10 : Compilation du projet sous Vijeo Citect

Si la compilation a échoué, double cliquer sur les erreurs de la liste pour aller directement à l'endroit de la mauvaise configuration.

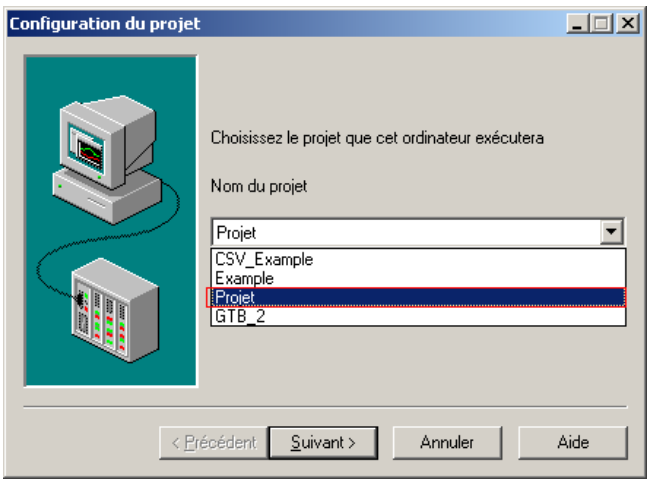
Configurer maintenant l'ordinateur à l'aide de l'assistant de configuration de poste.

Aller à l'explorateur Citect, sélectionner le projet en cours dans l'arborescence des projets, puis cliquer sur l'outil de configuration de poste dans la barre des menus.

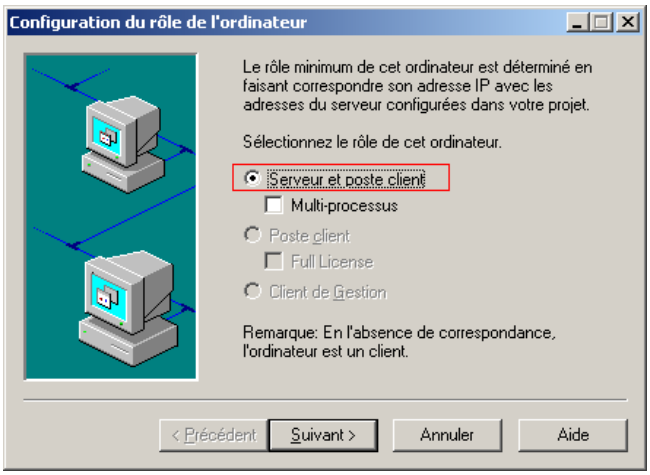
Sélectionner configuration express puis cliquer sur suivant.



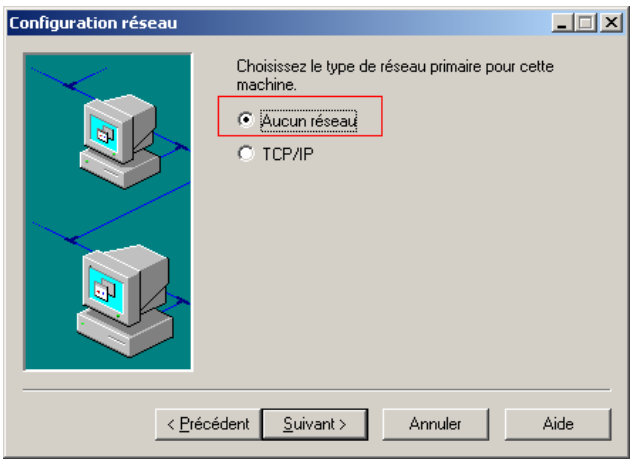
Au deuxième écran sélectionner le projet.



Cliquer sur serveur et poste client.



Cliquer sur Aucun réseau pour l'exemple ce qui lui permet de travailler seul sans automate connecté.

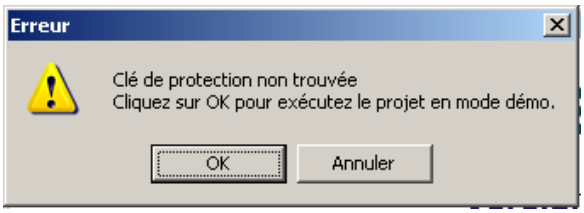


Cliquer maintenant sur Terminer pour terminer la configuration du poste.

Le projet est donc prêt à être lancé et à être testé.

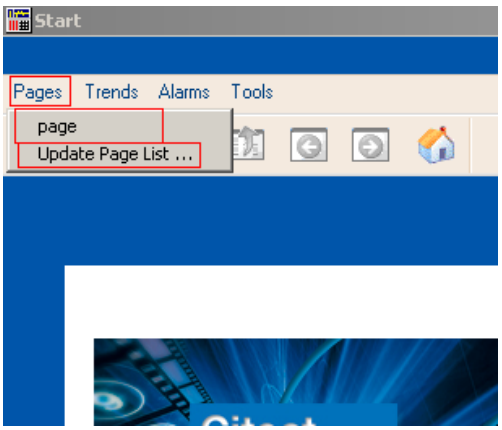
Cliquer sur le bouton « Exécuter »  . Le gestionnaire d'exécution CitectSCADA apparaît montrant le statut de progression de l'exécution.

S'il n'y a pas la clé de Vijeo Citect, le pop up suivant apparaît.



Cliquer sur OK pour lancer le projet en mode Demo.

Cliquer sur Page puis cliquer sur «Update Page List ». Les pages disponibles apparaitront donc. Sélectionner « page ». La page devrait apparaître.



Cliquer sur le bouton « Manuel » puis sur le bouton « Auto » et vérifier que la lampe appropriée change de couleur et devient verte et vérifier aussi qu'en mode automatique le bouton ON/OFF ne fonctionne pas. Laisser le curseur de la souris sur le bouton ON/OFF pendant quelques secondes pour voir la note d'aide apparaître.

On peut tester le bouton ON/OFF avec le mode manuel pour voir que la lampe change selon d'état.

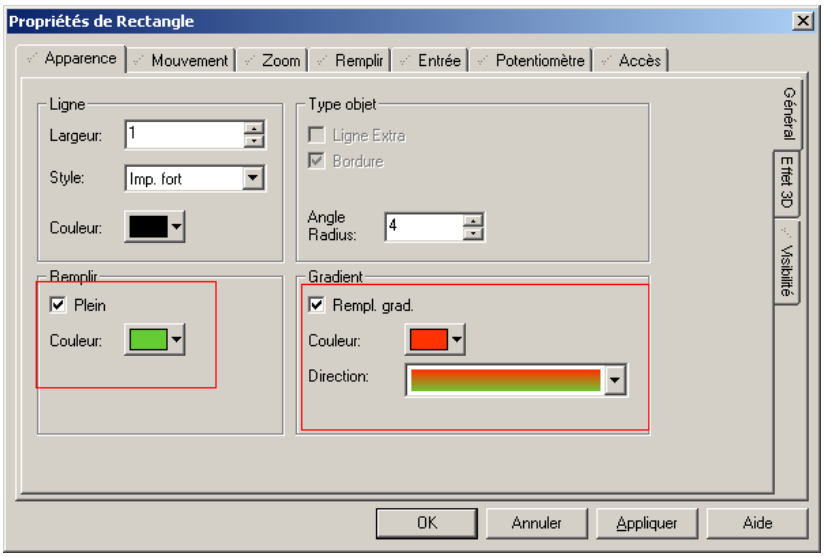
Cliquer sur ALT+ESPACE pour afficher le menu qui permet de changer de fenêtre ou d'arrêter la simulation.

Revenir maintenant à l'éditeur graphique.

2.4.9. Indicateurs analogiques et contrôles :

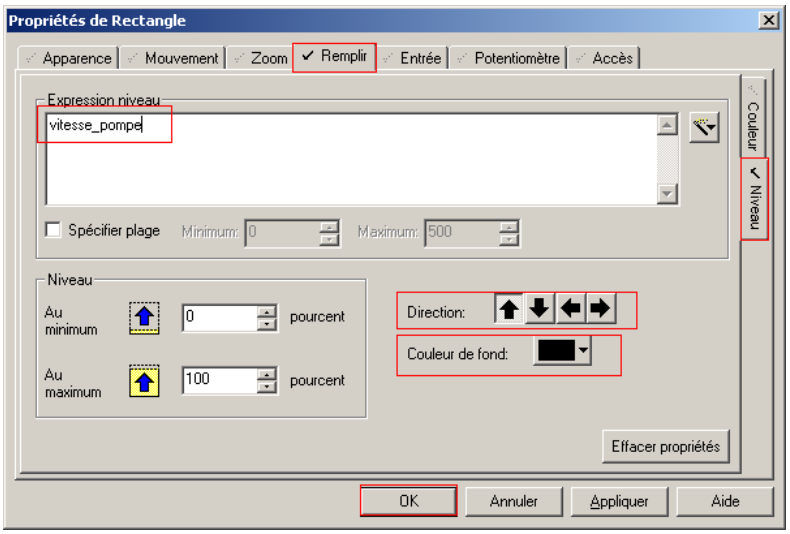
Créer un rectangle vert comme vu plus haut, cette fois-ci à coté du premier et plus fin, ou alors copier le premier et le modifier en plus fin et modifier sa couleur.

Double cliquer sur le rectangle pour avoir accès à ses propriétés et choisir remplissage graduel, et mettre la deuxième couleur, rouge par exemple, et modifier la direction de changement à la verticale.



Une fois la configuration de l'apparence du rectangle finie, se rendre à l'onglet Remplir, dans cet onglet aller dans l'onglet Niveau. Là mettre la variable analogique pour l'exemple

mettre « vitesse_pompe ». Changer la couleur de fond et mettre noir et confirmer en cliquant sur OK.

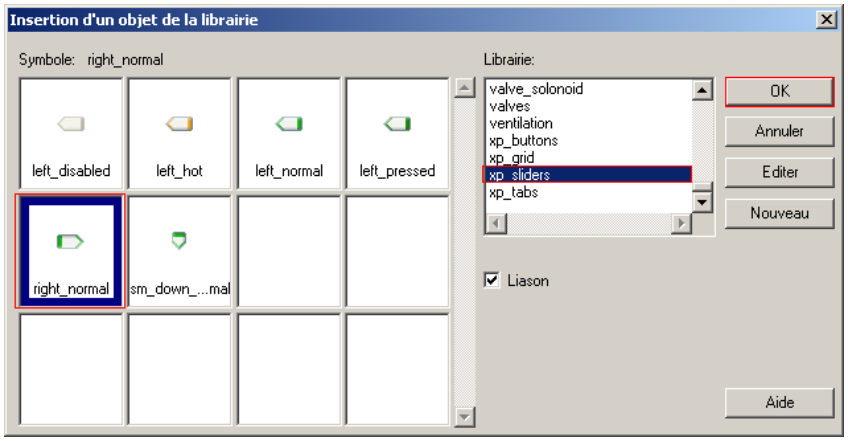


Ce rectangle va se remplir en fonction de la vitesse de la pompe en pourcentage.

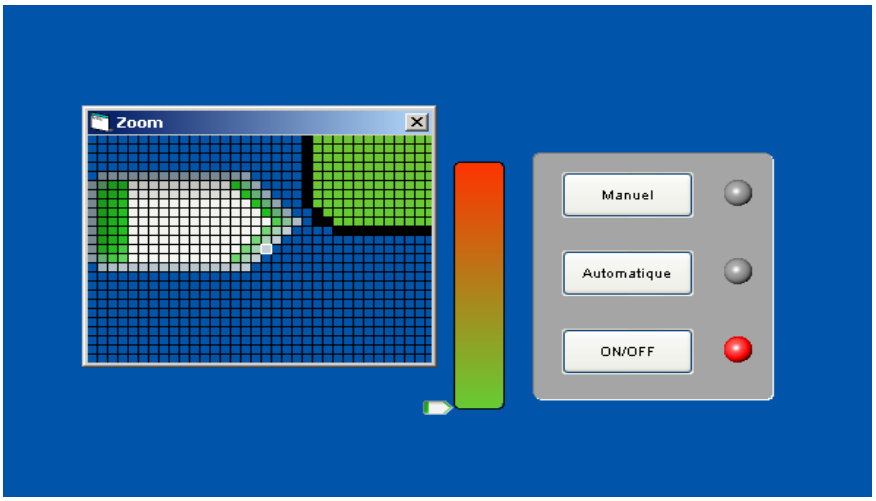
Pour créer un curseur qui va permettre à l'opérateur de modifier la vitesse de la pompe quand celle-ci est en mode manuel.

Cliquer sur Objet de la librairie dans la barre d'outils pour sélectionner un objet déjà existant dans la librairie.

Aller dans la librairie « XP_sliders » et sélectionner le symbole « right normal ».

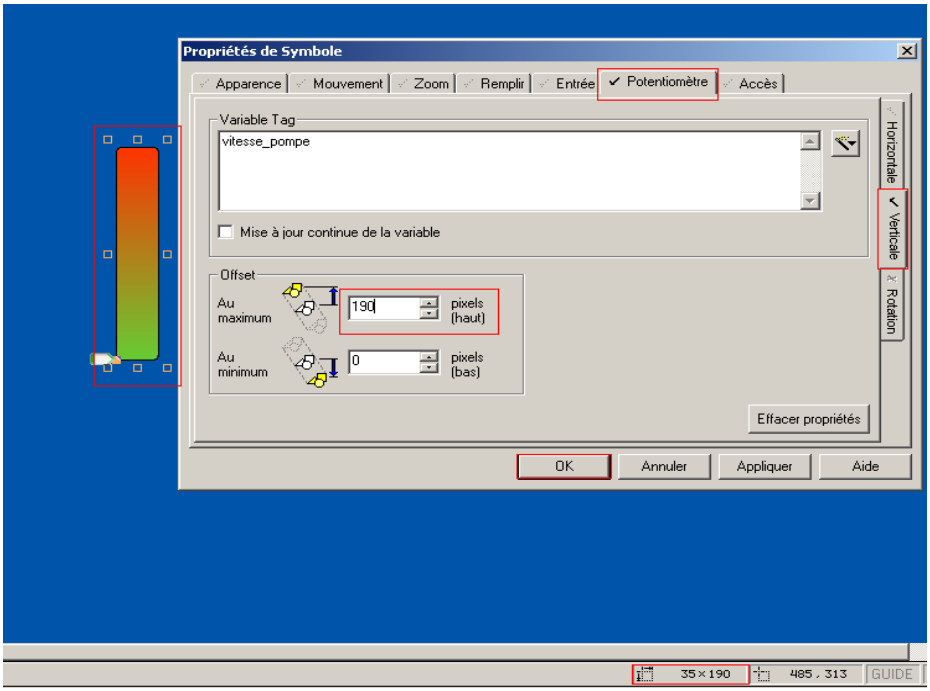


Positionner le curseur dans le coin gauche du bas du rectangle créé plus tôt et aligner le milieu du curseur avec le bas du rectangle en utilisant le zoom qu'on peut activer en cliquant sur F10 qui permet de travailler avec une grande précision.



Double cliquer sur le curseur pour le configurer. Et aller dans l'onglet « Potentiomètre » et dans cet onglet aller dans l'onglet « Verticale » et mettre comme variable « vitesse_pompe ».

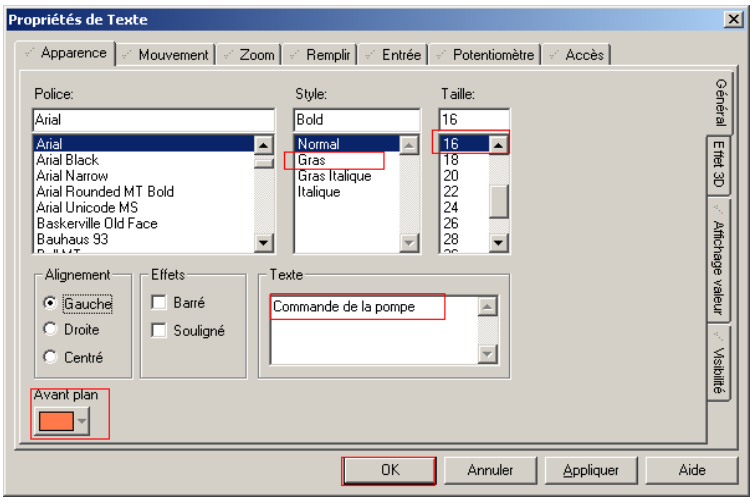
Maintenant changer la hauteur maximale dans Offset et mettre la hauteur du rectangle qui est visible dans la barre d'état qui donne la hauteur et la largeur en pixels de l'objet sélectionné.



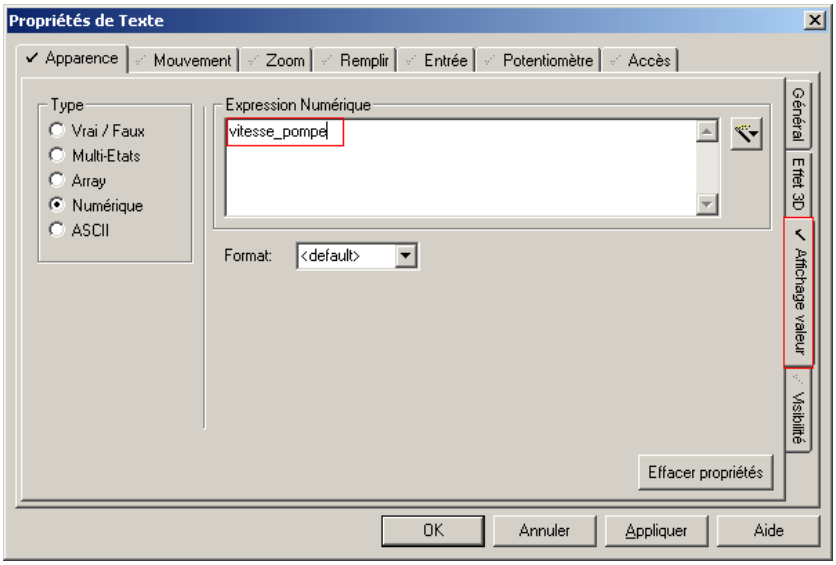
Cliquer sur OK pour confirmer.

2.4.10. Configuration des textes et des nombres :

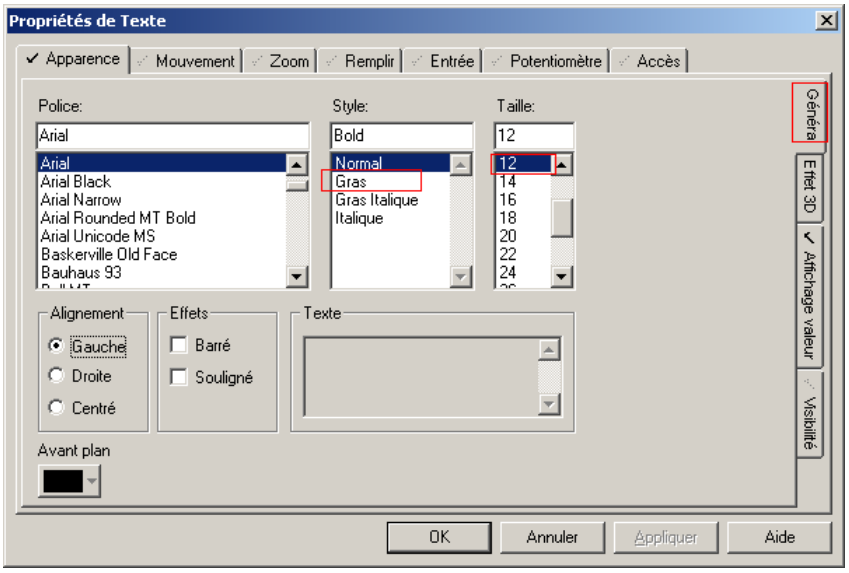
Cliquer sur A dans la barre d'outils, commencer à écrire, par exemple « commande de la pompe » puis cliquer là où one veut placer le texte, un pop up apparaît pour configurer le texte. Mettre par exemple l'ecriture en gras, la taille de la police à 16 et changer de couleur. Il sera toujours possible de double cliquer sur le texte pour le modifier ou changer de style.



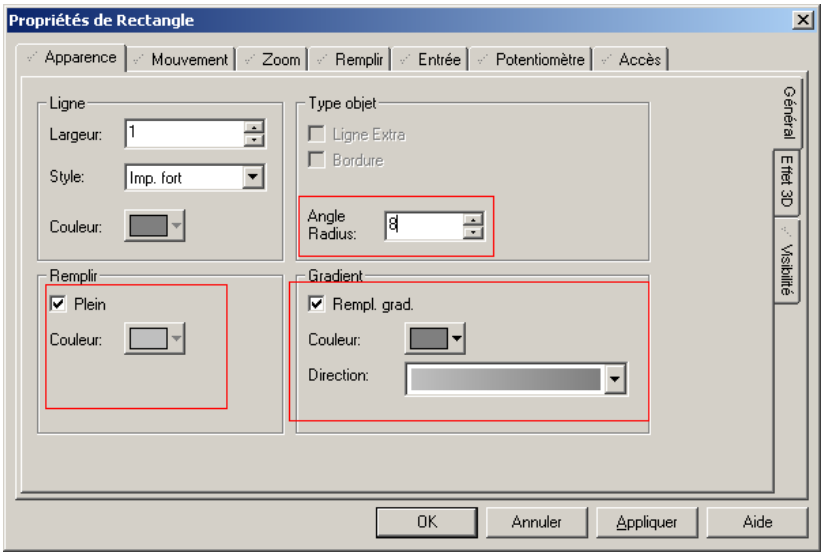
Cliquer sur l'outil nombre dans la barre d'outils ## et placer les nombres en dessous du rectangle graduel. Un pop up de configuration apparaît, mettre la variable « vitesse_pompe » dans le champ Expression Numérique.



Cliquer sur l'onglet Général et changer la police à 12 et en Gras.



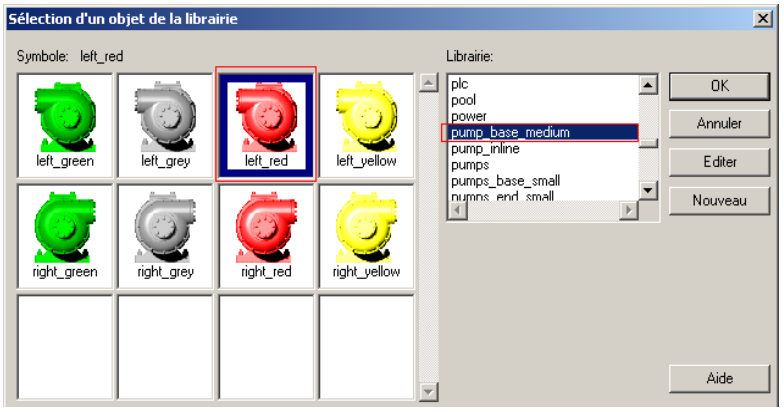
Par soucis d'esthétique il est possible de dessiner un rectangle autour de tous les éléments dessinés précédemment. Mettre le plein avec remplissage graduel avec deux nuances de gris par exemple. Mettre l'Angle Radius à 8 pour avoir des bords arrondis et mettre un effet 3D et sélectionner Rabaissé. Reculer le rectangle en arrière plan afin de pouvoir voir les autres éléments.

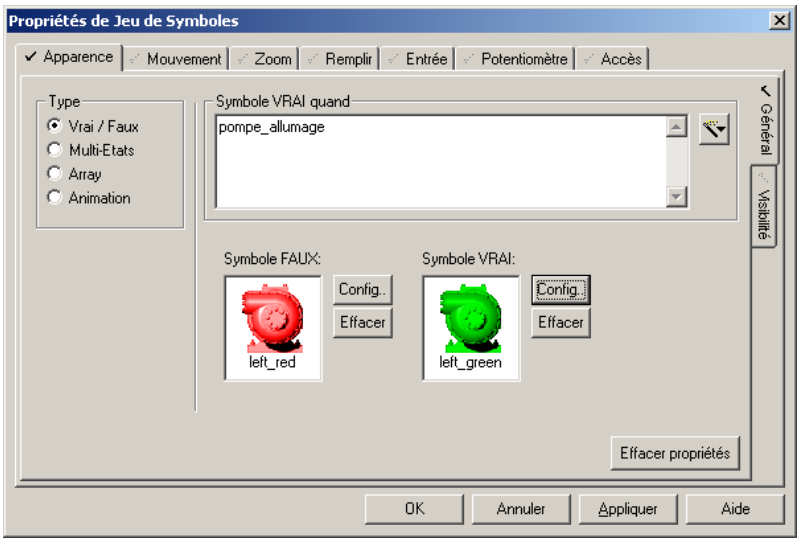


2.4.11. Pompes et tuyaux :

Pour ajouter les symboles de la pompe et des tuyaux.

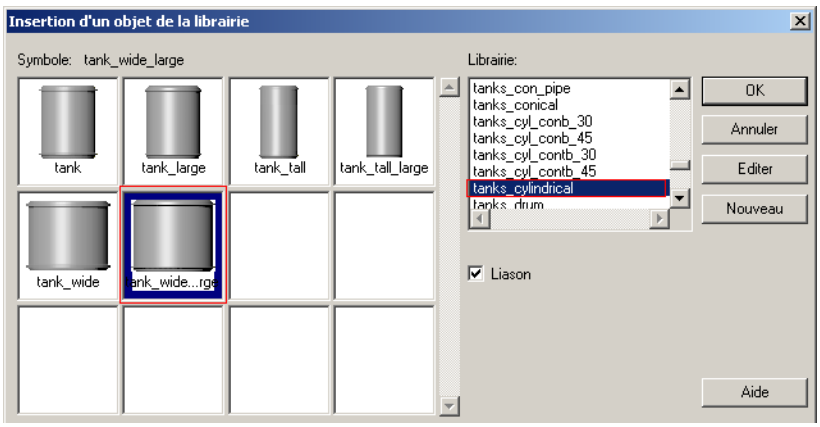
Cliquer sur le bouton « Jeu de symboles »  , cliquer sur la config du symbole FAUX et sélectionner une pompe rouge qui est dans la librairie « pump_base_medium » et choisir la pompe « left_red », et choisir la pompe verte pour le symbole VRAI. Et dans la variable du symbole choisir « pompe_allumage » et cliquer sur ok pour confirmer.



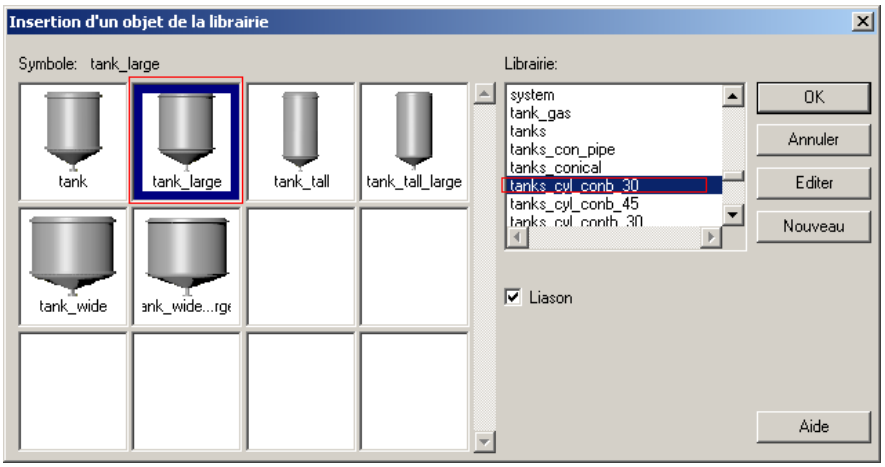


Ajouter maintenant une citerne d'arrivée et une citerne de destination et relier le tout avec des tuyaux.

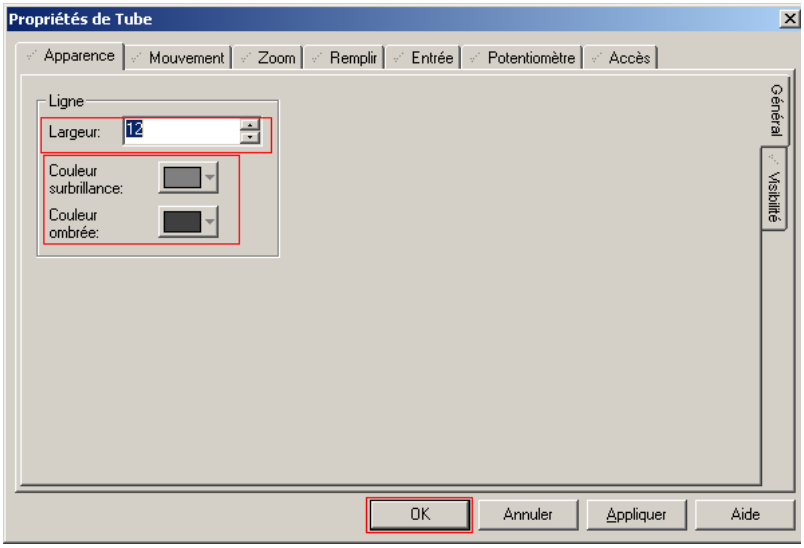
Cliquer sur « Objet de librairie »  pour ajouter des symboles statiques. Aller à la librairie « tank_cylindrical » et sélectionner la « tank_wide_large ».



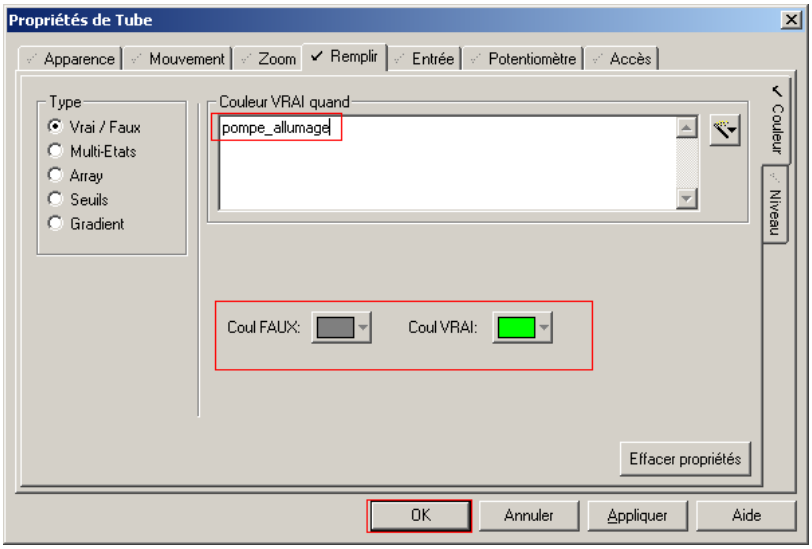
Ajouter une deuxième citerne cette fois dans la librairie « tank_cyl_conb30 » et prendre la « tank_large » positionnée à droite en haut de la pompe.



Cliquer sur « Tube »  dans la barre d'outils qui permet de dessiner des tuyaux avec un effet 3D auxquels il est possible de changer de couleur. Dessiner deux tuyaux distincts l'un allant du réservoir source vers la pompe et l'autre de la pompe vers le réservoir de sortie. Pour dessiner avec l'outil Tube maintenir la touche control appuyée pour dessiner seulement horizontalement ou verticalement pour éviter de dessiner des lignes obliques. Après relâchement la fenêtre de configuration de tube apparaît choisir sa largeur ainsi que ses couleurs et cliquer sur ok pour valider.



Il est possible d'ajuster le tuyau en cliquant dessus ou alors dans les angles il est possible de cliquer dessus pour changer d'angle. Double cliquer sur les tuyaux pour modifier dans l'onglet « Remplir » la couleur lorsque la pompe est en marche et lorsqu'elle est à l'arrêt, mettre vert et gris et mettre Couleur VRAI quand « pompe_allumage ».



La configuration graphique est finie pour cette partie d'apprentissage, il est possible de tester à nouveau cette page en compilant et en exécutant le projet.

Nous allons maintenant parler des alarmes et des tendances.

2.3. Les alarmes:

Le système d'alarme Vijeo Citect surveille les processus et alerte les opérateurs en cas d'événements critiques ou imprévus nécessitant une intervention. Il y a plusieurs types d'alarmes.

Les alarmes digitales, elles s'activent en fonction de l'état d'une ou de deux variables numériques. L'alarme devient active lorsque l'état de la condition de déclenchement reste vrai pendant un délai de temps défini lors de la configuration.

Les alarmes analogiques, elles sont déclenchées lorsqu'une variable analogique dépasse une ou plusieurs valeurs spécifiques. Chaque alarme peut être une combinaison des types suivants :

- Alarmes haut ou très haut, lorsque la variable dépasse une limite supérieure.
- Alarmes basse ou très basse, lorsque la variable descend en dessous d'une limite inférieure.
- Alarme de déviation, lorsque la variable dévie d'une valeur prédéfinie.
- Alarme de fréquence, lorsqu'un important changement de valeur intervient dans un intervalle de temps donné.

Les alarmes horodatées, qui sont semblables aux alarmes digitales mais qui utilisent un Timer pour dater les conditions de déclenchement au lieu de l'heure de déclenchement.

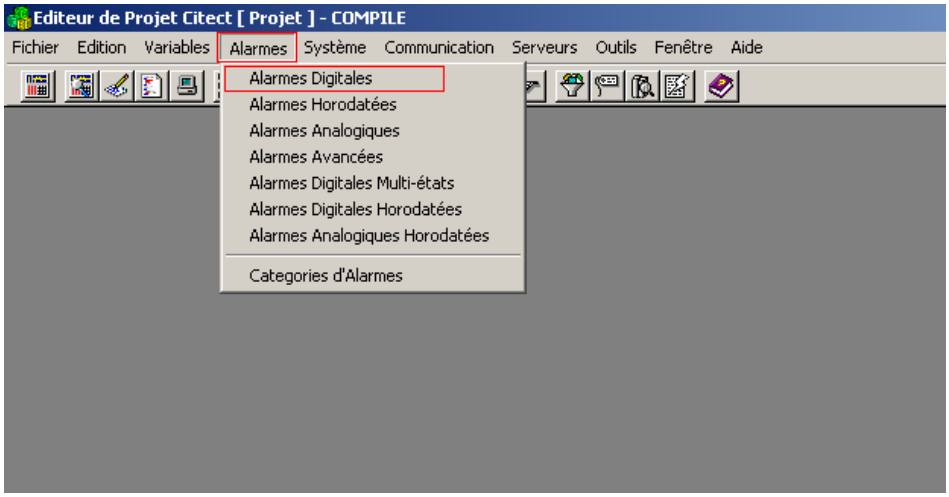
Les alarmes avancées, qui sont activées lorsqu'une expression Cicode change.

Les alarmes digitales multi-états, elles testent la sortie de trois variables numériques (par exemple : tags A, B, et C) pour définir huit états. Les états représentent toutes les combinaisons possibles de valeurs vrai/faux que les variables peuvent avoir.

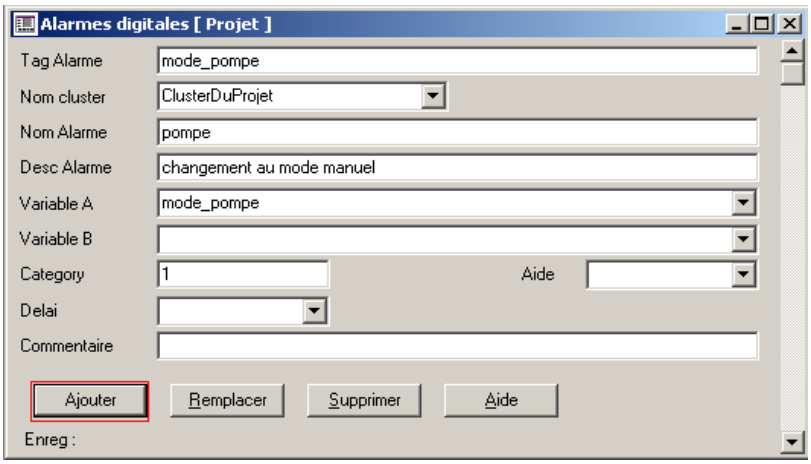
Les alarmes analogiques et numériques horodatées, n'interrogent pas des variables pour déterminer des conditions d'alarme. Le serveur d'alarme est informé de tout changement de valeur d'une variable spécifiée grâce à la fonction Cicode [Alarm]NotifyVarChange.

2.3.1. Configuration d'une alarme :

Pour configurer une alarme sur le mode de la pompe « mode_pompe » pour qu'elle envoie une alerte lorsque la pompe sera mise en mode manuel. Aller dans l'Editeur de Projet Citect, dans le menu cliquer sur Alarmes puis sur Alarmes Digitales.



Remplir le formulaire pour une seule alarme comme sur la figure ci-dessous. Avec le tag de l'alarme, le nom du cluster, le nom de l'alarme, sa description et la variable à surveiller, dans ce cas « mode_pompe », puis cliquer sur Ajouter.



Il aurait été possible de créer une alarme qui s'active lors du passage au mode manuel en mettant dans le champ de la variable « mode_pompe=0 » pour inverser le sens de detection.

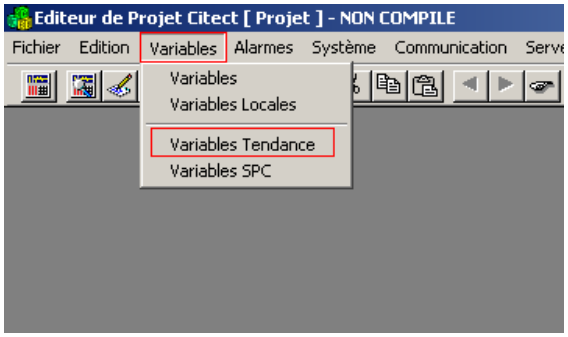
Fermer la fenêtre des alarmes digitales.

2.4. Les tendances :

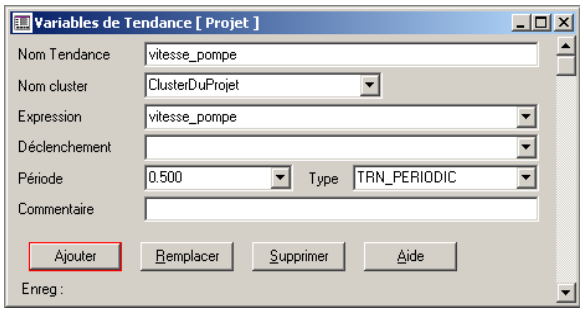
Les tendances sont des représentations graphique de l'évolution des valeurs d'une ou de plusieurs variables (ou expressions) au sein d'une installation.

2.4.1. Configuration de tendances :

Pour configurer une variable tendance afin d'enregistrer la tendance de la vitesse de la pompe, aller dans l'Editeur de Projet Citect, dans le menu cliquer sur Variables puis sur Variable Tendance.

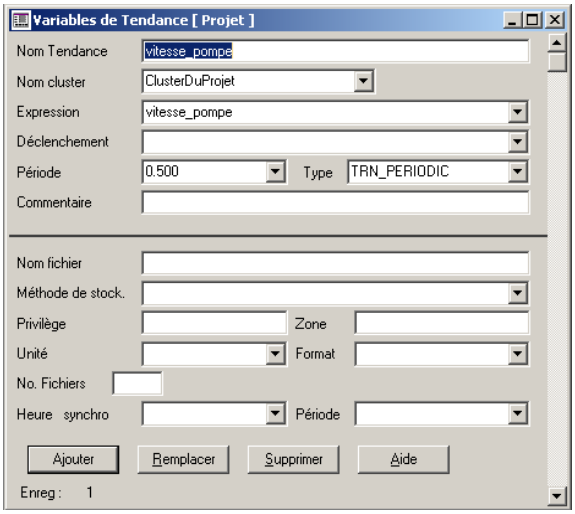


Remplir le formulaire avec le nom de la variable de tendance, le cluster, l'expression de la variable à surveiller, dans ce cas mettre la variable « vitesse_pompe », la période d'échantillonnage, prendre par exemple 0.5 secondes, ainsi qu'un commentaire, puis cliquer sur ajouter pour valider.



Cliquer sur ajouter et fermer ce formulaire.

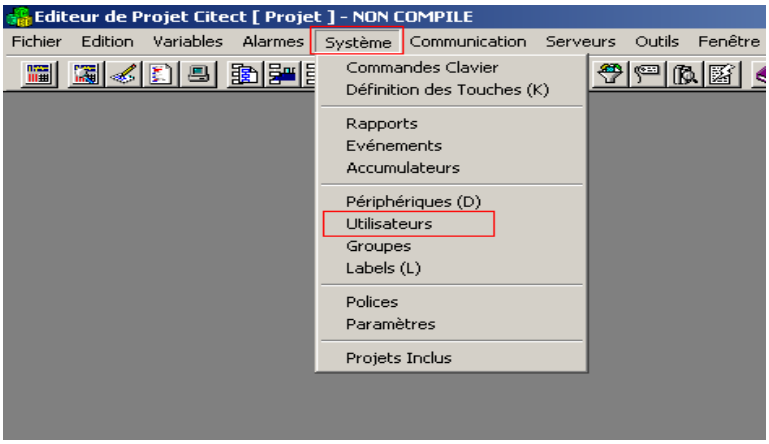
Il est aussi possible cliquer sur F2 pour avoir plus d'options sur le stockage de la donnée, par exemple en choisissant le fichier où les données seront sauvegardées, en choisissant la méthode de stockage des données, en entier court ou en virgule flottante, en choisissant les privilèges requis à un opérateur pour accéder à cette tendance, l'unité de travail, le format, le nombre de fichiers d'historique enregistrés sur le disque dur (pour la variable concernée), l'heure de synchronisation du début du fichier d'historique, ainsi que la période de début du fichier d'historique.



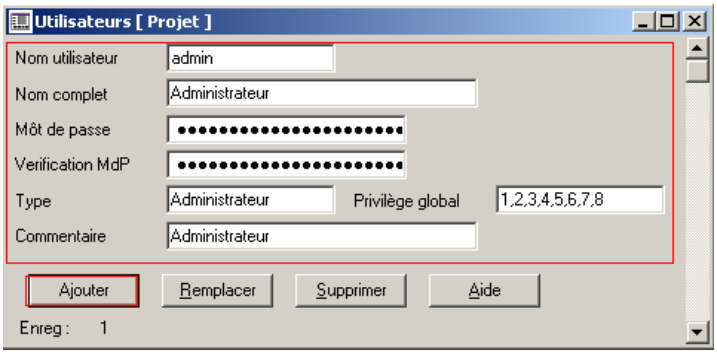
2.5. Configuration de la sécurité :

Dans la plupart des projets d'interface homme-machine, on trouve un système d'authentification qui permet de créer des classes d'utilisateurs aux privilèges différents, pour ne pas permettre à des personnes n'ayant pas le droit de modifier des paramètres qui peuvent s'avérer dangereux s'ils sont changés sans précautions.

Dans le menu de l'Editeur de Projet Citect, cliquer sur système puis sur utilisateurs.



Un formulaire apparaît, le remplir avec le nom d'utilisateur, le nom complet, le mot de passe pour accéder à ce compte, le type de compte ainsi que le privilège global de ce compte. Remplir le formulaire comme suit.



Le champ pour le mot de passe contient des points mais il faut taper le mot de passe réel, pour l'exemple prendre « citect », ce système est fait pour éviter qu'une personne passant par là connaisse le mot de passe si elle n'en a pas le droit.

Cet utilisateur a été configuré pour le maximum de privilèges de sécurité à travers le projet.

2.5. Exécution du projet :

Il est temps d'exécuter le projet afin de le tester et de voir ce que donne réellement notre projet.

Compiler le projet et l'exécuter.

Cliquer sur le bouton « auto » puis sur le bouton « manuel », appuyer maintenant sur le bouton ON/OFF, on remarque que la pompe ainsi que les tuyaux changent de couleur selon l'état de la pompe.

On voit aussi qu'une alarme est générée et affichée dans la barre d'alarmes en bas de la page quand on passe au mode manuel. L'icône d'alarme en bas à droite clignotera aussi pour indiquer qu'une nouvelle alarme est arrivée. Cliquer sur cette icône d'alarme pour afficher la page des alarmes.

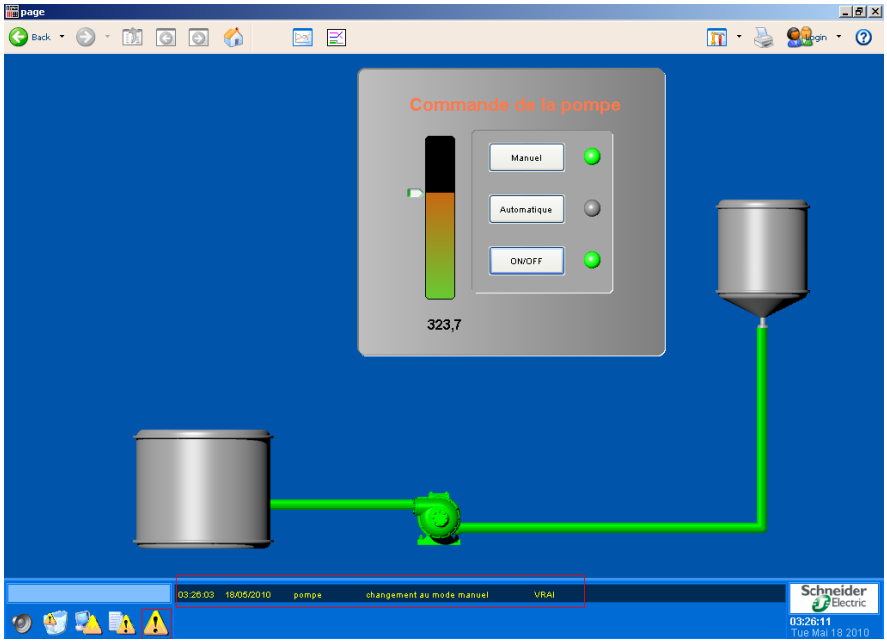
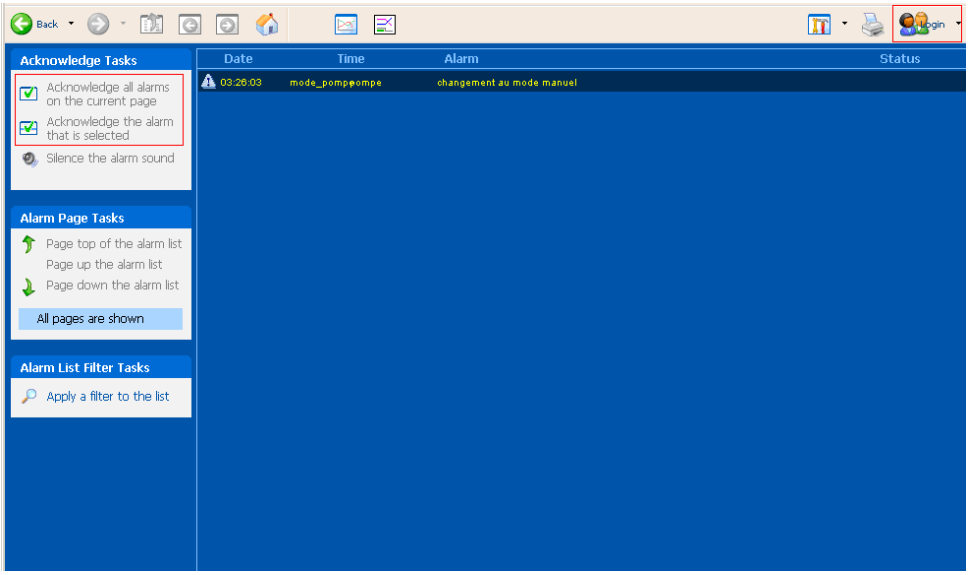


Figure III.11 : Execution du projet sous Vijeo Citect

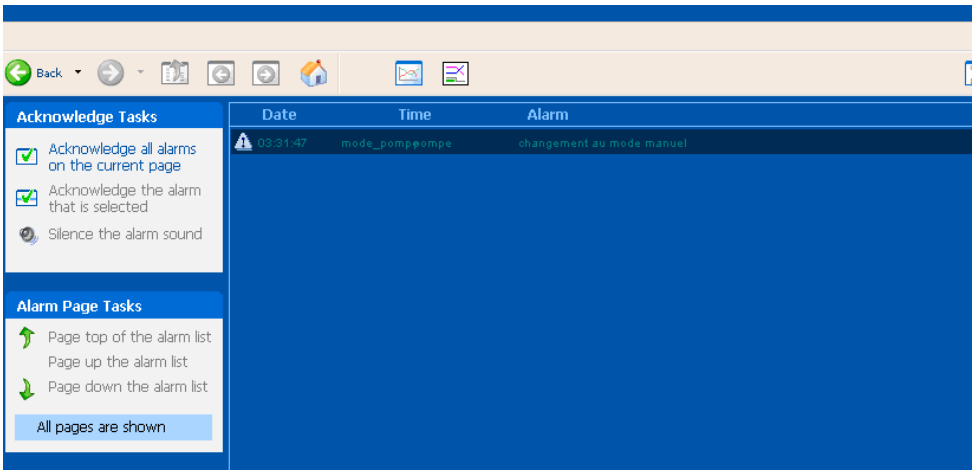
En haut à droite on peut accuser réception de l'alarme, en cliquant sur acknowledge alarm, mais si on essaye on voit qu'on n'a pas accès à cette fonction. Par défaut un privilège de niveau 1 est requis pour avoir accès à cette fonction.



Cliquer donc sur le bouton login pour se connecter. Taper le nom d'utilisateur et mot de passe définis dans la partie précédente.

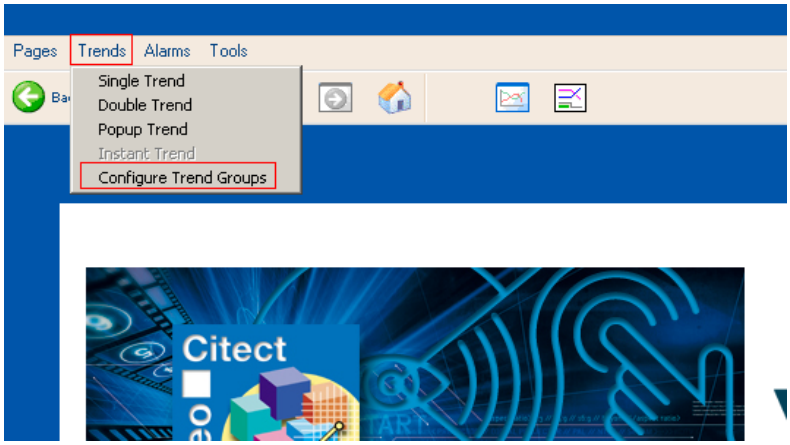
Une fois connecté avec les droits d'accès à tous les privilèges, il est possible d'accuser réception des alarmes. On notera que l'alarme change de couleur et qu'elle disparaît si on met la pompe ne mode automatique, alors qu'elle n'aurait pas disparu si on n'avait pas accusé

réception, ainsi l'opérateur ne peut pas rater une alarme même si elle s'est produite lorsqu'il était absent.

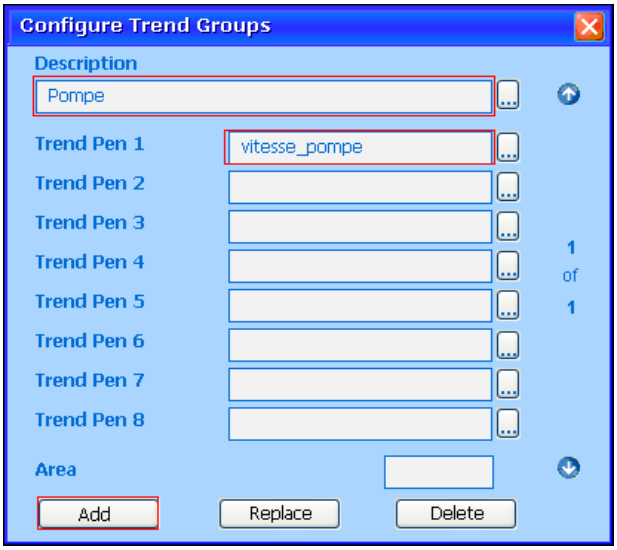


2.5.1. Les tendances :

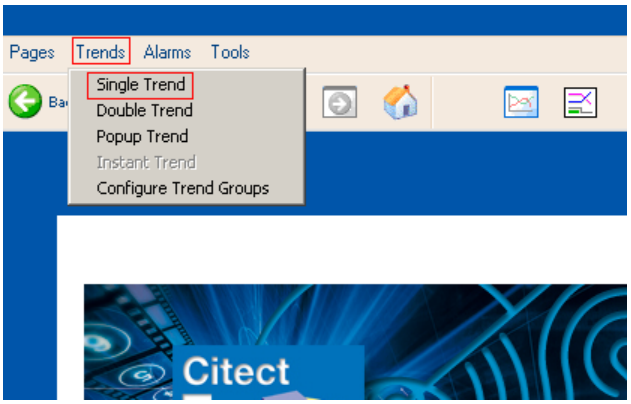
Dans le menu de navigation cliquer sur Trends puis sur Configure Trend Groups.



Donner une description puis donner les variables à tendance disponibles, dans cet exemple il n'y a que la vitesse de la pompe, puis cliquer sur Add et enfin fermer la fenêtre de configuration des groupes de tendances.



Sélectionner maintenant, Single Trend du menu Trends.



Utiliser le bouton de sélection de groupes de tendance  pour sélectionner le groupe pompe déjà créé.

En modifiant la valeur de la vitesse de la pompe à l'aide du curseur créé dans la page graphique ou en la modifiant directement à partir de l'automate on remarque le changement dans la courbe de tendances.

Il est possible de savoir à quel moment s'est produit un événement en cliquant sur la courbe à tout moment et accéder à des données de l'historique facilement en utilisant l'outil « Display history mode » qui permet de se déplacer dans la courbe, il est possible de zoomer et d'enregistrer à tout moment l'historique de la courbe avec le bouton « Export to file ».

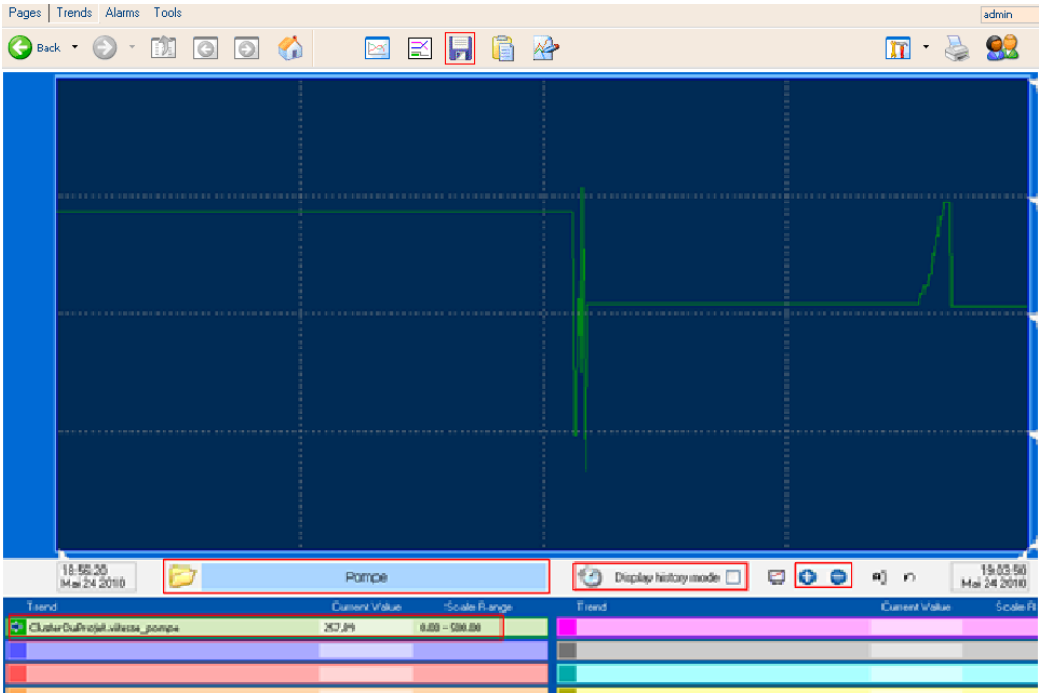


Figure III.12 : Courbe de tendances sous Vijeo Citect

2.6. Sauvegarde du projet et restauration de projets :

Pour sauvegarder, aller dans l'Explorateur Citect, sélectionner le projet et le sauvegarder à l'aide du bouton Sauvegarde  et saisir le chemin du dossier où sera placé le dossier, un fichier au format « CTZ » est alors créé, c'est un moyen sûr de sauvegarder ses projets.

Pour restaurer, cliquer sur la touche « Restauration »  et indiquer le chemin du fichier « CTZ », on voit alors apparaître le projet dans la liste des projets.

Conclusion :

Nous avons pu voir à travers ce chapitre que Vijeo Citect est un logiciel performant qui permet la supervision des processus ainsi que l'exploitation des données acquises à partir de ces derniers.

Aussi, il nous offre la possibilité d'intégrer nos propres programmes dans le but de piloter les processus.

CHAPITRE IV

APPLICATION

Introduction :

Le siège Schneider Electric gère plusieurs tâches à la fois, intégrant des systèmes d'automatisme de pointe, il assure une économie d'énergie de l'ordre de 60% d'une part, et d'autre part la sécurité des employés, leur confort tout en facilitant la gestion technique du bâtiment

Afin de réaliser une gestion technique globale de ce siège on a dû passer par 3 étapes principales:

- La première étape consiste à définir des cahiers de charge qui vont rendre les taches réalisées par ce bâtiment plus autonomes.

- Dans la seconde étape on a programmé et réalisé sous Unity Pro plusieurs blocs DFB implémentables dans les Automates Programmables, dans notre cas l'automate M340.

- Créer des IHM à l'aide du logiciel Vijeo Citect facilitant la supervision de ces différentes tâches.

Pour ce faire, on a traité les 5 principales parties de la gestion technique du bâtiment qui sont :

- La gestion d'éclairage.
- La gestion d'électricité.
- La gestion de climatisation.
- La gestion du parking.
- Le contrôle d'accès.

Dans la partie supervision, on a essayé de se rapprocher au maximum a des image réelles, pour cela, on s'est basé d'une part sur des photos prise du siège Schneider Electric et on les a ensuite importées dans Vijeo Citect, et d'autre part on a créer des vues en 3D à l'aide du logiciel Google Sketchup7 pour avoir une interface complète et facile a utiliser.

1. Le Cahier de charge :

Il se compose de 11 étages, dont 10 comportent des espaces de travail, des bureaux, et salle de réunion, en plus de ça, il y'a un étage technique qui se trouve au rez-de-chaussée qui gère l'électricité, le parking et les systèmes de ventilation en assurant la supervision de tous les secteurs du bâtiment.

On a deux types d'étage :

- Des étages qui contiennent 5 salles dont 2 sont des salles de réunion et les autres sont des bureaux de travail.
- Des étages qui contiennent un seul espace de travail.

En plus de ça, le siège Schneider Electric contient un parking qui se compose de deux parties séparées par un portail.

Dans notre application on a prévu 11 Automate de la famille M340, chaque automate va gérer l'éclairage et le contrôle d'accès dans les différentes salles se trouvant dans chaque étage.

On a aussi prévu un Sepam S80 ainsi que des PM710 et des variateurs de vitesse ATV71 et différents types de capteurs (lumière, de présence, des fin de course, …) ainsi que des disjoncteurs communicants en Modbus.

On a un automate qui se situe dans l'étage technique assurant le contrôle des ventilateurs, la gestion d'électricité et la gestion du parking.

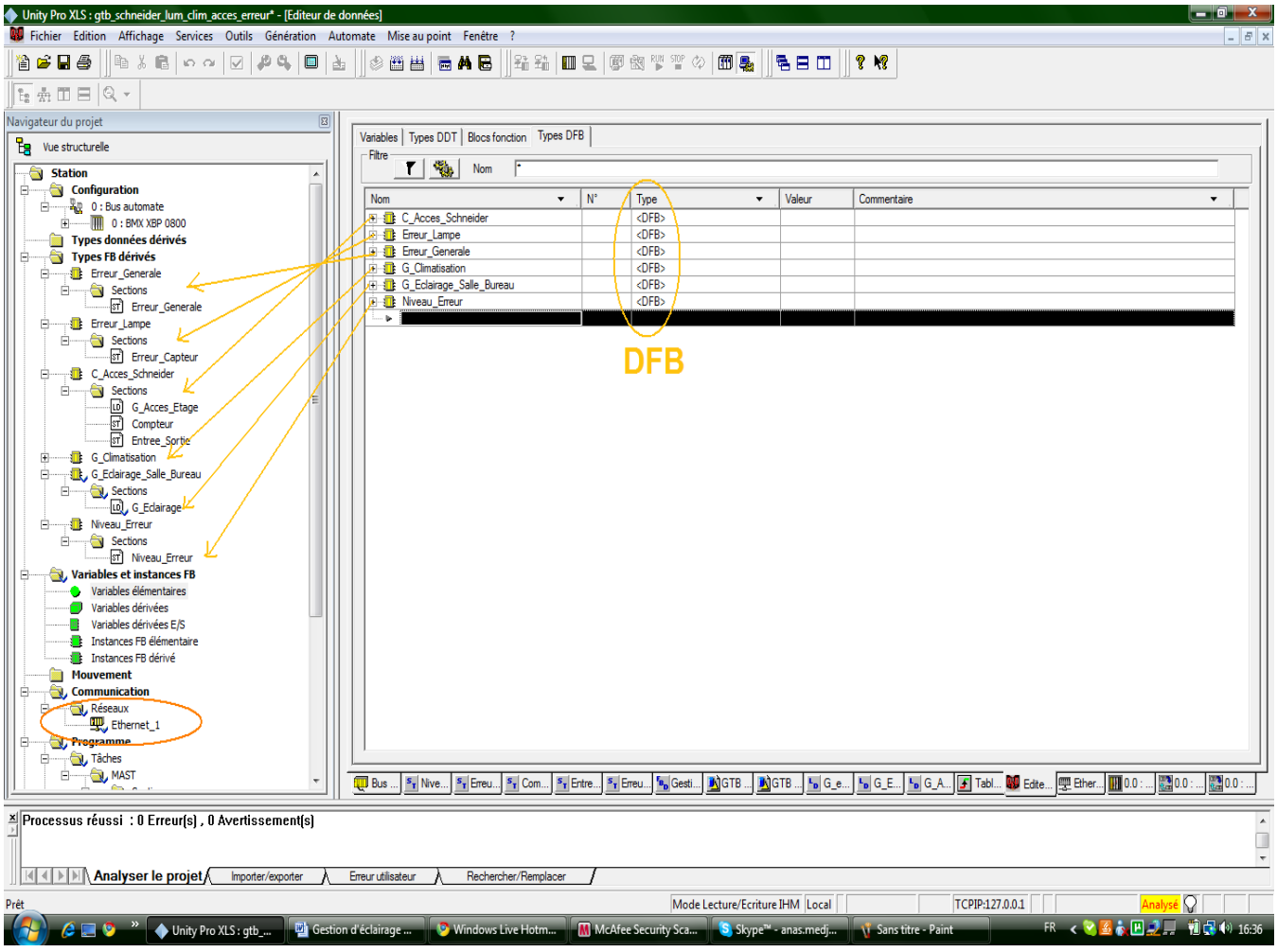


Figure IV.1: Les différents blocs Utilisés dans L'application_1

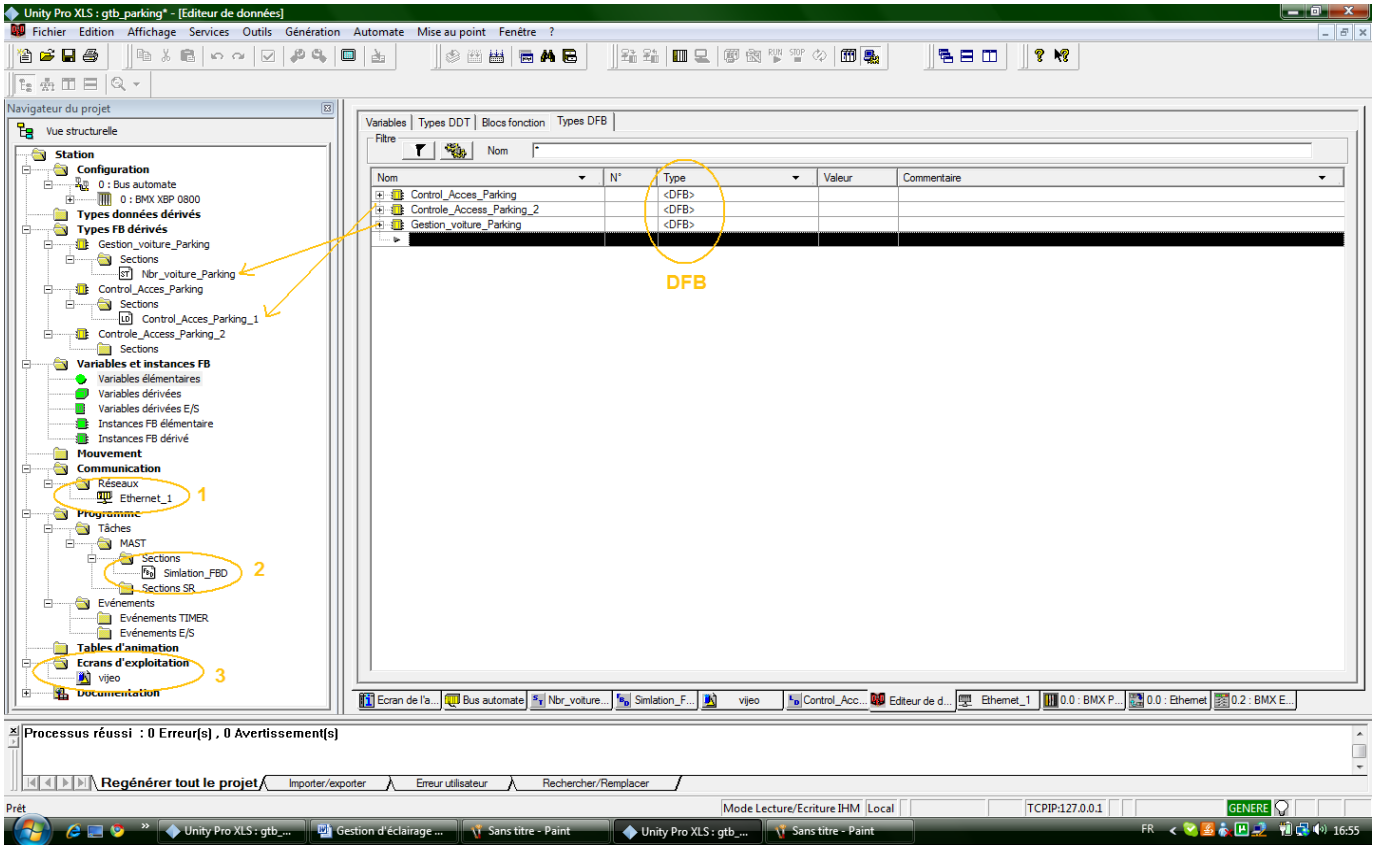


Figure IV.2: Les différents blocs Utilisés dans L'application 2

1. Réseau.
2. Simulation avec une tâche Mast.
3. Simulation avec Ecran d'exploitation d'Unity Pro.

1.1. Configuration de l'automate :

1.1.1. Communication de l'automate :

L'automate contient deux types de ports de communication, un port Modbus assurant la communication avec les PM710, les variateurs de vitesse ATV71, les Sepam S80, et avec tout les matériels communicants en Modbus, et un port Ethernet assurant la communication en réseau et le transfert des données de la couche commandée vers la couche de supervision.

La figure suivante résume les différentes communications et les échanges de données réalisées par l'automate.

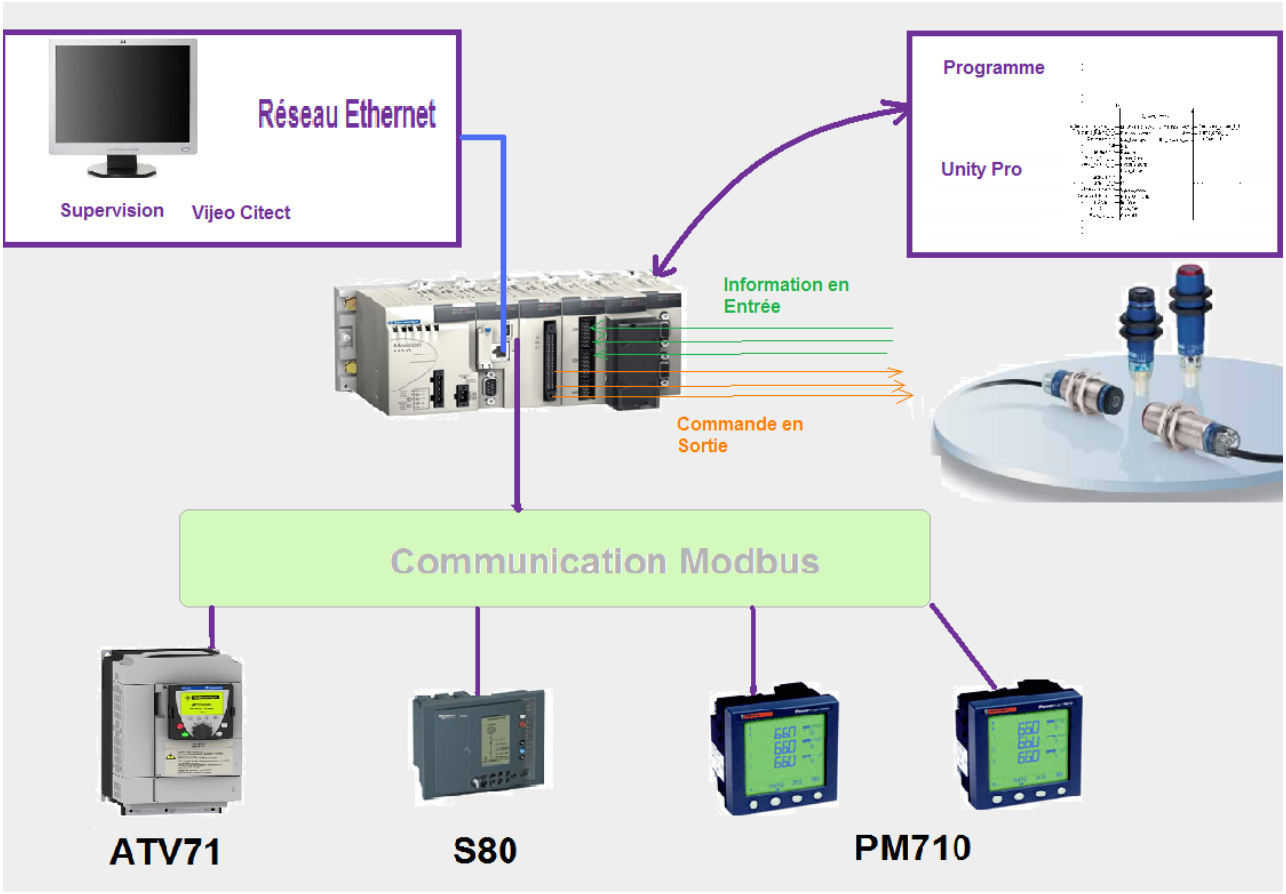


Figure IV.3 : Communication de l'automate M340

1.1.2. Configuration de l'automate sous Unity Pro:

L'automate sera configuré de sorte à assurer le transfert d'informations des capteurs d'une part et des matériels communicants, et il doit aussi communiquer en réseau, et cela nécessite une configuration sous Unity Pro.

La figure suivante résume cette configuration :

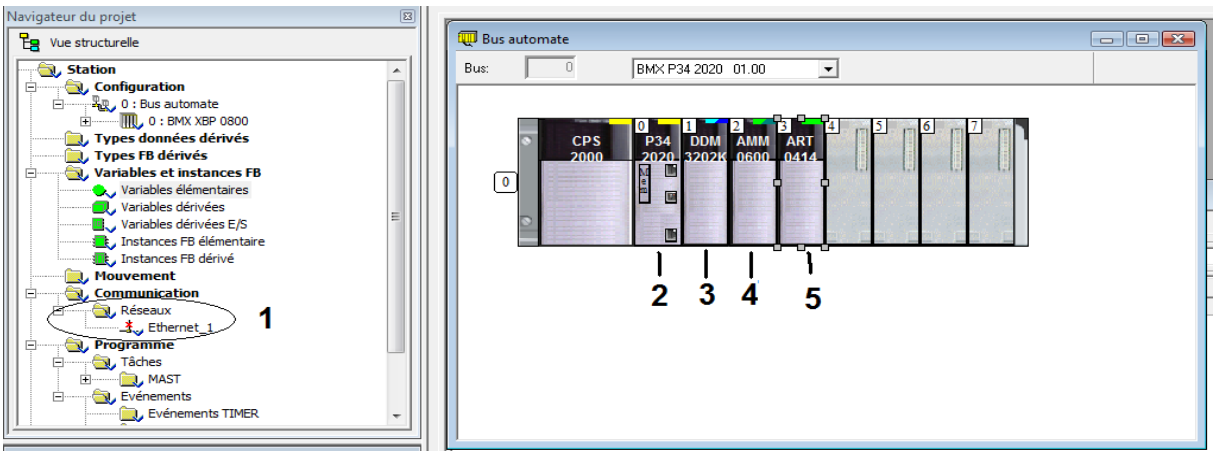


Figure IV.4 : Configuration de l'automate sous Unity Pro

1→Création d'un Réseau Logique
2→Configuration du module de communication (Modbus ,Ethernet dans notre application)
3→Les Entrées de Type TOR (Tout ou rien)
4→Les Entrées analogique pour courant et les tensions…
5→Les Entrées De Type ISOL pour les Température

2. La gestion d'éclairage :

L'éclairage est le deuxième poste de consommation après le chauffage et la climatisation, il en représente 27%, dans bâtiment Schneider Electric sa consommation moyenne est de 58% dans les bureaux, 24% pour les espaces de circulation, 14% pour les communs et 4% pour les sanitaires.

Dans notre application on a programmé des scénarios et on a créé des blocs DFB pour les 3 principaux types de salle qui sont :

- Les bureaux
- Les salles de réunion
- Les salles de conférence
- Les Espace de travail

Pour cela on s'est basé sur un cahier de charge général d'éclairage, qu'on a programmé a l'aide du logiciel Unity Pro ensuite on a essayé de rendre ce bloc compatible avec tous les types de salle en introduisant dans chaque salle une entrée bien spécifique pour faire comprendre au bloc qu'on est dans la salle de type bureau et non dans une salle de type espace de travail.

A la fin on a créé des IHM pour la supervision pour visualiser tous les détails concernant l'éclairage dans les différents types de salle.

Remarque :

Pour la salle de conférence on n'a pas fait la supervision, on a programmé le bloc d'une manière à éteindre la lumière, allumer le data-Show et fermé le Rideau si on appuie sur le bouton conférence

2.1. Cahier de charge générale:

Au cours de la journée :

On a un capteur de présence qu'il va détecter la présence ou pas d'un individu:

- Si oui, le capteur de lumière va détecter à son tour la présence ou pas de la lumière du jour, on distingue deux cas :
  - Si Absence lumière, (ou bien lumière insuffisante) : la lampe s'allume et reste allumée tant que ces deux conditions restent vérifiées.
  - Présence lumière, le système va vérifier l'état du rideau- Via un capteur de fin de course- on aura les deux cas suivants :
    - Si le rideau est fermé: la lampe s'allume.
    - Si le rideau est ouvert : la lampe s'éteint après une temporisation T1.
- Si non, la lampe s'éteint après une Temporisation T0.

Pendant la nuit :

Si le capteur détecte une présence la lampe s'allume et le programme se boucle avec les processus cité auparavant sans prendre en considération la lumière externe.

Une lampe extérieure reste allumée (pour cela on met un simple capteur de lumière, et un interrupteur pour choisir le mode (Automatique ou Manuel).

2.2. Les Critères :

Lorsqu'une temporisation se déclenche, le système doit suivre à temps réel tous les changements d'état que les capteurs peuvent subir à tout moment (la personne est toujours dans la salle mais ne bouge pas. Si elle quitte la salle et revient avant que la temporisation soit écoulée...Etc).

Le système doit aussi détecter les défauts lors de la transmission des données internes.

On peut à tout moment basculer vers la mode manuel ou automatique et ceci peut être effectué par la personne présente dans la salle ou bien par le superviseur.

En cas d'erreur, un bouton d'urgence et un capteur d'incendie sont prévus pour mettre en arrêt le système et envoyer l'information à l'automate pour localiser cette urgence.

2.3. La commande dans les différentes salles :

- Un Bureau de travail contient un seul interrupteur qui alimente une lampe
- Une salle de réunion ou de conférence contient un seul interrupteur qui alimente plusieurs lampes (Dans notre cas 3 lampes).
- Un espace de travail contient un interrupteur principal qui alimente et espace, et on a aussi plusieurs interrupteurs qui commandent chaque rangée séparément (Dans notre cas on a 3 rangées).

2.3.1. Dans les bureaux et les salles de réunion :

On a un seul interrupteur pour plusieurs lampes, donc, une fois que le bloc gestion d'éclairage traite la nécessité d'allumer ou non la lampe, il va transmettre à l'automate l'information via une seule Sortie 'Inter-EBOOL_Etage_Salle' qui va à son tour commander cet interrupteur.

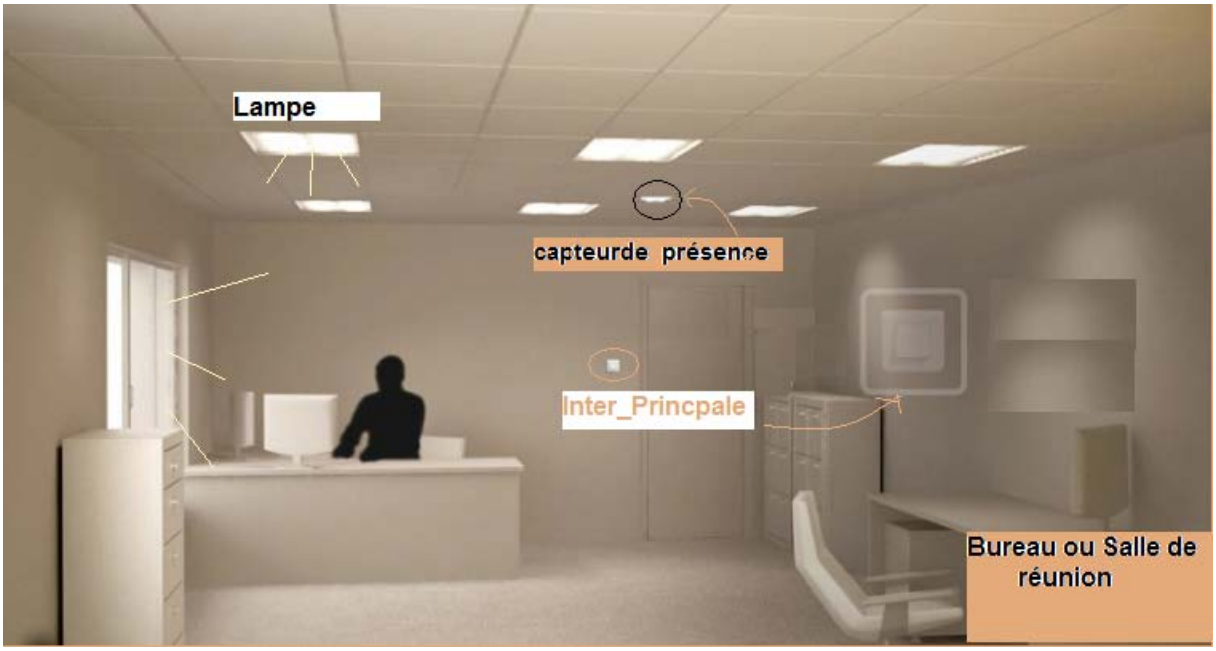


Figure IV.5 : La Commande dans les salles et les bureaux.

2.3.2. Dans les espaces de travail :

Pour commander l'éclairage dans les espaces de travail, le programme va procéder de la même manière que dans un bureau normal pour commander l'interrupteur principal d'une part , et d'autre part il va commander 3 interrupteurs, dont chaqu'un va alimenter une rangée contenant une ou plusieurs lampes.
La sorties prévu pour cette commande est :
'Inter_Numéro_Etage_Numéro_Salle_R_Numéro_rangé'.

Par exemple pour l'étage 2, espace de travail 1. La variable qui va commander la rangée numéro 3 est : **Inter_2_1_3**

Voici le schéma résumant la commande des différents interrupteurs dans un espace de travail :

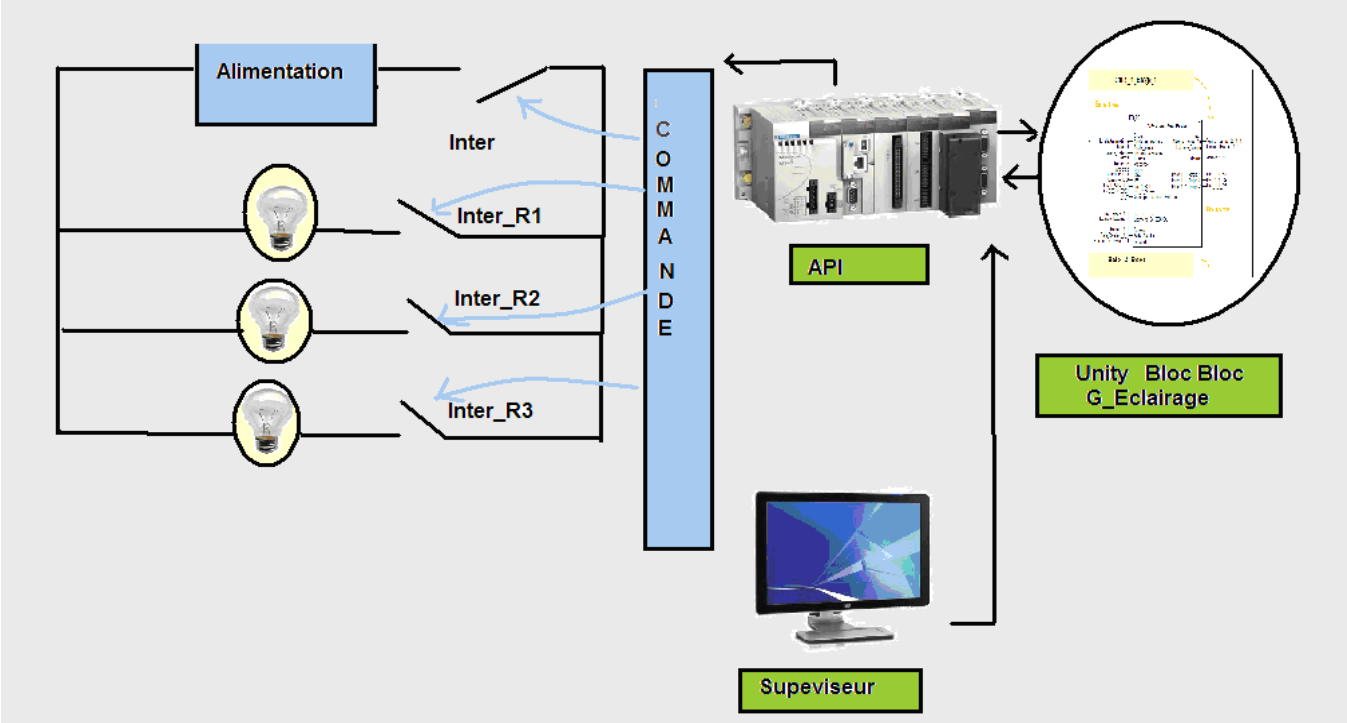


Figure IV.6: Commande des Interrupteur dans les espaces de travail.

Figure IV.7: Eclairage dans un espace de travail

2.3.3. Dans les salles de conférence :

Dans les salles de conférences, on a programmé le cahier de charge général d'éclairage appliqué dans les bureaux et les espaces de travail. Et on a aussi créé un bloc DFB pour automatiser la salle et ajouter un plus à notre application, pour cela on a prévu un bouton poussoir qu'on a attribué à la variable 'Conférence', qui va fermer le rideau, éteindre la lumière, bloquer l'accès à la salle, et lancer la projection.

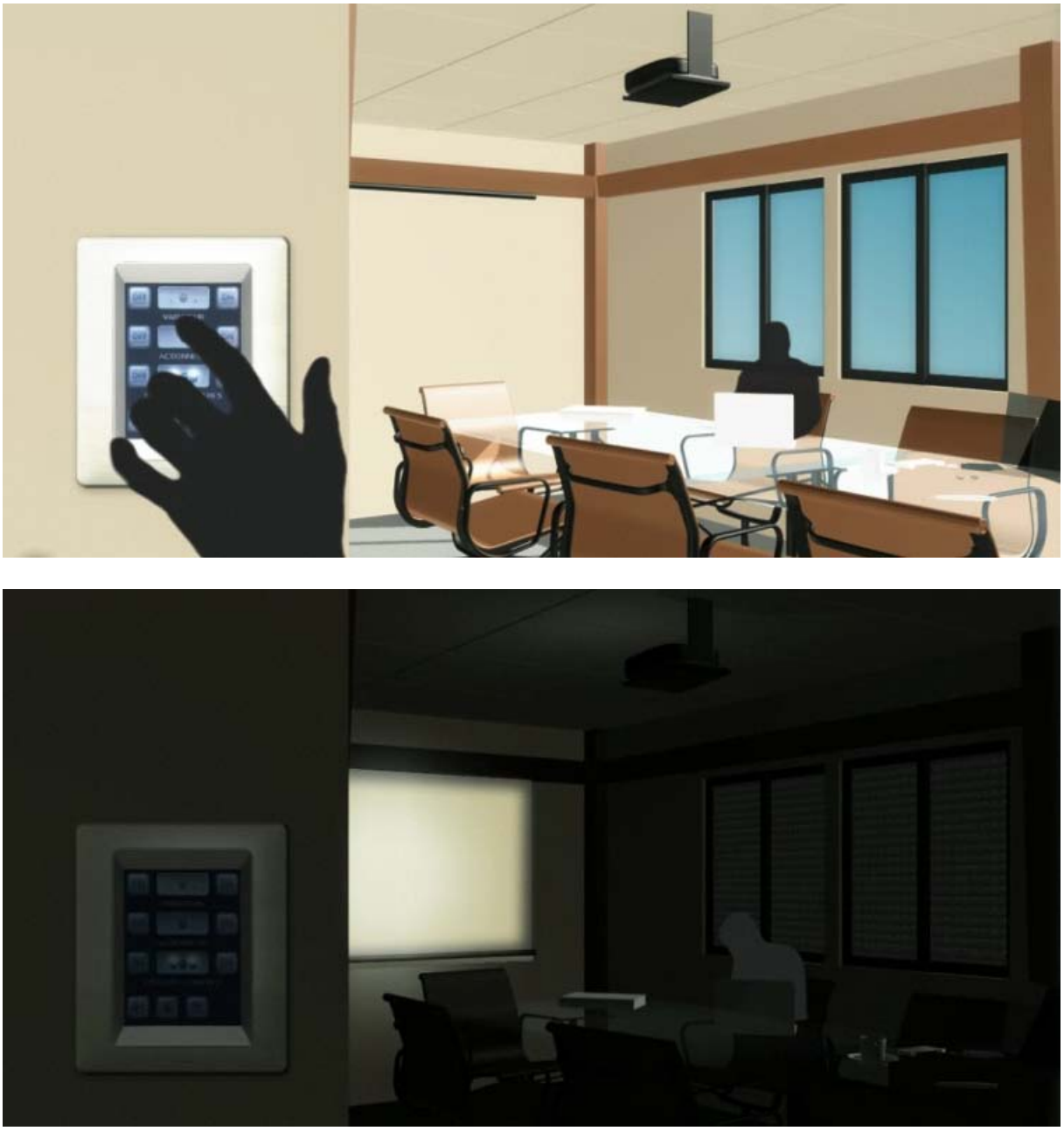


Figure IV.8 : Commande d'une salle de conférence

Remarque : dans les salles de conférence on a prévu un système de projection et un rideau qui peuvent êtres commandés par un moteur.

2.4. Programmation Sous Unity Pro :

Afin de réaliser ces scénarios plusieurs blocs fonction on été créés pour la gestion d'éclairage comportant les DFB suivants:

- Un bloc DFB de commande d'éclairage Compatible avec les différents types de salle.
- Un bloc DFB pour la détection d'erreur provenant des lampe et des fin de course rideau.

2.4.1. Le bloc Gestion d'éclairage :

Ce bloc contient une section programmée en Ladder qui va traiter les donné provenant des différents capteurs et les traduire en sortie afin de commander les différent interrupteurs qui se trouvent dans la salle.

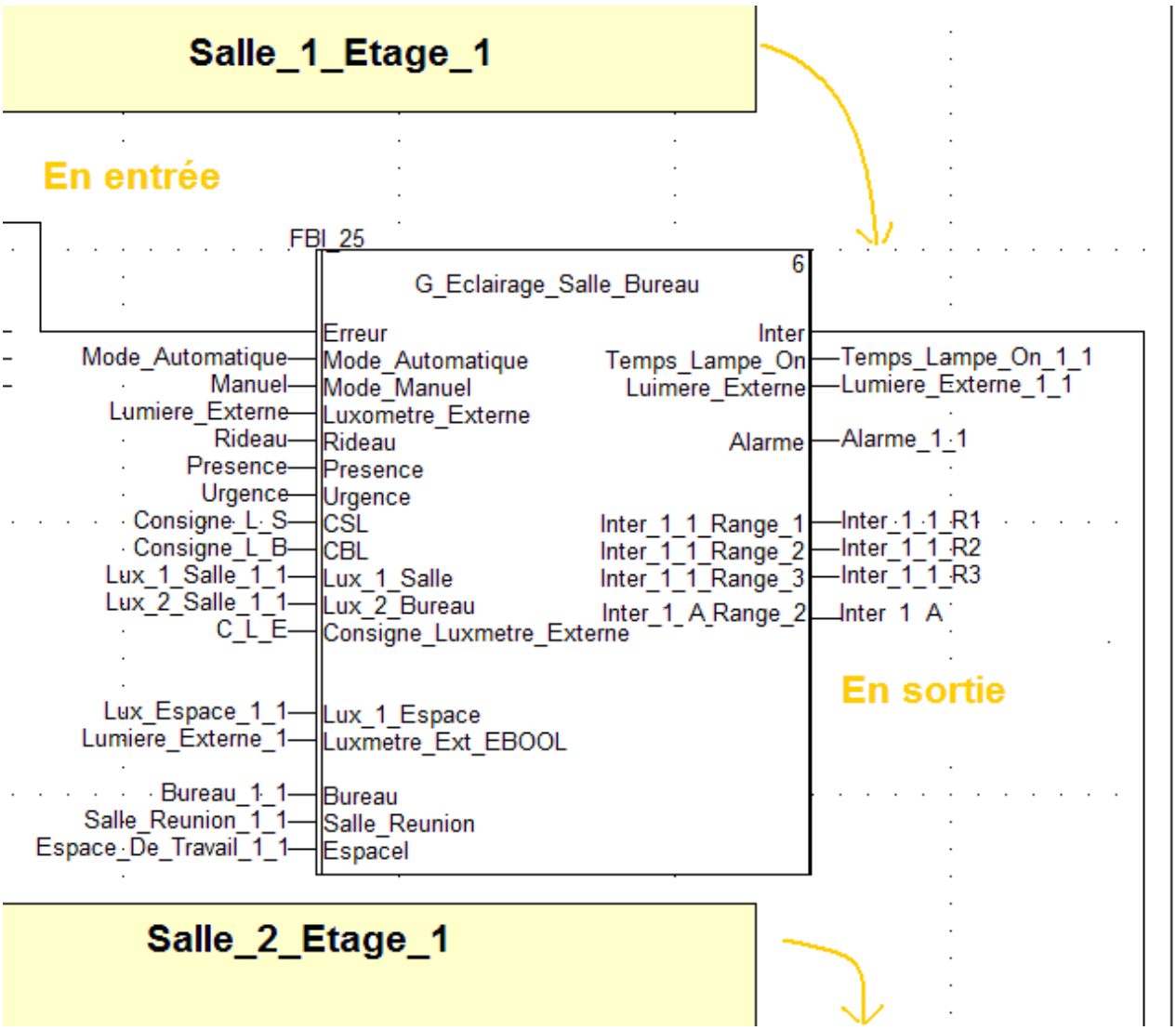


Figure IV.9 : Bloc DFB Gestion d'éclairage

Il contient :

En entrée :

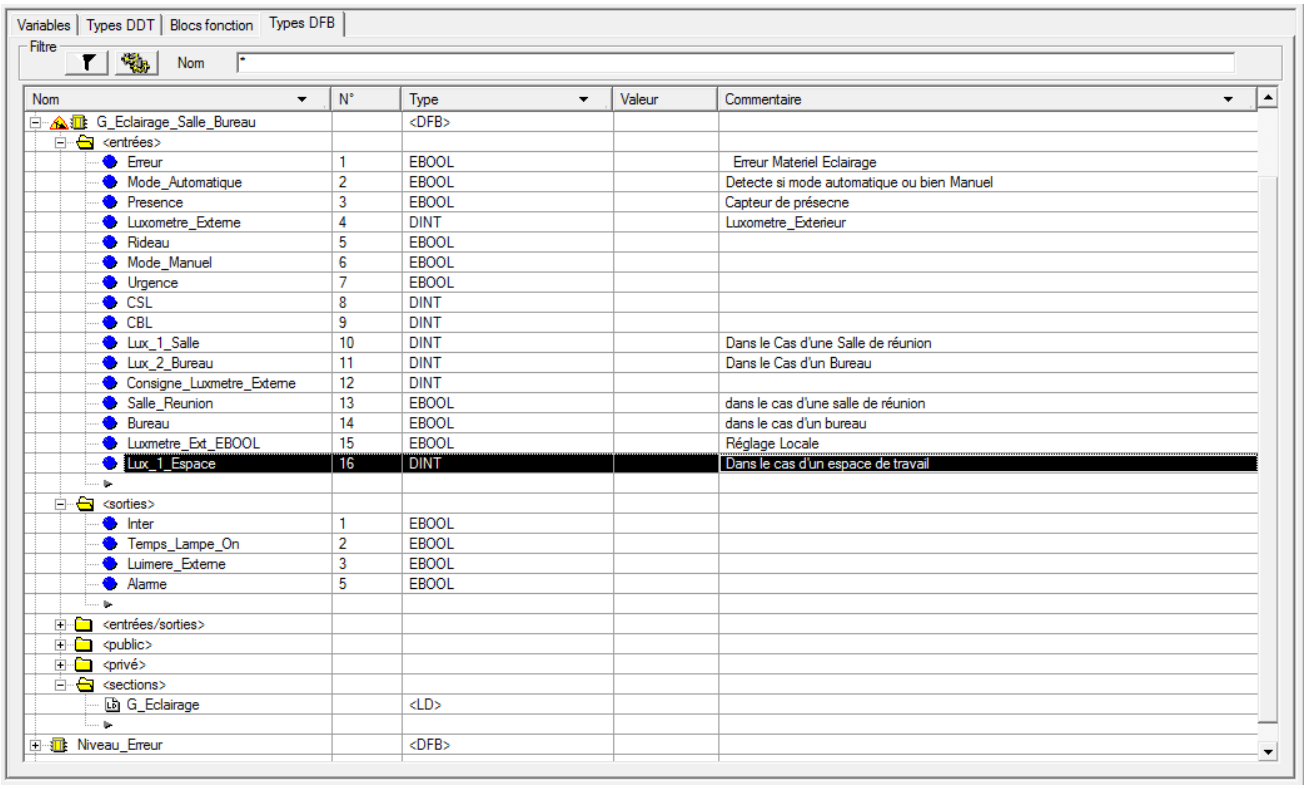
- Mode Automatique : bouton poussoir (EBOOL)
- Mode Manuel : bouton poussoir (EBOOL)
- Capteur de présence (EBOOL)
- Luxmètre externe (REAL)
- Luxmètre externe (EBOOL)
- Luxmètre Salle
- Luxmètre Bureau
- Luxmètre Espace
- Fin de course rideau (EBOOL)
- Consigne lumière Salle de Réunion (DINT)
- Consigne lumière Bureau de travail (DINT)
- Consigne lumière Espace de travail (DINT)
- Commande de l'interrupteur (EBOOL)
- Une entrée Urgence qui provient des différents Blocs d'Erreur

En sortie :

- Commande Interrupteur
- Temps_lampe_On
- Lumière Externe
- Lumière Interne
- Alarme
- Inter_i_Etage_Salle (pour les espaces de travail)

Les variables internes :

- L'horloge Interne de la temporisation (durée de la temporisation).
- Tempo_Lampe (déclenche le lacement de la temporisation)
- Les états Xi



Nom	N°	Type	Valeur	Commentaire
G_Eclairage_Salle_Bureau		<DFB>		
<entrées>				
Erreur	1	EBOOL		Erreur Materiel Eclairage
Mode_Automatique	2	EBOOL		Detecte si mode automatique ou bien Manuel
Presence	3	EBOOL		Capteur de présecne
Luxometre_Exteme	4	DINT		Luxometre_Exterieur
Rideau	5	EBOOL		
Mode_Manuel	6	EBOOL		
Urgence	7	EBOOL		
CSL	8	DINT		
CBL	9	DINT		
Lux_1_Salle	10	DINT		Dans le Cas d'une Salle de réunion
Lux_2_Bureau	11	DINT		Dans le Cas d'un Bureau
Consigne_Luxmetre_Exteme	12	DINT		
Salle_Reunion	13	EBOOL		dans le cas d'une salle de réunion
Bureau	14	EBOOL		dans le cas d'un bureau
Luxmetre_Ext_EBOOL	15	EBOOL		Réglage Locale
Lux_1_Espace	16	DINT		Dans le cas d'un espace de travail
<sorties>				
Inter	1	EBOOL		
Temps_Lampe_On	2	EBOOL		
Luimere_Exteme	3	EBOOL		
Alarme	5	EBOOL		
<entrées/sorties>				
<public>				
<privé>				
<sections>				
G_Eclairage		<LD>		
Niveau_Erreur		<DFB>		

Figure IV.10: Création du bloc G_Eclairage_Salle_Bureau

2.4.2. Le bloc Contrôle_Conf :

Ce bloc lit en entrée la commande provenant du bouton poussoir « conférence » et il va fermer le rideau, éteindre la lumière en commandant l'interrupteur principal, lancer la projection et bloquer l'accès à la salle par la sortie « Elt_Aimant ».

Commande_Conf		<DFB>		
<entrées>				
Conference	1	EBOOL		
Mode_Automatique	2	EBOOL		
<sorties>				
Moteur_Rideau	1	EBOOL		
Inter_Salle	2	EBOOL		
Projection	3	EBOOL		
Elt_Aimant	4	EBOOL		
<entrées/sorties>				
<public>				
<privé>				
<sections>				
Commande_Conference		<ST>		

Figure IV.11 : Création du bloc Control_Conf

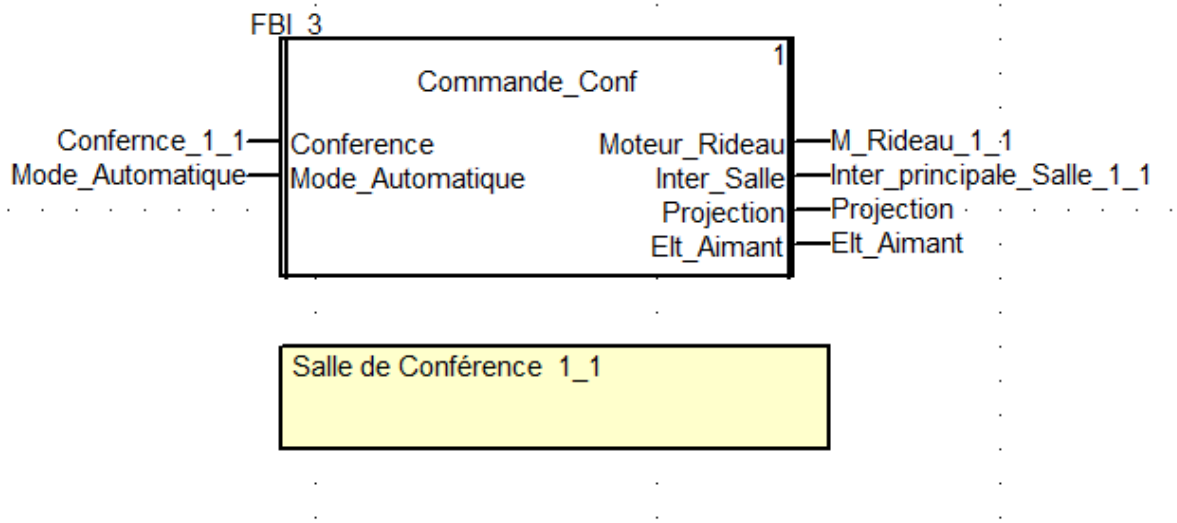


Figure IV.12 : Bloc Commande_Conf

2.4.3. Les Blocs des Erreurs :

Afin d'avoir un système autonome et fiable à la fois, on a crée des blocs pour 3 types d'erreur

2.4.3.1. Le Bloc DFB Erreur_Eclairage_L

Ce bloc est programmé afin de détecter si la lampe est défectueuse, et de localiser sa position dans la salle. Il nous donne aussi le nombre d'Erreurs de lampe défectueuses si on est dans le cas d'une salle qui contient plusieurs lampes.

Ce bloc va comparer entre les différentes informations provenant des capteurs de lumière externe et interne et les interrupteur, donc par exemple si l'interrupteur de la salle 1 est en ON et que le luxmètre indique qu'il ne y'a pas suffisamment de lumière dans la salle, donc on détecte une erreur de lampe…etc.

2.4.3.1.1. Localisation de lampe défectueuse :

Pour localiser la position de l'erreur on fait appel au luxmètre et aux consignes de lumière présentée par les variables CSL et CBL (des bureaux et des salles), avec ces deux informations on pourra savoir quelle lampe est défectueuse.

Exemple :

Si par exemple l'interrupteur de la salle 1 est 'On' et que le luxmètre indique que le taux de luminosité est entre CSL/4 et CSL/3 donc on aura la lampe [i=numéro de la salle, j=numéro de la lampe] est défectueuse, après il va sommer les erreurs pour nous donner le nombre d'erreur dans la salle.

Remarque :

Plusieurs plages ont été prévues pour localiser d'une manière précise la lampe défectueuse, mais le fonctionnement de cette méthode dépend de la position du luxmètre dans la salle et de la consigne qu'il faut donner avec exactitude.

Ce bloc aura *:*

En entrée :

- Les différentes commandes des interrupteurs des 5 salles de l'étage.
- Le taux de luminosité de chaque salle et de chaque rangé si on est dans le cas d'un espace de travail.
- le taux de luminosité externe.
- Les Consigne de lumière (dans un état normal, Càd une salle bien éclairé).

En sortie :

5 Erreurs d'éclairage (une Erreur pour chaque salle), et qu'il va envoyer au bloc DFB Erreur Générale.

La lampe défectueuse (sous forme d'une matrice pour localiser l'erreur.)

Le nombre de lampe défectueuse dans chaque salle.

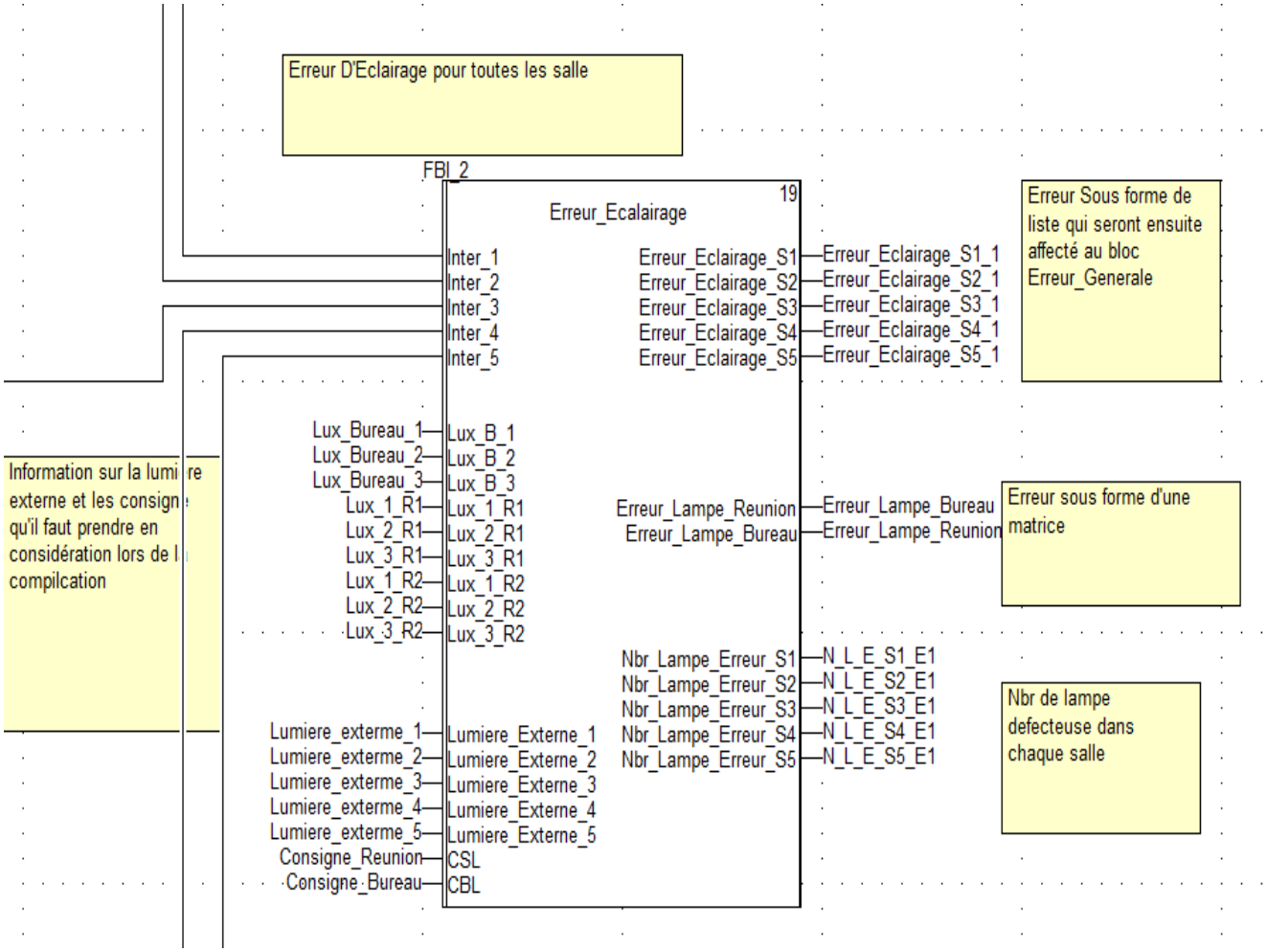


Figure IV.13 : Bloc Erreur_Eclairag

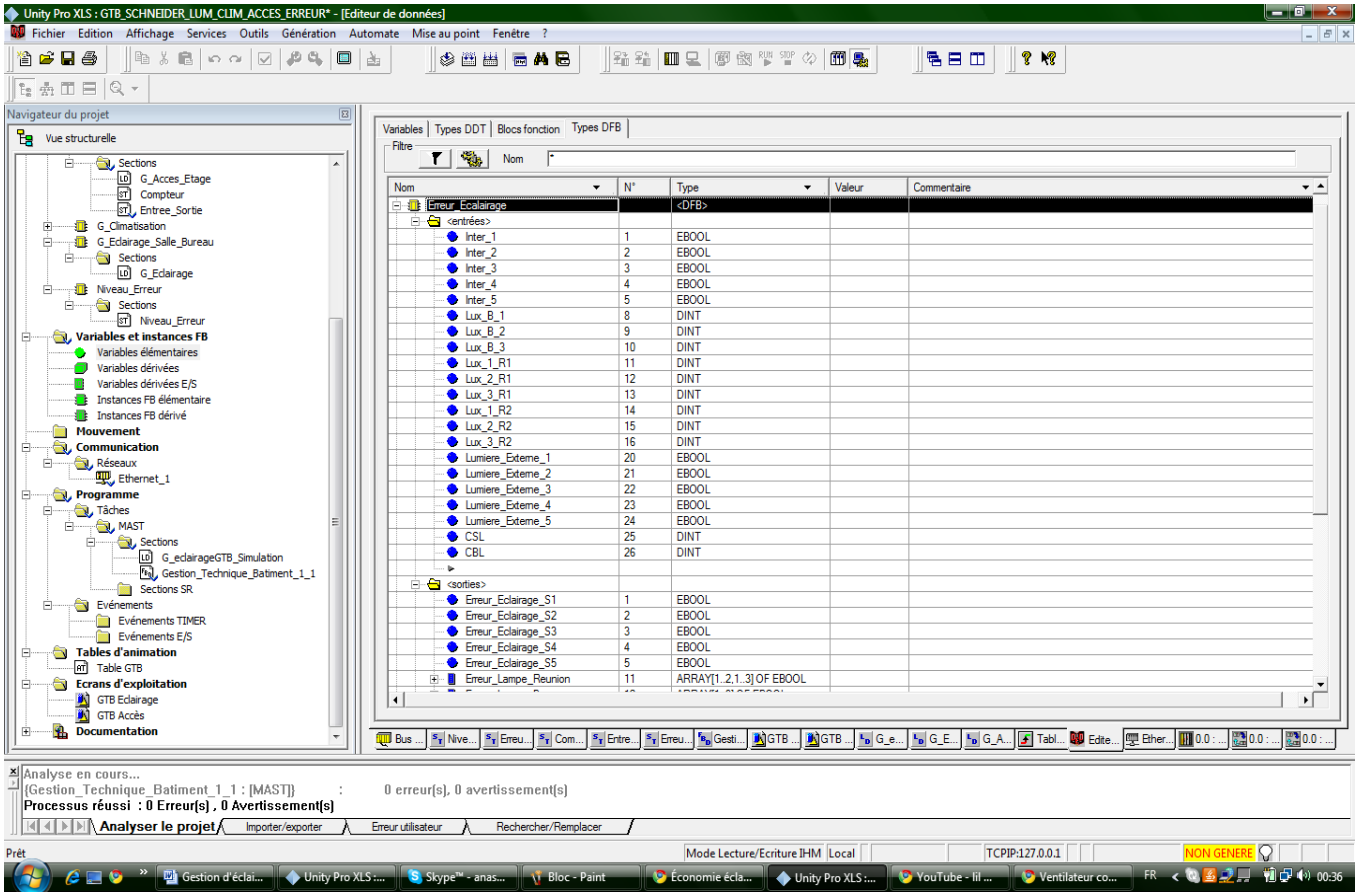


Figure IV.14 : Création du bloc Erreur_Eclaireage

2.4.3.2. Le bloc DFB Erreur Générale:

Ce bloc lit directement les erreurs provenant des capteurs, et les Erreurs provenant du bloc DFB Erreur_Eclaraiga_L. Il va ensuite les traiter et les classer en 3 catégories.

Par exemple si on a une erreur dans le capteur de lumière, ou bien dans le capteur de présence ou une erreur de fin de course rideau...il va signaler qu'il y'a une Erreur dans le Système d'éclairage, idem pour le systèmes de climatisation et d'accès.

Il va aussi donner le niveau d'erreur qui correspond au nombre d'erreur dans chaque salle et tout cela dans le but de donner un certain ordre de priorité aux pannes et de faciliter la maintenance.

Exemple:

Si on a une erreur dans le fin de course Rideau et Erreur des Volet de climatiseur il va nous dire que le niveau d'erreur est de 2.

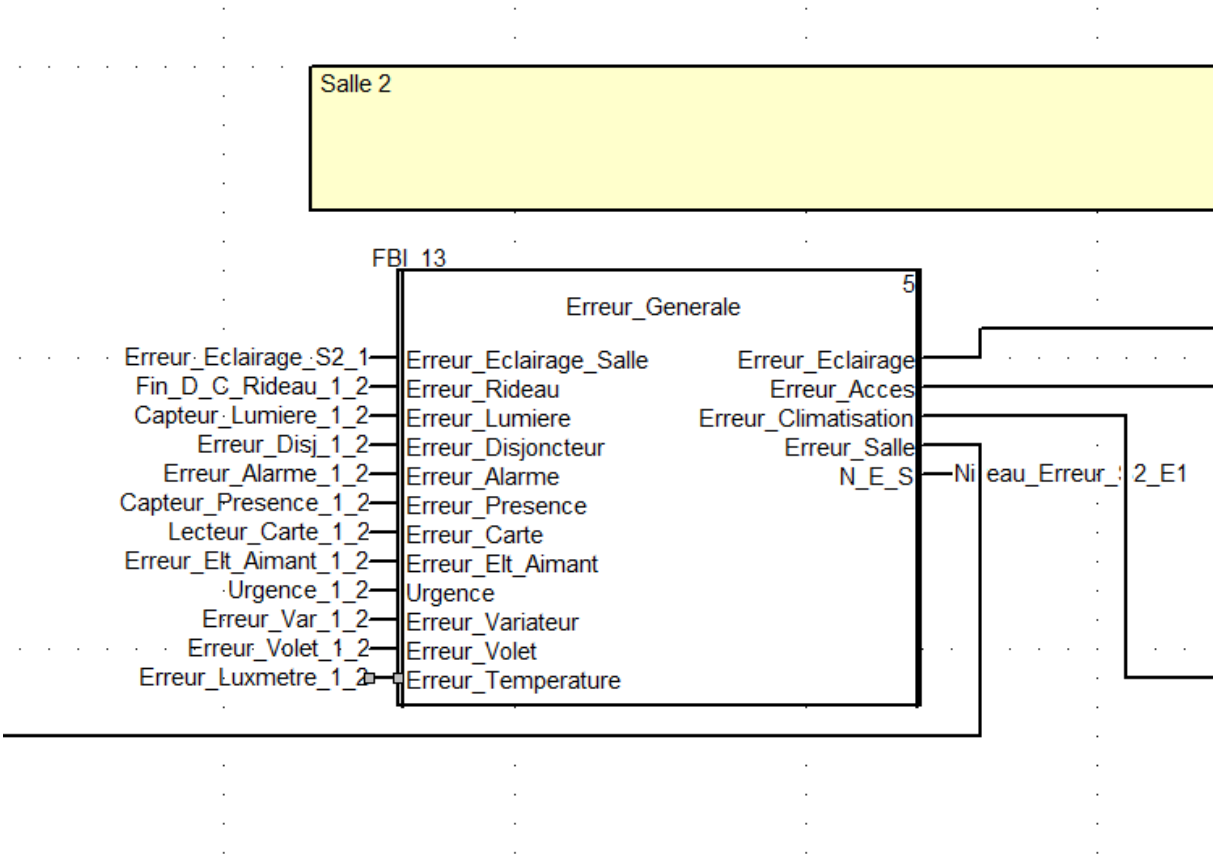


Figure IV.15: E/S Bloc DFB Erreur_Generale

Erreur_Generale		<DFB>		
<entrées>				
Erreur_Eclairage_Salle	1	EBOOL		
Erreur_Rideau	2	EBOOL		
Erreur_Lumiere	3	EBOOL		
Erreur_Disjoncteur	4	EBOOL		
Erreur_Alarme	5	EBOOL		
Erreur_Presence	6	EBOOL		
Erreur_Carte	7	EBOOL		
Erreur_Elt_Aimant	8	EBOOL		
Urgence	9	EBOOL		
Erreur_Variateur	10	EBOOL		
Erreur_Volet	11	EBOOL		
Erreur_Temperature	12	EBOOL		
<sorties>				
Erreur_Eclairage	1	EBOOL		
Erreur_Acces	2	EBOOL		
Erreur_Climatisation	3	EBOOL		
Erreur_Salle	4	EBOOL		
N_E_S	5	INT		Niveau d'erreur dans chaque salle
<entrées/sorties>				
<public>				
<privé>				
<sections>				
Erreur_Generale		<ST>		

Figure IV.16 : Création du Bloc Erreur Générale

2.4.3.3. Le bloc Niveau Erreur :

Ce bloc aura en entrée tous les niveaux d'erreur provenant de chaque salle, il va les traduire en sortie en un niveau d'erreur général (de l'étage), et suivant le niveau de l'erreur il va déclencher des alarmes, Alarme de niveau A (la plus élevée), Alarme B, ou bien une Alarme de niveau C.

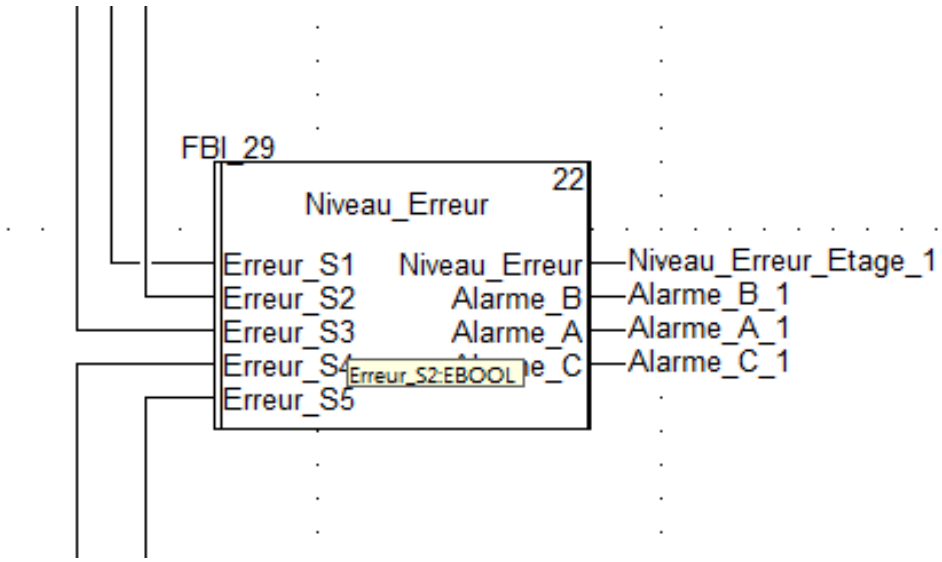


Figure IV.17: Bloc DFB Niveau_Erreur

Niveau_Erreur		<DFB>		
<entrées>				
Erreur_S1	1	EBOOL		
Erreur_S2	2	EBOOL		
Erreur_S3	3	EBOOL		
Erreur_S4	4	EBOOL		
Erreur_S5	5	EBOOL		
<sorties>				
Niveau_Erreur	1	INT		
Alarme_B	2	EBOOL		
Alarme_A	3	EBOOL		
Alarme_C	4	EBOOL		
<entrées/sorties>				
<public>				
<privé>				
S1		INT		
S2		INT		
S3		INT		
S4		INT		
S5		INT		
<sections>				
Niveau_Erreur		<ST>		

Figure IV.18 : Création du bloc Niveau d'Erreur

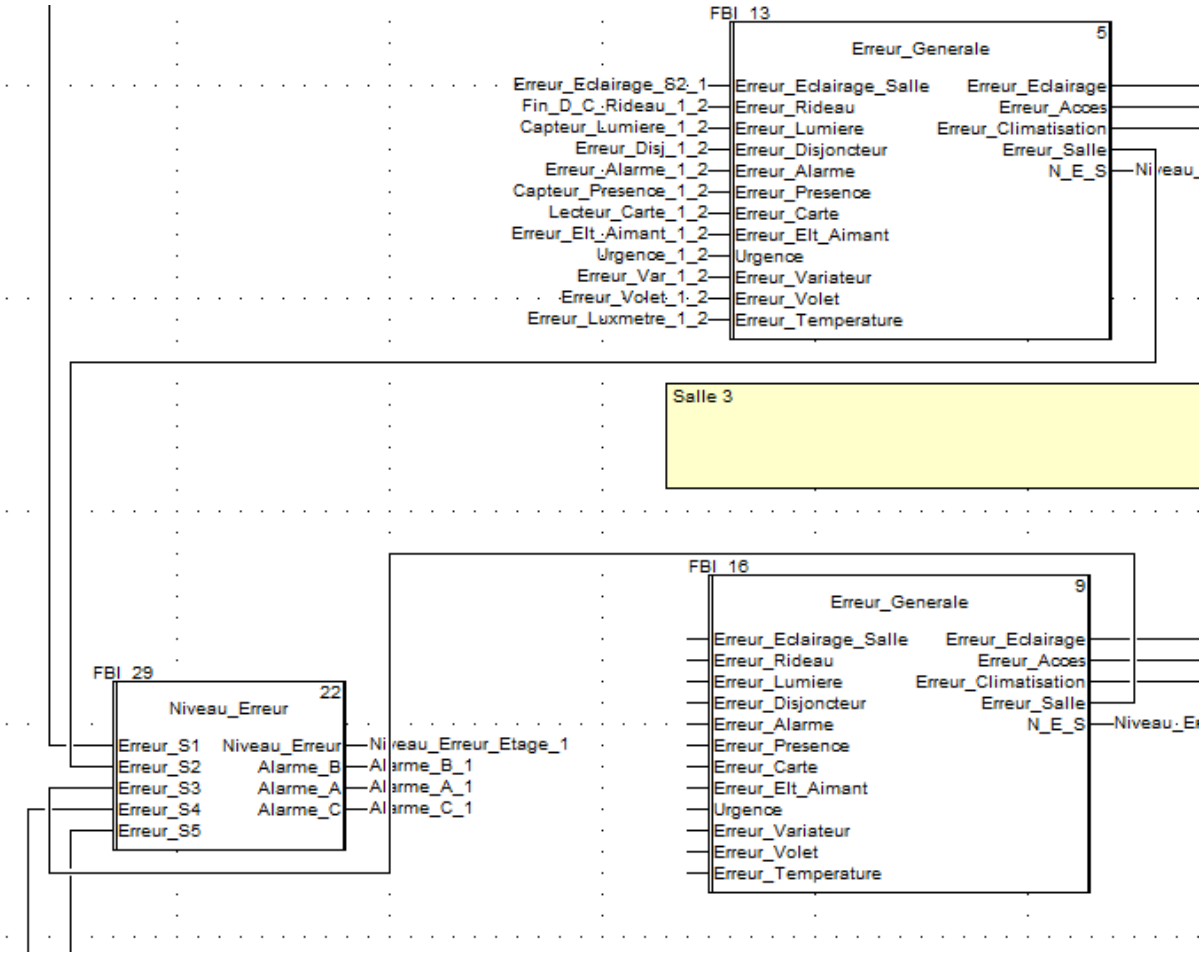


Figure IV.19 : Branchement des blocs

2. Gestion de climatisation :

La gestion de climatisation permet de commander d'une part les ventilateurs centraux qui vont fournir tout le bâtiment en air conditionné, et d'autre part les volets des climatiseurs dans chaque salle afin de régler avec précision les besoins en air conditionné dans chaque salle.

3.1. Cahier de charge :

Nous avons une centrale de ventilation qui contient quatre ventilateurs qui alimentent tout le bâtiment et des volets dans chaque climatiseur qui se trouve dans chaque salle pour régler l'alimentation en air conditionné de chaque salle de façon autonome.

Afin de démarrer les quatre ventilateurs nous allons créer un bloc qui va les démarrer l'un après l'autre avec une temporisation entre chaque allumage pour éviter toute surcharge sur le réseau électrique, un variateur de vitesse Altivar ATV71 s'occupera aussi de commander la vitesse de rotation des ventilateurs qui sera commandable à partir de l'interface de supervision, nous pourrons aussi si nous le souhaitons lancer le mode manuel qui permet de commander l'allumage des ventilateurs en local, de cette manière l'automate n'aura plus aucun droit sur le démarrage des ventilateurs.

Pour ce qui est des volets des climatiseurs, nous avons créé un bloc qui commande leur angle d'ouverture en fonction de la différence entre la consigne de température pour la salle et la température de la salle à tout instant, nous avons créé des plages de différence.

3.2. Le variateur de vitesse ATV71 :

L'ATV71 est un variateur de vitesse de la gamme Altivar de chez Schneider Electric, il est constitué principalement d'un convertisseur statique et d'une électronique de commande. Il contient aussi un étage de correction du facteur de puissance afin de respecter les normes de compatibilité électromagnétiques.

Figure IV.20 : Variateur de vitesse ATV71

Son principe de fonctionnement est comme suit, dans un moteur à courant alternatif, la vitesse mécanique du rotor est liée à la fréquence des courants au stator. Ce lien mathématique rend possible une commande de la vitesse du rotor par la commande de la fréquence du courant au stator. C'est ce que l'on appelle la condition de synchronisme qui s'exprime différemment selon que l'on considère une machine synchrone ou une machine asynchrone.

Pour une machine synchrone, la condition de synchronisme est :

$$Ns = \frac{60 \times f}{p}$$

Avec :

- *Ns*, la vitesse de synchronisme en tours par minute
- *f*, la fréquence d'alimentation en hertz
- *p*, le nombre de paires de pôles

Pour une machine asynchrone, la condition de synchronisme est :

$$g = \frac{Ns - N}{Ns} \times 100$$

Avec :

- *g*, le glissement en %
- *Ns*, la vitesse de synchronisme en tours par minute
- *N*, la vitesse de l'arbre (vitesse réelle) en tours par minute

Ainsi, il existe une relation directe entre le pilotage de la fréquence du courant au stator et la vitesse mécanique du rotor qui permet, pour toute vitesse mécanique souhaitée, de fixer la fréquence statorique correspondante. C'est sur ce principe que se base le fonctionnement du variateur de vitesse : commander une vitesse de rotation mécanique en commandant la fréquence du courant statorique.

3.3. Programmation :

Nous avons donc créé les blocs pour le démarrage des ventilateurs et un autre pour la commande des volets.

Pour le programme de démarrage nous avons :

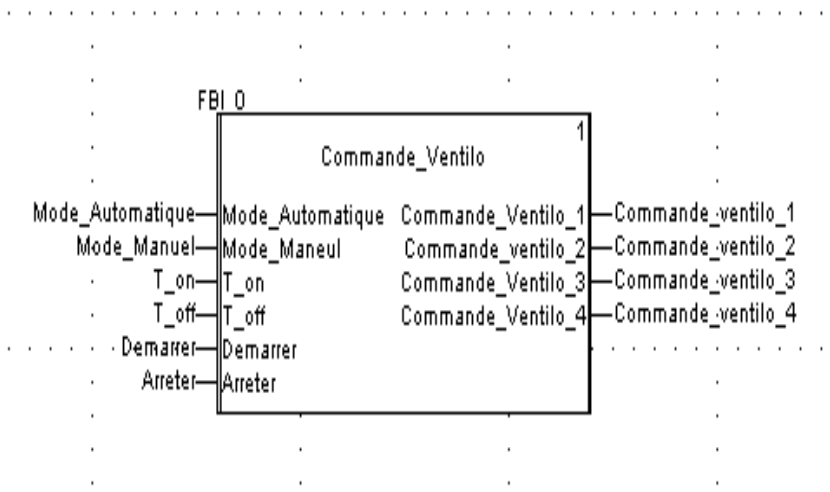


Figure IV.21 : Bloc de démarrage des ventilateurs

En entrées : Nous avons donc en entrées la commande en mode manuel ou automatique ainsi que les temporisations.

- Mode_manuel
- Mode_automatique
- Ton
- Toff

En Sorties : Nous avons en sorties les commandes des ventilateurs.

- Commande_ventilo1
- Commande_ventilo2
- Commande_ventilo3
- Commande_ventilo3

Pour le programme de commande des volets nous avons :

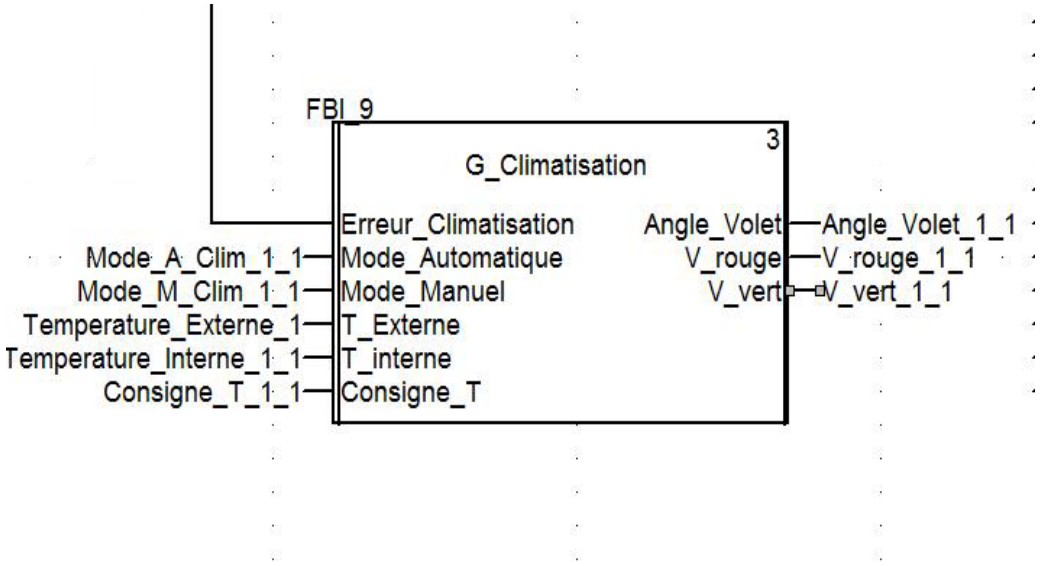


Figure IV.22 : Bloc gestion de climatisation

En entrées : Les données recueillies par les capteurs, la consigne ainsi qu'une commande pour le mode manuel ou automatique, nous avons aussi l'entrée erreur climatiseur qui arrête le processus en cas d'erreur dans la climatisation.

- Temp externe
- Temp_interne
- Consigne
- Mode auto
- Mode manuel
- Erreur_clim

Sorties : En sortie nous avons la commande de l'angle du volet, et un voyant rouge ou vert.

- L'angle du volet
- Voyant rouge
- Voyant vert

3.4. La supervision :

Pour la partie supervision nous avons créé des génies et des super génies des ATV71 qui nous permettent d'avoir à tout moment la fréquence délivrée par ces dernières nous pouvons aussi interagir avec ces ATV71 si nous le souhaitons, s'il y a une erreur sur l'ATV71 un carré rouge l'entoure pour que le superviseur le remarque tout de suite.

.

Figure IV.21 : Pop Up de l'atv 71

Sinon dans chaque salle nous avons créé un Pop Up qui s'affiche lorsqu'on clique sur le climatiseur pour connaître la température, la consigne ainsi que l'angle d'ouverture du climatiseur, sinon dans chaque salle s'il y a une erreur sur le climatiseur un carré rouge l'entoure.

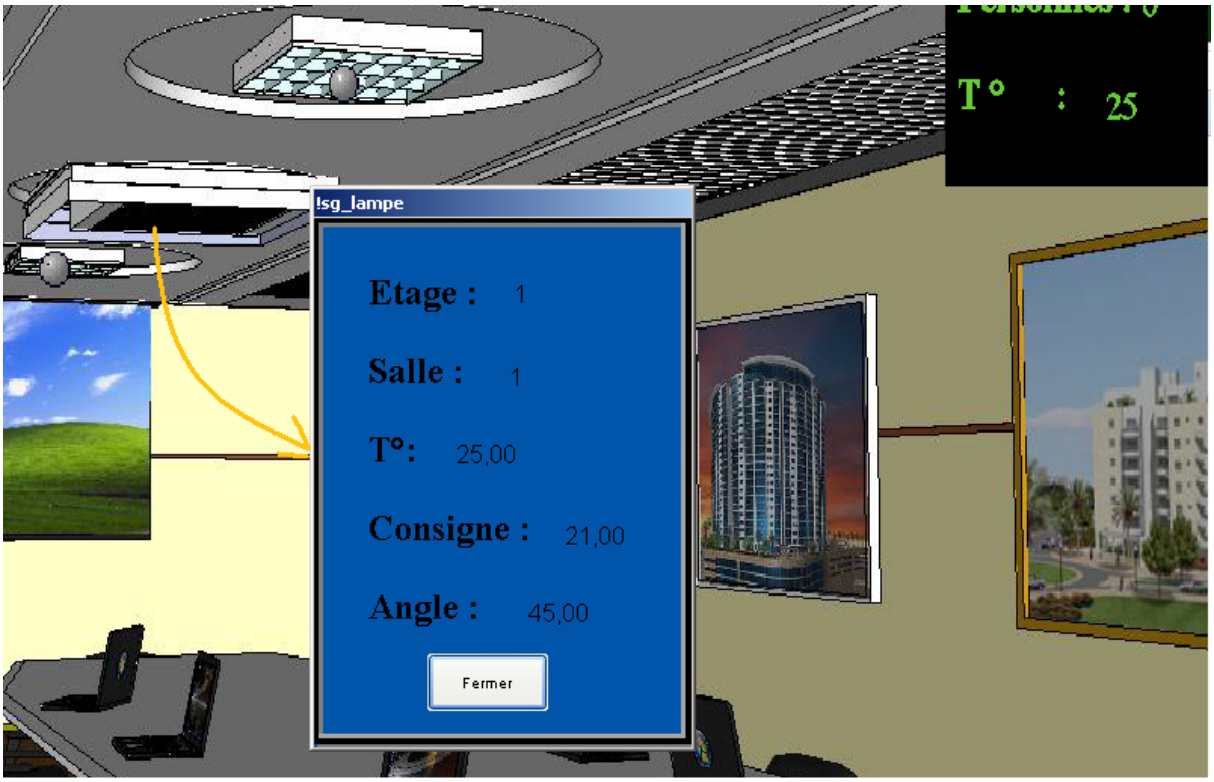


Figure IV.22 : Pop Up du climatiseur

4. Le Contrôle d’accès :

Pour assurer la sécurité des employés on organisera l’accès aux différentes salles, on a programmé un système qui va contrôler l’accès dans chaque salle.

Pour la commande, on a réalisé un cahier de charge et on l’a programmé sous forme de bloc DFB qu’il sera ensuite implémenté dans l’automate, pour la supervision on a crée des pages sous vijeo citect pour suivre en temps réel le déroulement de ce système.

4.1. But du système :

On désire construire un système chargé de contrôler l'accès de certaines personnes aux divers bâtiments d'un "lieu de travail".

4.2. L’autorisation :

Dans notre cas, c.à.d. du bâtiment Schneider Electric, le contrôle s'effectue sur la base de l'autorisation que chaque personne concernée est censée posséder. Cette autorisation doit lui permettre, sous le contrôle du système, de pouvoir pénétrer dans certaines salles et pas dans d'autres. Par exemple une certaine personne, **personne1** est autorisée à accéder à la **salle1** et pas dans la **salle2**, par contre, une autre personne **personne2** a le droit de pénétrer dans ces deux salles. Ces autorisations sont données de façon "permanente": autrement dit, elles ne changent pas durant le fonctionnement normal du système.

Lorsqu'une personne se trouve à l'intérieur de d’une salle, sa sortie doit également être contrôlée par le système de façon à ce qu'il soit possible de savoir à quelle heure elle est rentrée et à quelle heure elle est sortie.

4.3. Les Cartes magnétiques :

Chaque personne reçoit une carte magnétique qui lui est assignée en propre au moyen d'un identificateur unique gravé sur celle-ci. Des lecteurs de cartes sont installés à chaque entrée et à chaque sortie de bâtiment. À proximité de chaque lecteur, on trouve deux voyants : un voyant rouge et un voyant vert. Chacun de ces voyants peut être allumé ou éteint.

Pour cela on distingue 7 types de cartes magnétiques selon la fonction de la personne (classées par droit d'accès)

- Carte Directeur (il peut accéder partout)
- Carte Service de maintenance et de nettoyage (ils peuvent accéder partout et au bureau du Directeur/Employés s'il y'a une demande)
- Carte Employé normal (ils accèdent aux locaux de travail /bureaux)
- Carte Invité (il accède aux salles de conférence au restaurant)
- Carte Stagiaire (Classe de cours, locaux de travail, restaurant…)
- Carte Visiteur (Showroom, Restaurant)
- Carte Parking (Ayant deux niveaux d'accès 1 et 2 pour chaque parking)

Le lecteur de cartes va envoyer les informations concernant le niveau d'accès au parking, aux salles, et un code personnel qui permet à l'employé d'accéder à son propre bureau sans contrôle.

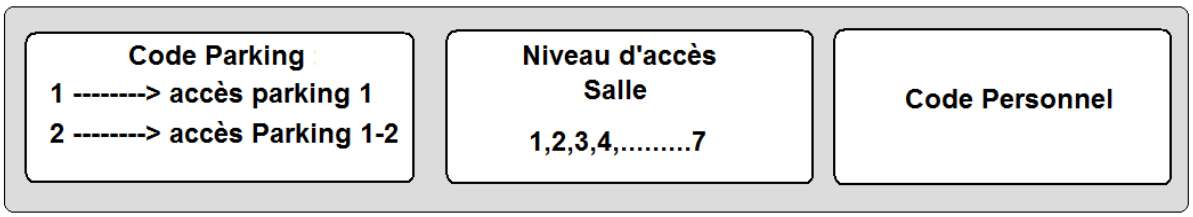


Figure IV.25 : les informations provenant du lecteur carte

Exemple :

Le code 14777 correspond a un employé qui a le droit d'accéder au parking 1 et aux salles qui on un niveau de sécurité égal ou inférieur à 4, et il accède sans contrôle à son bureau qui corresponde au 777

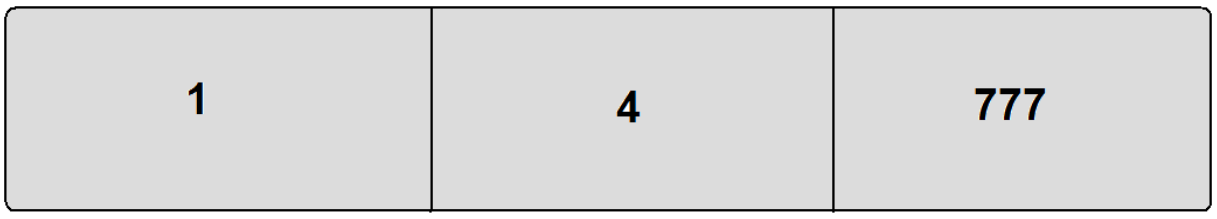


Figure IV.26 : exemple d'un code de carte

Remarque : chaque salle aura un code qui sera introduit d'une manière locale.

4.4. L'électroaimant :

L'accès des personnes d'une salle à une autre s'effectue sous la commande des électroaimants qui sont normalement bloqués : personne ne peut les franchir sans le contrôle du système. Lorsqu'un électroaimant est débloqué par le système (voir le paragraphe ci-dessous), le passage éventuel d'une personne est détecté par un capteur.

4.5. Protocole d'accès :

L'entrée-sortie dans une salle obéit à une procédure systématique composée d'une suite d'événements qui sont les suivants :

Une personne souhaitant entrer ou sortir d'une salle introduit sa carte dans le lecteur carte concerné.

On se trouve alors devant l'alternative suivante :

- Si la personne est autorisée à pénétrer dans le la salle en question (elle est toujours autorisée à sortir), le voyant vert s'allume et l'électroaimant se débloque pour 30 secondes. On se trouve alors en présence d'une nouvelle alternative:
    - Dès que quelqu'un franchit effectivement la porte avant la fin du laps de temps de 30 secondes le voyant vert s'éteint aussitôt et l'électroaimant se bloque.
    - Si, par contre, 30 secondes passent sans que personne ne franchisse la porte, le voyant vert s'éteint alors et l'électroaimant se bloque également.
- Si la personne n'est pas autorisée à pénétrer dans la salle, le voyant rouge s'allume pour deux secondes et, bien sûr, l'électroaimant /porte reste bloqué.
- Par contre si la personne n'est pas autorisée à pénétrer dans la salle et qu'elle parvient à y accéder en forçant l'accès, une alarme se déclenche.
- La personne n'ayant pas accès à une salle peut en revanche demander au propriétaire de libérer l'accès.

4.5.1. Les Critères :

En cas d'urgence, un bouton et un capteur d'incendie sont prévus pour mettre en arrêt le système et localiser cette urgence.

On peut à tout moment basculer vers la mode manuel ou automatique et ceci peut être effectué par la personne présente dans la salle ou bien par le superviseur.

4.6. La programmation sous Unity Pro :

On a les variables suivantes :

En e*ntrée :*

- Etat_Mode (Mode manuel ou automatique)
- Presence_Etage_Salle (Un capteur de présence)
- E_Tourniquet_Etage_Salle (Bloqué ou ouvert – l'état)

- Demande_Etage_Salle (Demander de libérer l'accès)
- Carte1_Etage_Salle (Carte magnétique Niveau 1)
- Carte2_Etage_Salle (Carte magnétique Niveau 2)
- Carte3_Etage_Salle (Carte magnétique Niveau 3)
- Carte4_Etage_Salle (Carte magnétique Niveau 4)
- Carte5_Etage_Salle (Carte magnétique Niveau 5)
- Carte6_Etage_Salle (Carte magnétique Niveau 6)
- Carte6_Etage_Salle (Carte magnétique Niveau 7)
- Defaut__Accès
- Defaut_Lecteur_Carte_Etage_salle
- Urgence.
- Capteur_Externe_Etage_Salle
- Capteur_interne_Etage_Salle

En sortie :

- Alarme_Etage_Salle
- EltAimant_Etage_Salle (libérer/bloquer l'accès)

Les variables internes :

- Tempo_Etage_Salle (durée de la temporisation). Pour la fermeture et l'ouverture
- Tempo_Presnence
- Xi

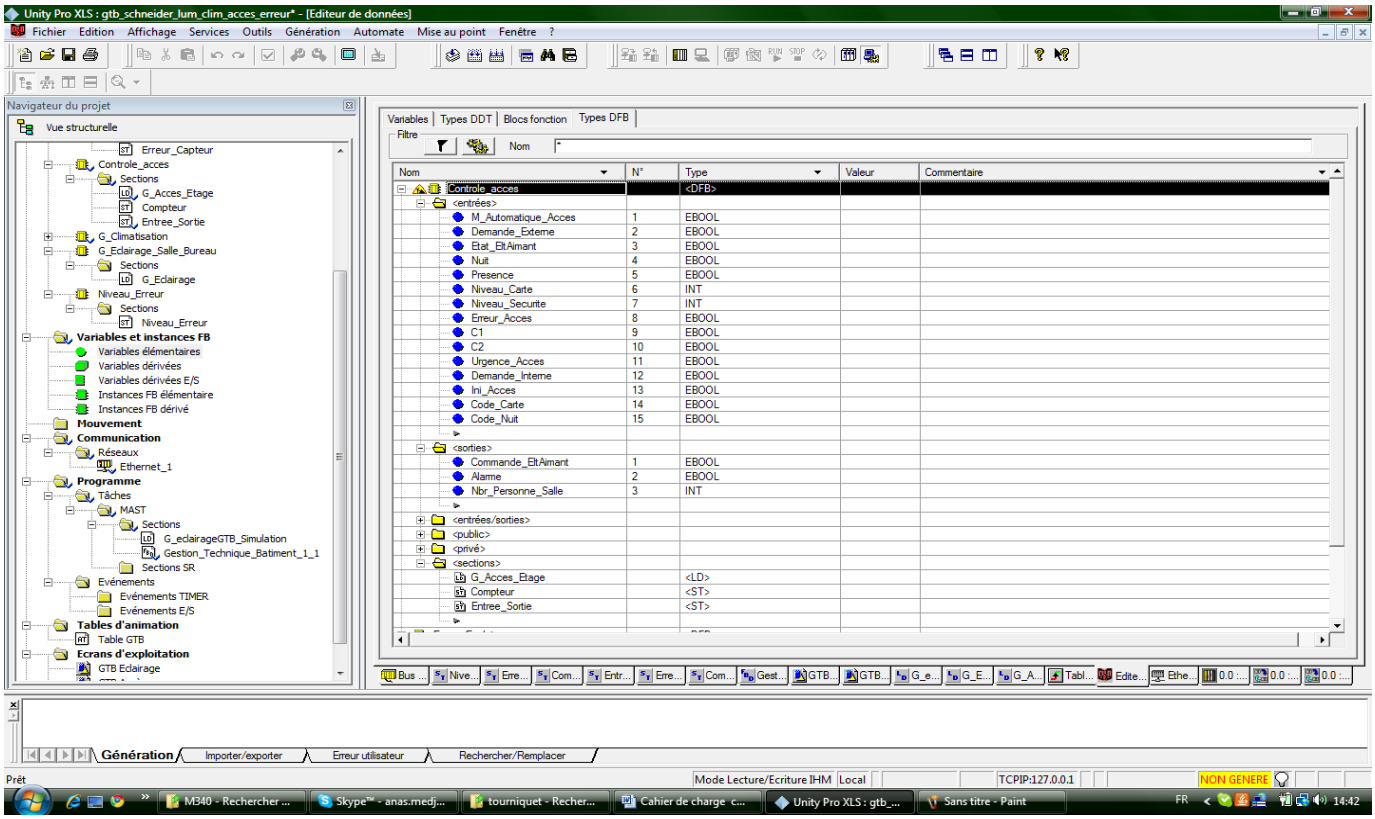


Figure IV.27 : Création du bloc DFB Contrôle D'accès

4.6.1. Interprétation des programmes:

La section du bloc Contrôle d'accès contient 3 sous-programmes, le programme Gestion accès étage pour le contrôle d'accès, un compteur pour donner le nombre exact des personnes présentes dans la salle).

On trouve aussi le sous-programme Entrée_Sortie qui va détecter si la personne est entrée dans la salle ou pas, idem si la personne quitte la salle, pour cela on a utilisé deux capteurs, un capteur se trouvant a l'extérieurs de la salle et l'autre à l'intérieur , ensuite le programme va combiner les changement des état de ces deux derniers pour savoir si la personne est entrée ou bien a quitté la salle.

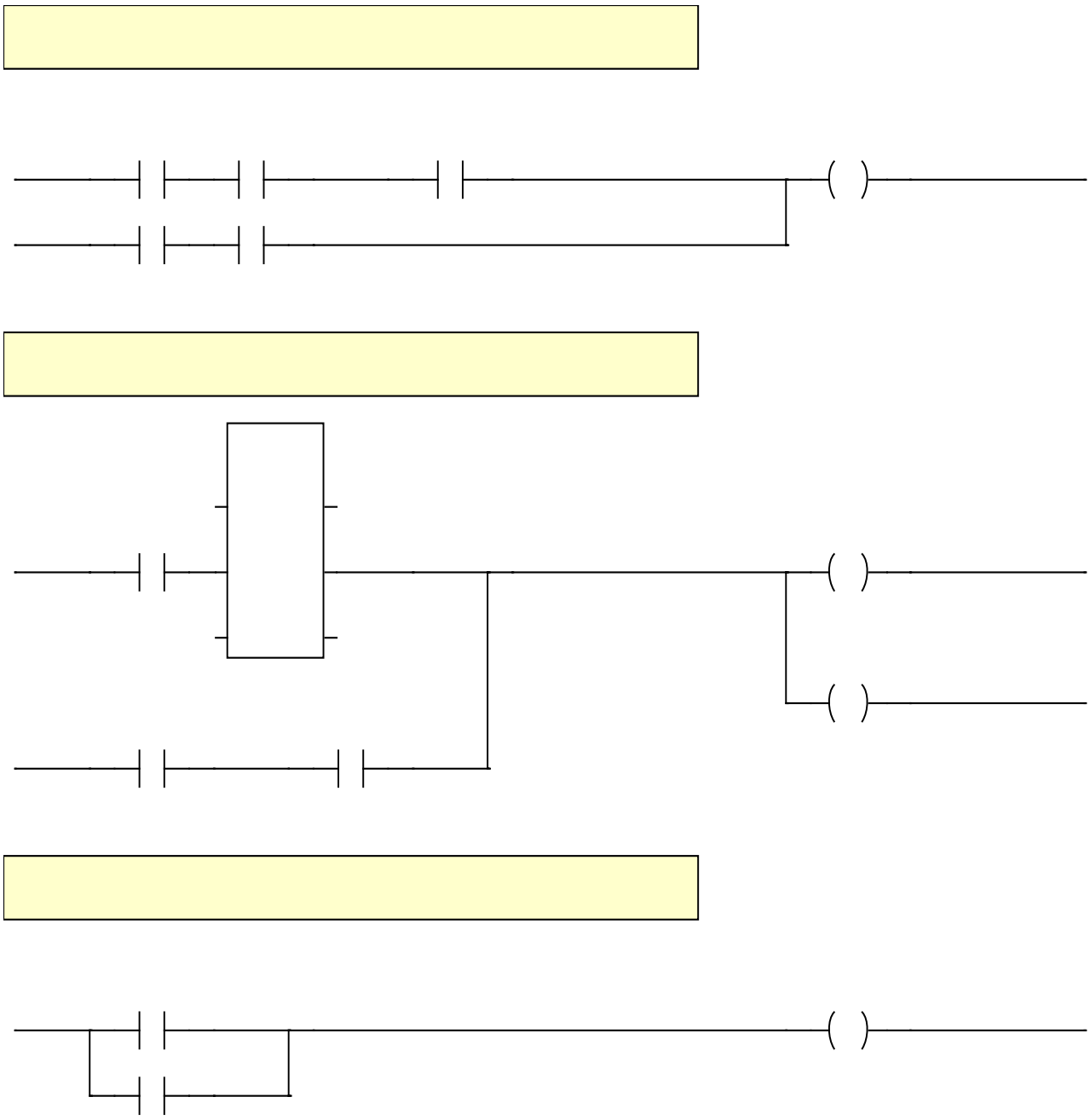


Figure IV.28 : Exemple L'appel des différents sous programme des DFB

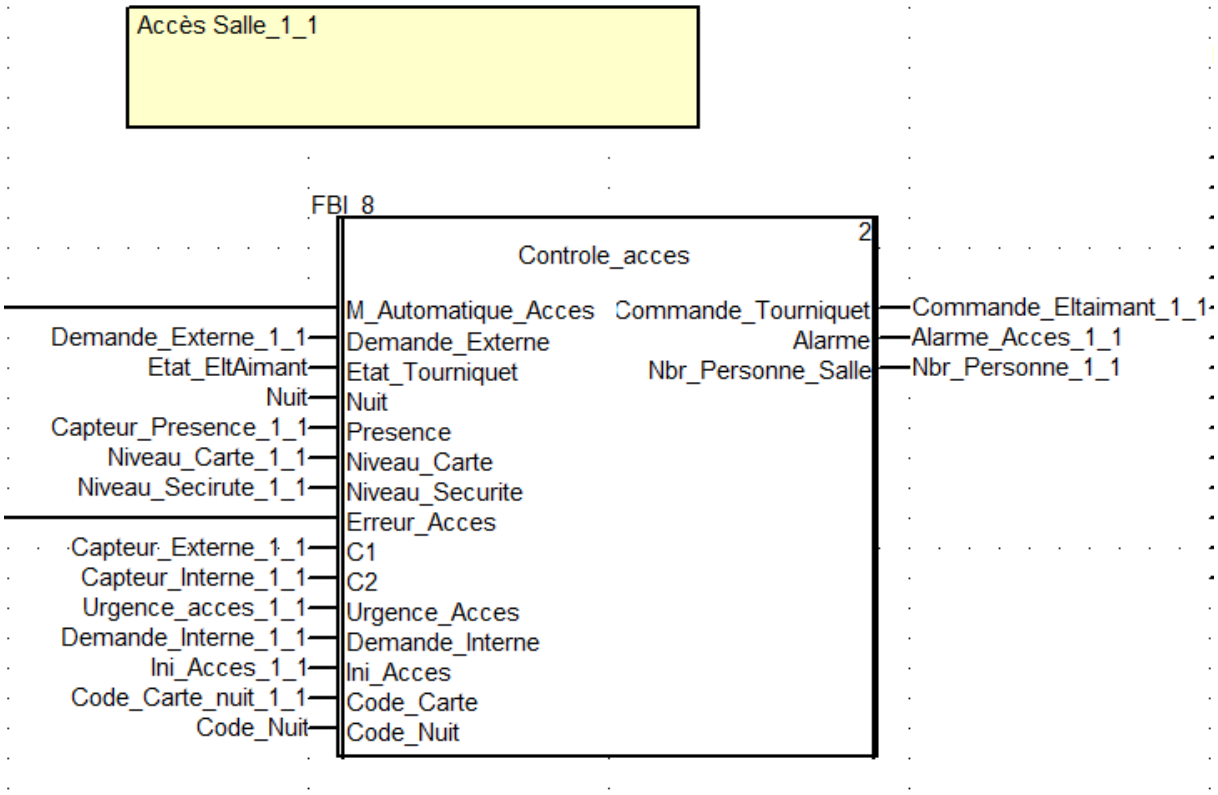


Figure IV.29 : Le bloc DFB Control d'accès

Le bloc DFB « contrôle_accès » comporte en entrée toutes les informations qui proviennent des différents capteurs.

Principalement il va lire :

- Le niveau de sécurité de la salle (qui provient du superviseur)
- Le niveau de la carte de la carte (qui provient du lecteur carte)

Ensuite il va combiner entre ces données pour décider si oui non l'électroaimant sera débloqué

Les détails et les programme concernant le contrôle d'accès se trouve en **Annexe[D].**

5. Gestion de parking :

La gestion de parking permet de contrôler en temps réel les entrées et les sorties des voitures au parking et comme il est mauvais pour les plantes que les voitures stationnent vers l'arrière il est possible de mettre un détecteur de CO2 afin d'avertir le superviseur en cas de non respect de cette règle, avec la gestion de parking nous avons non seulement un contrôle en temps réel des places restantes dans le parking mais en plus nous protégeons la nature en ne laissant personne stationner vers l'arrière et ainsi tuer les plantes.

5.1. Cahier de charge :

Nous avons un parking séparé en deux parties l'une pour les visiteurs et l'autre pour les employés, avec donc deux niveaux d'accès différents, pour accéder au parking réservé aux employés il faut passer sa carte avec un niveau d'accès au moins égale à celui d'un employé, nous avons un capteur de CO2 à chaque emplacement du parking pour nous avertir en cas de stationnement non-conforme aux règles, et nous avons aussi un détecteur de présence à chaque place pour nous avertir que la place est prise et dans le cas où le parking est plein nous avertir et bloquer l'accès à d'autres personnes, jusqu'à ce qu'une place se libère.

Le parking extérieur est composé de 4 places et le parking intérieur en a 16.

Lorsqu'une personne arrive est qu'elle passe sa carte devant le lecteur de carte, il y a donc deux niveaux d'accès, s'il a accès au parking réservé aux invités un voyant vert s'allume devant le premier portail lui indiquant qu'il a le droit d'y accéder, une temporisation se lance donc s'il reste à cheval entre l'entrée et la sortie et que la porte ne peut pas se refermer, une alarme est enclenchée afin de prévenir qu'il y a un problème, s'il a l'accès au deuxième niveau du parking les voyants devant les deux portails deviennent verts.

Nous avons deux moteurs pour commander les portails qui pourront tourner dans les deux sens ou alors s'arrêter.

Nous avons aussi le cas de nuit, où c'est le même processus sauf que l'accès n'est réservé qu'à certaines personnes dont le code de la carte a été préalablement enregistré, l'accès au bâtiment étant prohibé sauf dans le cas à il est signalé à l'avance et que le code de la personne est préalablement enregistré.

5.2. Le matériel utilisé :

Pour faire la gestion de parking nous avons utilisé des détecteurs de métaux qui nous préviennent à tout moment la présence ou non d'une voiture garée, nous avons pris aussi un détecteur de CO2 qui envoie un signal dès qu'un certain seuil de CO2 est dépassé preuve qu'une voiture est en train de se garer dans le mauvais sens, nous avons aussi deux moteurs pour commander l'ouverture et la fermeture du portail, des détecteurs de passage qui envoient un faisceau lumineux qui lorsqu'il est coupé envoie un signal, afin d'avertir si les voitures restent à travers la porte et ne la laissent pas se fermer.

5.3. Programmation :

On a créé un bloc fonction pour la commande des moteurs et des voyants, il a en entrée les données nécessaire pour traiter les cas et pour savoir s'il s'agit du parking 1 ou 2 et en sortie la commande en sens direct, ou sens indirect du moteur ainsi que des alarmes qui nous préviennent en cas de problème.

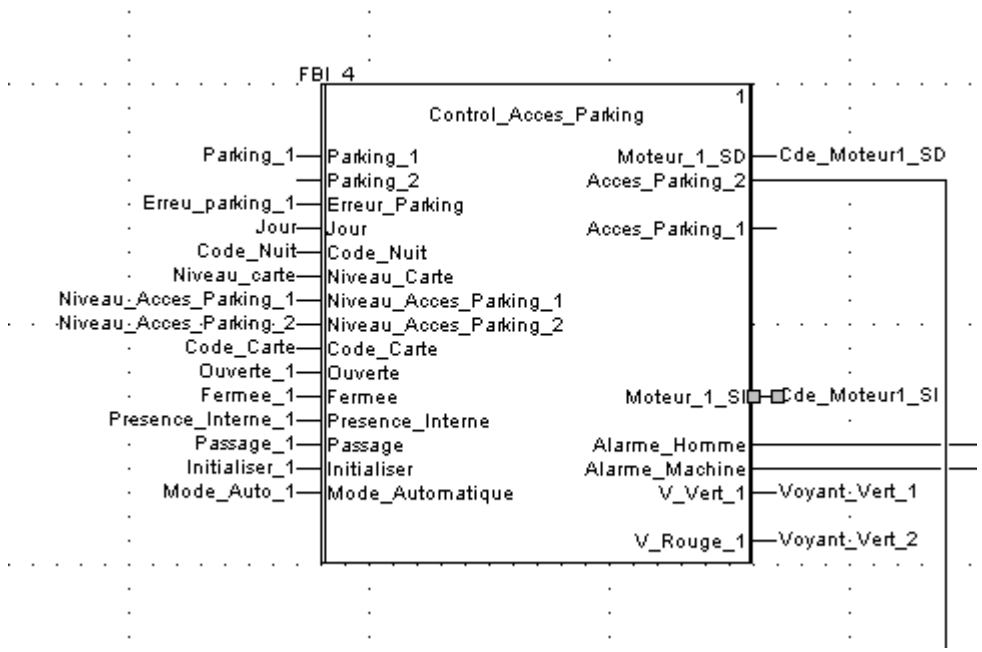


Figure IV.30 : Bloc contrôle d'accès au parking

En entrée : Nous aurons les informations recueillies par les capteurs et le lecteur de carte ainsi qu'une entrée erreur du parking qui bloque le processus s'il y a une erreur dans le parking.

- Erreur_parking.
- Niveau Acces parking 1 et 2.
- Code d'accès Nuit.
- Niveau de la carte recueillie par le lecteur de carte.
- Porte Ouvert/Fermée.
- Mode Automatique ou pas.

En sortie : Nous aurons la commande des moteurs, les alarmes et les voyants.

- Commande des moteurs en Sens Direct et Sens Indirect
- Les alarmes.
- La commande du voyant de chaque portail.

Nous avons aussi créé un bloc qui a comme entrées les informations recueillies par les capteurs, en vecteur de vingt positions et le seuil permis de Co2 et en sorties nous sort un vecteur nous indiquant les positions du parking occupées ou libres, le nombre de voitures dans

chaque parking, et le nombre de voitures mal garées dans chaque parking, nous avons aussi en sortie un vecteur de temps qui nous donne à tout moment le temps qu'est restée une personne garée dans le parking.

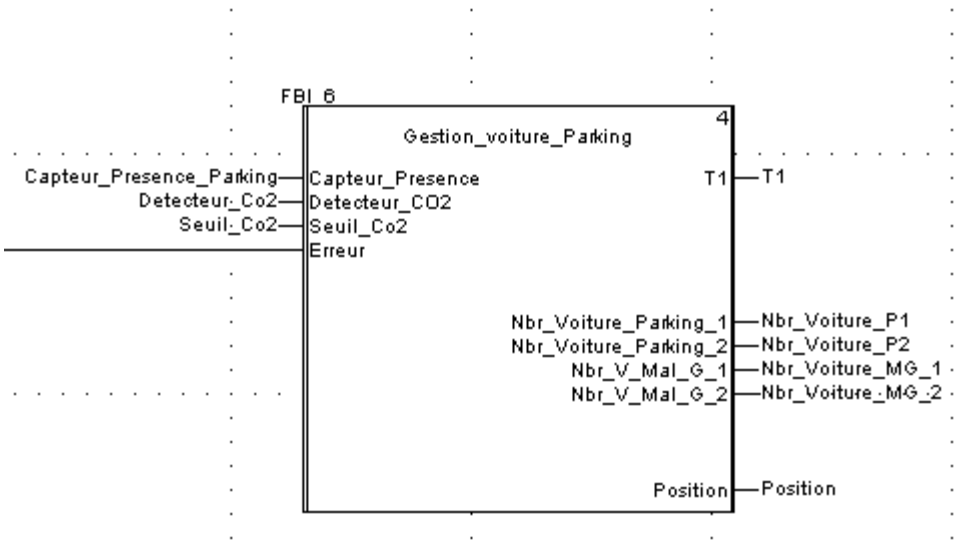


Figure IV.31: Bloc d'informations sur le parking

En entrée : Nous avons les informations recueillies par les capteurs de présence et de Co2 en vecteur de vingt lignes ainsi que le seuil de Co2 permis qu'on peut mettre à la valeur qu'on veut, une sortie du bloc d'erreurs entre aussi dans ce bloc pour bloquer le processus en cas d'erreur.

- Capteur de présence en vecteur pour toutes les positions.
- Détecteur de Co2 de chaque position en vecteur aussi.
- Le seuil de Co2 permis.
- Les erreurs.

En sortie : Nous aurons le temps qu'est restée chaque voiture à sa position actuelle, le nombre de voitures dans chaque parking, le nombre de voitures mal garées dans chaque parking, ainsi qu'un vecteur position nous donnant à tout moment les positions occupées et les positions libres.

- Le vecteur T1 qui donne le temps resté à toutes les positions occupées.
- Le nombre de voitures dans chaque parking.
- Le nombre de voiture mal garées dans chaque parking.
- Le vecteur position qui donne les positions occupées et libres.

5.4. Supervision :

Dans la partie supervision nous avons créé une page qui nous donnes le récapitulatif de l'état du parking en temps réel, il nous donne les informations de sorties des deux blocs afin que le superviseur puisse d'un coup d'œil avoir une information générale sur le parking

6. Gestion d'électricité :

La gestion d'électricité permet de commander l'alimentation en électricité de tous les secteurs du bâtiment ainsi que l'acquisition à tout moment de toutes les informations électriques de tous ces secteurs, dans le but de repérer d'éventuelles anomalies.

6.1. Cahier de charge :

Nous avons une entrée principale qui arrive de la compagnie d'électricité ainsi qu'un générateur électrique en cas de panne à l'arrivée, les disjoncteurs d'arrivée et du générateur sont inversés afin qu'il n'y ait qu'une seule source d'énergie à la fois.

Dans notre application nous avons programmé des blocs d'acquisition des informations depuis la Sepam S80 et les PM710 ainsi qu'un programme pour commander les disjoncteurs qui distribuent l'électricité à tous les secteurs du bâtiment, nous avons ensuite créé une page de supervision qui nous permet de voir à tout moment s'il y a un problème matériel et afin de lire la consommation et les informations électriques générales, courants, tensions, puissances et cos(φ).

Nous avons aussi programmé un bloc pour que le disjoncteur général d'un étage s'ouvre en cas de problème majeur dans cet étage.

6.2. Le matériel utilisé :

Pour faire la gestion d'électricité nous avons utilisé des PM710 qui permettent d'avoir les informations électriques nécessaires ainsi qu'une Sepam S80 qui non seulement permet de faire ça mais en plus permet de protéger notre réseau électrique et de couper le disjoncteur. Nous utiliserons pour chaque départ vers les étages un disjoncteur commandable afin de pouvoir à tout moment arrêter la distribution en électricité de l'étage en question, en cas d'incident majeur, par exemple en cas d'incendie.

6.3. Programmation :

On a créé un bloc fonction pour l'acquisition des données électriques qui va utiliser des read_var pour lire les informations en modbus à partir du matériel.

En sortie : nous aurons toutes les informations électriques voulues en un vecteur donc on prendra les informations voulues à chaque fois.

- Les courants
- Les tensions
- Les puissances
- Le Cos(φ)

Nous avons aussi créé une commande de tous les disjoncteurs.

6.4. Supervision :

Dans la partie supervision nous avons créé des génies et super génies des Sepam S80 et des PM710 afin d'afficher à tout moment toutes les sorties du bloc programmé, dans le super génie de la Sepam S80 nous avons aussi intégré une fonction pour commander le disjoncteur d'entrée, en cliquant sur le matériel nous avons un pop up qui apparaît donnant toutes les informations voulues. Un carré rouge entoure le matériel en cas d'erreur. Sur la figure suivante nous voyons un cas où la Sepam S80 et les PM710 des départs vers les étages 1 et technique ont des erreurs, tandis que la PM710 du départ vers l'étage 6 est parfaitement en marche.

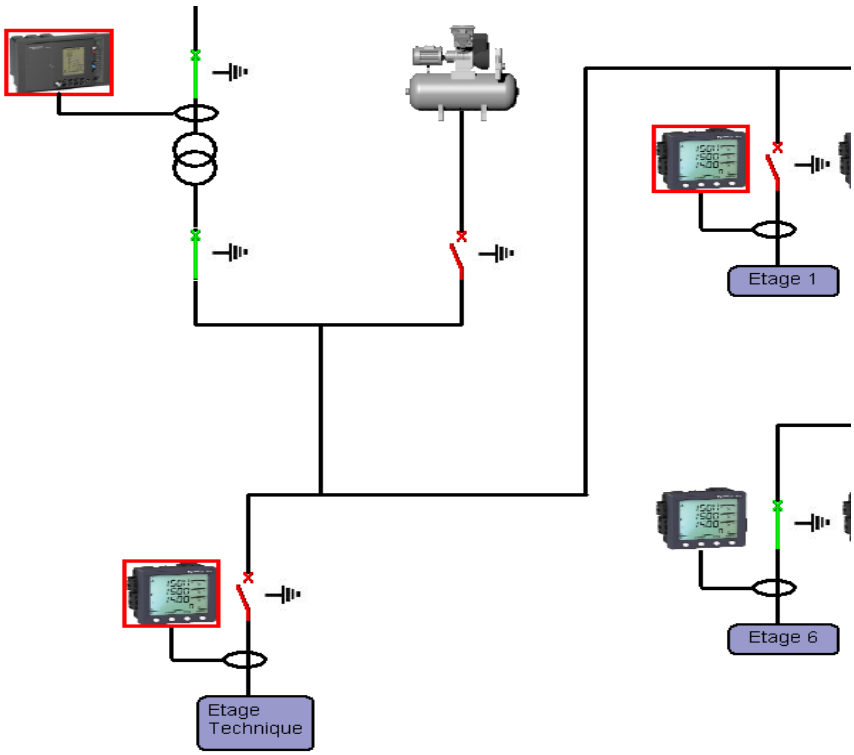


Figure IV.32 : Réseau électrique supervisé

Dans le Pop Up des PM70 nous pouvons choisir les variables à afficher, les courants, les tensions, les puissances ou bien le Cos(φ), en cliquant sur le bouton correspondant.

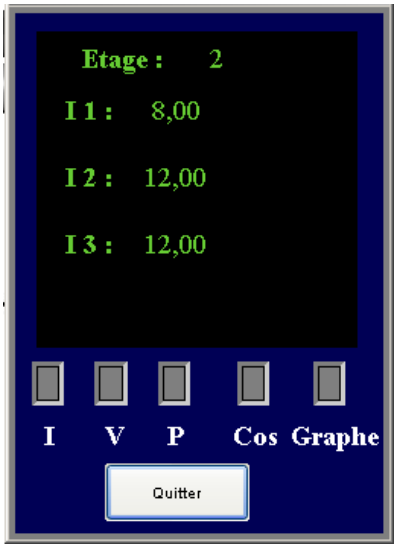


Figure IV.33 : Pop Up de la PM710

Dans le Pop Up de la Sepam S80 nous avons les mêmes informations que pour la PM710 et nous avons en plus la possibilité de commander le disjoncteur d'entrée principale grâce au bouton ouvrir et fermer, qui change d'apparence selon le cas présent.

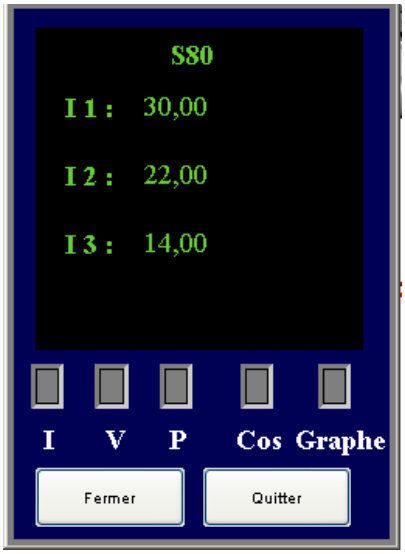


Figure IV.34 : Pop Up de la Sepam S80

7. Les pages de supervision :

Dans cette partie on a essayé de se rapprocher au maximum a des image réelles, pour cela, on s'est basé d'une part sur des photos prise du siège Schneider Electric et on les a ensuite importées dans Vijeo Citect, et d'autre part on a créer des vues en 3D à l'aide du logiciel Google Sketchup7 pour avoir une interface complète et facile a utiliser.

Les figures suivantes présentent les écrans créés pour la supervision des différentes parties du bâtiment :

- Le synoptique qui nous donne des informations globales concernant le bâtiment (la température, les erreurs éventuelles, …)
- La page étage qui nous donne une vue de l'ensemble des salles de l'étage, donnant accès à différentes données relatives à l'étage (niveau d'accès, les erreurs, …)
- La page salle qui permet la commande et la supervision des différents types de bureaux se trouvant dans l'étage.
- La page réseau électrique pour les informations électriques des différentes parties du bâtiment.
- La page réseau de communication donne une vue d'ensemble de l'intercommunication de ce système.
- La page parking qui nous donne les informations sur la gestion du parking.

Figure IV.35 : La page Main

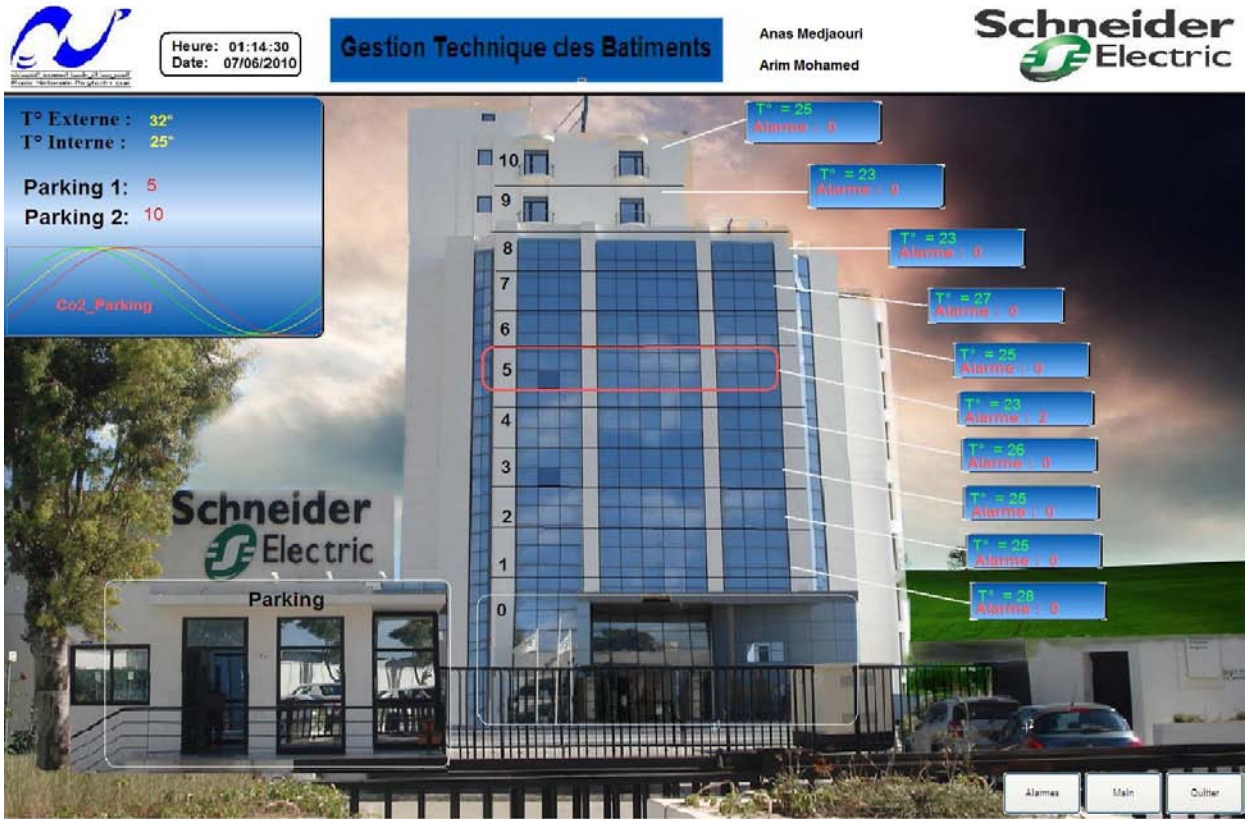


Figure IV.36 : Synoptique Générale Du Bâtiment

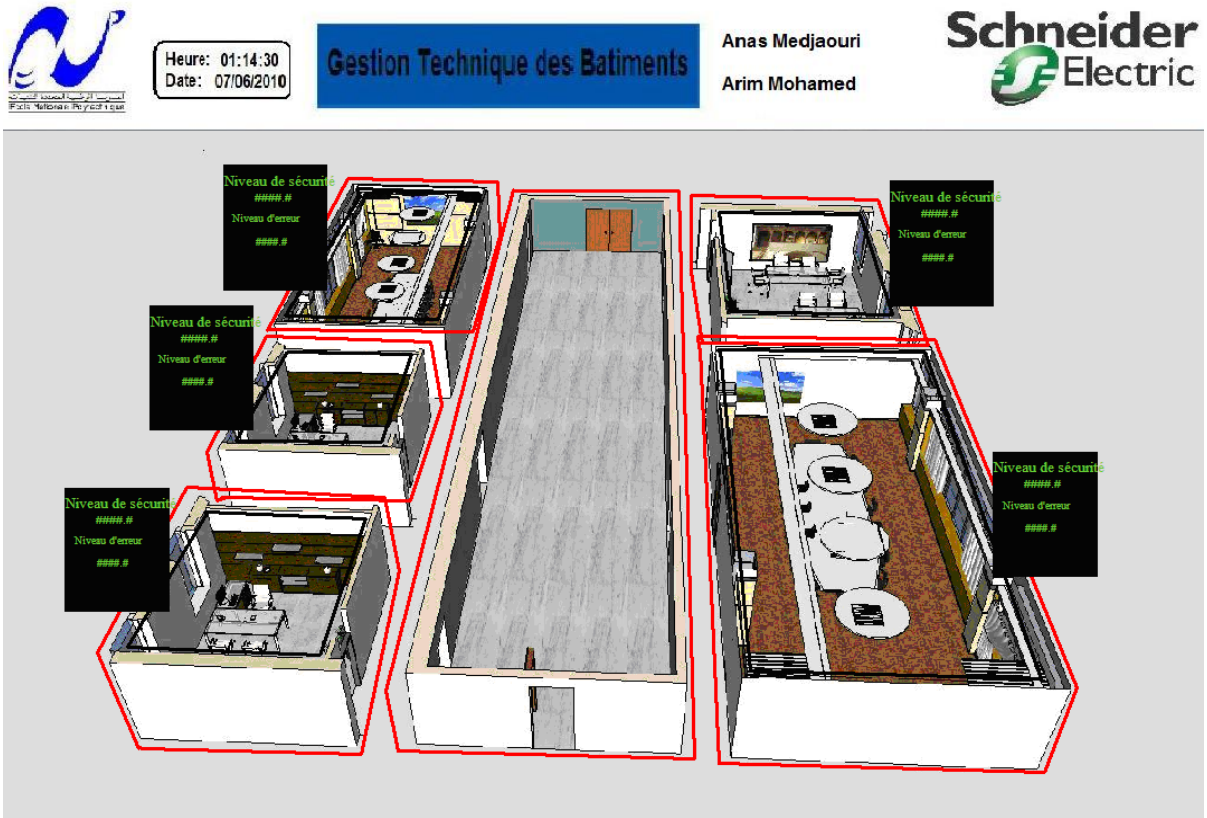


Figure IV.37: La Supervision de l'étage

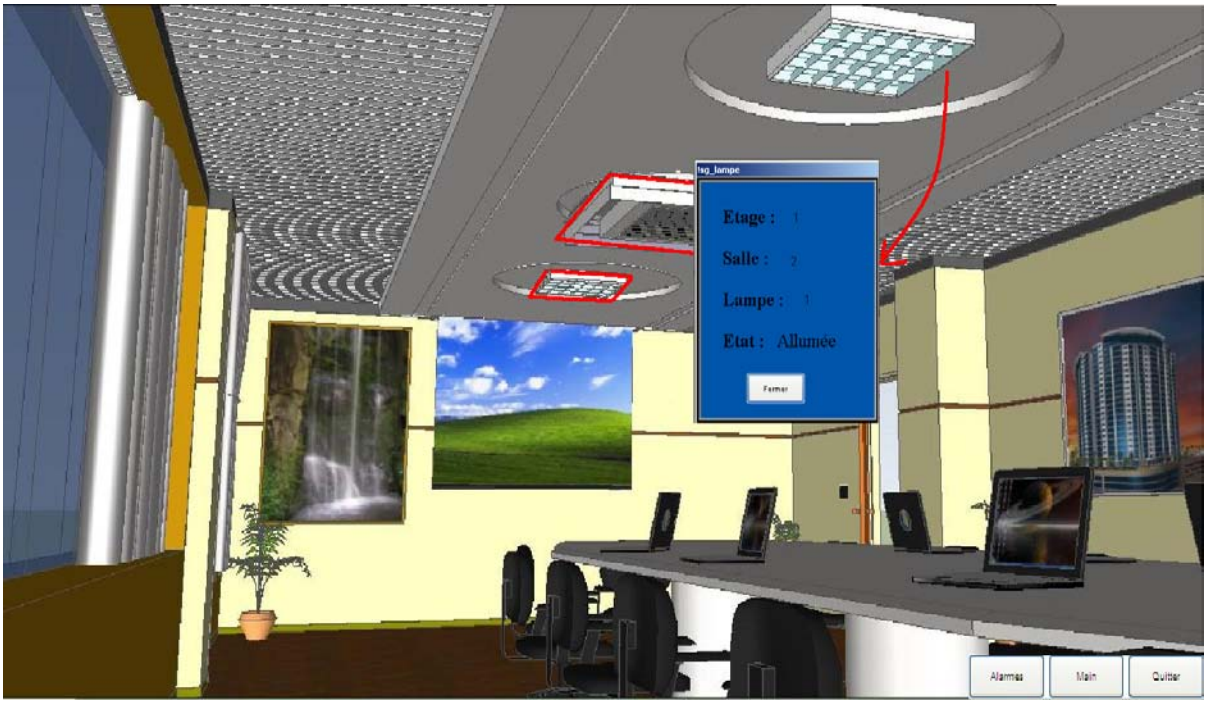


Figure IV.38: La supervision de la salle

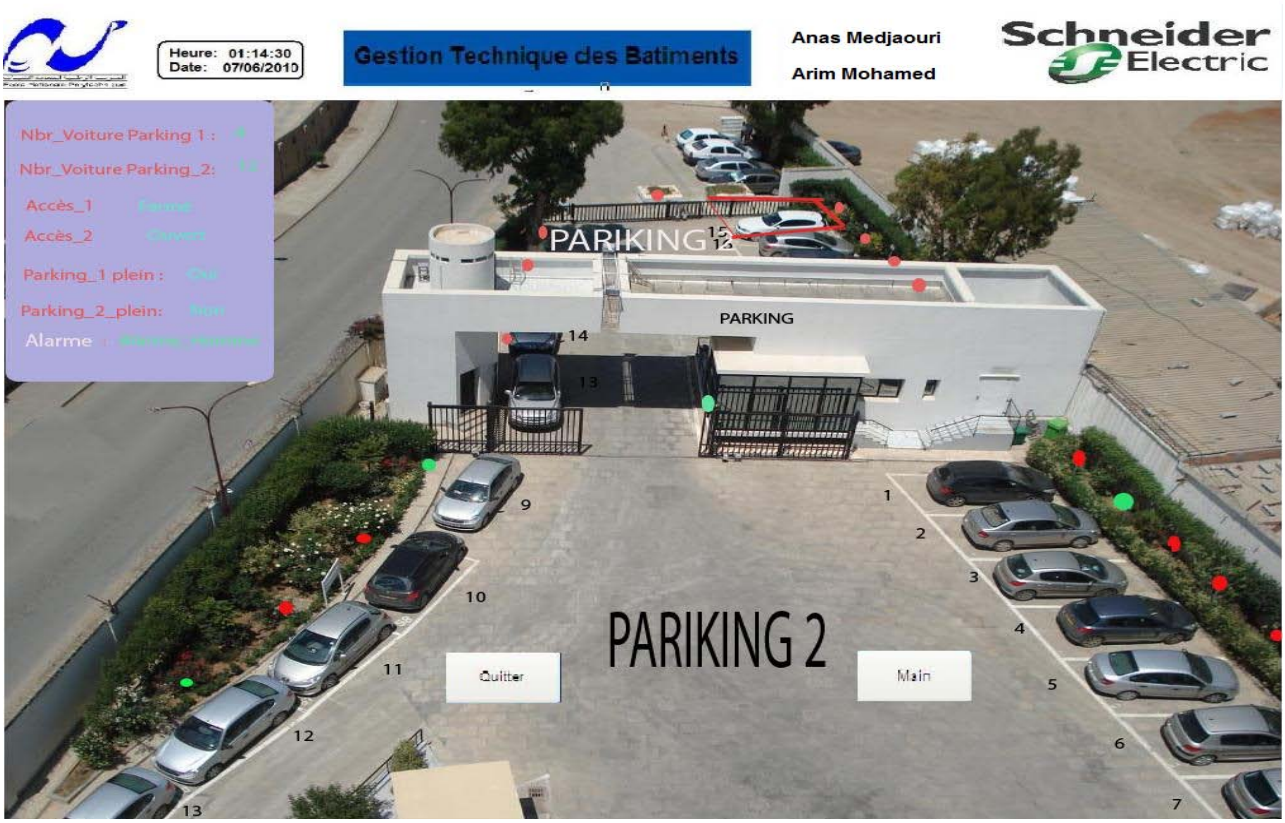


Figure IV.39 : La supervision du parking

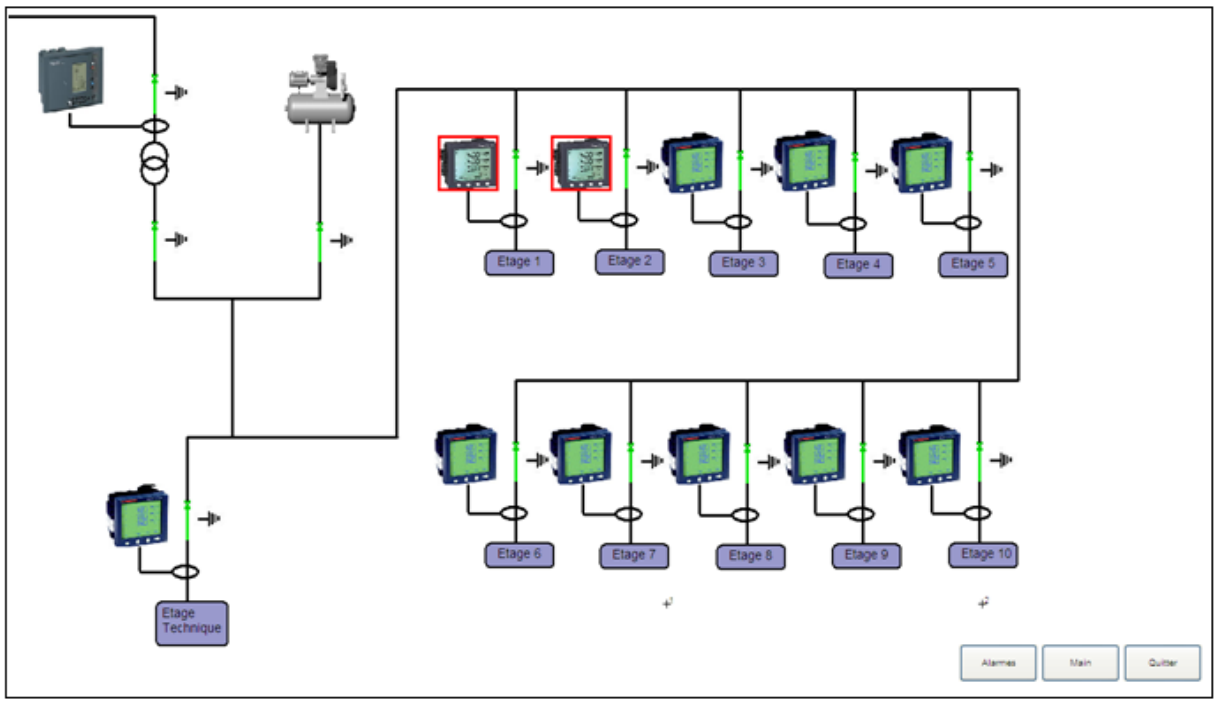


Figure IV.40: La supervision du réseau électrique

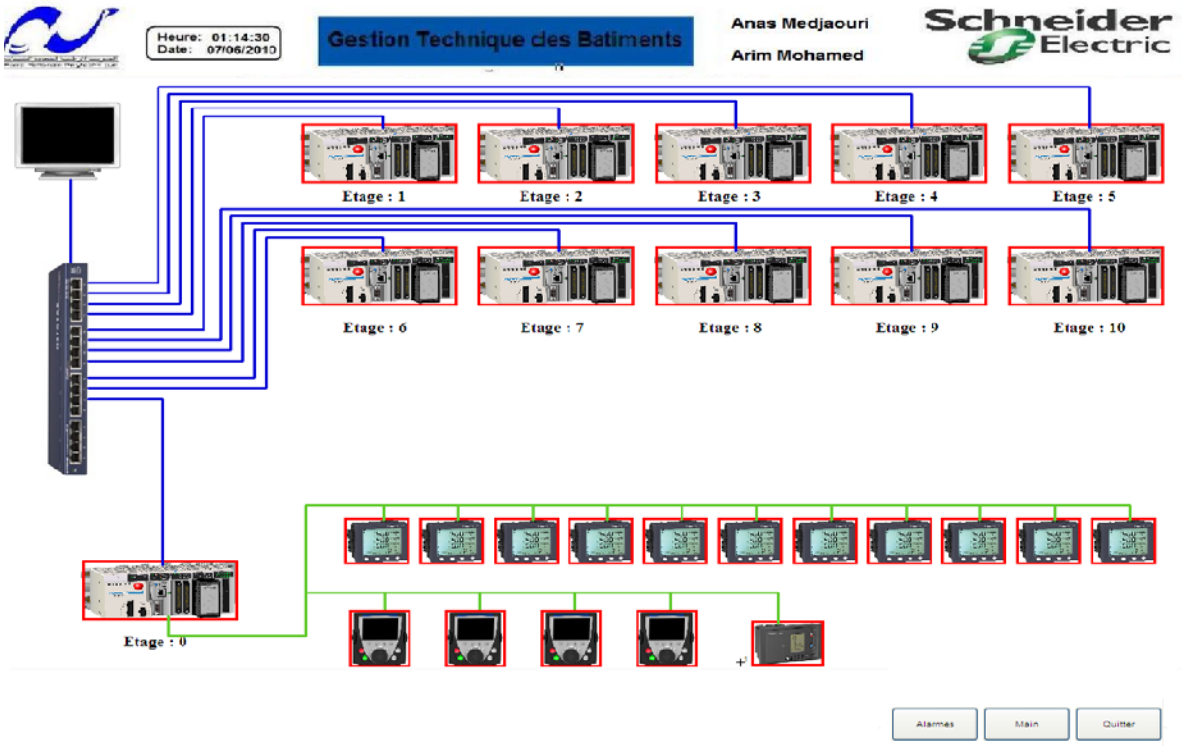


Figure IV.41: La supervision du réseau de communication

Conclusion :

L'éclairage consomme environ 40% de l'énergie dans le bâtiment Schneider Electric, la gestion technique qu'on a réalisée peut réduire cette consommation avec des solutions de contrôle de supervision appliquées sur les différents secteurs du bâtiment, ainsi elle permet de baisser des coûts d'installation et d'exploitation, améliorer la souplesse d'utilisation dans des différents espace de travail ,et assurer le confort des employé.

La gestion de climatisation que nous avons réalisée permet d'allumer en toute sécurité les ventilateurs et de commander leur vitesse de rotation, et elle permet aussi d'assurer un confort dans chaque salle en commandant le débit d'air conditionné selon la différence entre la consigne et la température actuelle de la salle.

La gestion d'accès qu'on a réalisé permet de contrôler et donner un certain ordre de priorité dans l'accès aux différents secteurs du bâtiment.

La gestion de parking que nous avons effectuée permet non seulement la gestion autonome du parking et de l'ouverture et fermeture des portails mais en plus elle permet la protection de l'environnement grâce aux capteurs de Co2 présents à coté des plantes.

La gestion d'électricité que nous avons réalisée permet de repérer en temps réel les erreurs qui peuvent survenir dans le réseau électrique, et elle augmente la sécurité de l'installation électrique tout en offrant une interface simple d'utilisation qui permet qui permet d'avoir un œil en temps réel sur les variables électriques du réseau.

CONCLUSION GENERALE

Conclusion générale :

Munis de deux logiciels très performants, Unity Pro et Vijeo Citect, les automates programmables industriels Schneider forment des unités de traitement et de commande de grande flexibilité.

En effet, le logiciel de programmation Unity Pro permet l'accès de base aux automates programmables de la gamme Modicon de Schneider, pour sa programmation en différents langages. Il assure également la fonction de moyen de communication en prenant en compte leurs réseaux. Le logiciel de conception des interfaces homme-machine Vijeo Citect est quant à lui, un logiciel d'ingénierie et de supervision, qui offre des fonctions de surveillance d'automatismes.

Notre projet nous a permis d'explorer un domaine innovant en plein essor qu'est la gestion technique de bâtiment et de se familiariser avec les outils de programmation d'API et de supervision.

Notre contribution s'est portée sur la programmation de gestion, de l'éclairage, de l'électricité, de la climatisation et du contrôle d'accès ainsi que la réalisation d'une interface graphique permettant la supervision du bâtiment en temps réel.

Il serait intéressant d'orienter les futurs projets de développement des bâtiments intelligents vers des axes traitant d'une part la facilité de renouveler l'énergie utilisée ,et d'autre part les différentes technologies de diminution des nuisances de ces bâtiment sur l'environnement, et tout cela dans le but de se rapprocher au maximum de ce qu'on appelle, un bâtiment intelligent bioclimatique.

Bibliographie :

[1] Intelligent Building Systems (The International Series on Asian Studies in Computer and Information Science),
Edition Chuck Ehrlich

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

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

[4] Télé catalogue Schneider Electric.

[5] Modicon M340 sur Unity Pro (help).

[6] Help du logiciel Unity Pro.

[7] Help du logiciel Vijeo Citect

[8] Formation Unity Pro (Schneider Electric).

[9] Formation Vijeo Citect (Schneider Electric).

[10] Formation sur les Protocols de Communication

ملخص :
العمل المنجز في هذه المذكرة يتمحور أساسا حول استخدام برمجة شنيدر لإدارة عمارة ذكية ودلك باستخدام برنامج" يونيتي برو" . وتصميم شاشة إشراف من خلالها نستطيع مراقبة جميع أجهزة العمارة باستخدام برنامج "فيجيو سايتيكت".
العمارة ا الذكية المدروسة هي عمارة شنيدر في نهاية شراقة بالجزائر. نوع الآلي المبرمج المستخدم هو **M340.**

الكلمات الرئيسية:
مسيرصناعي مبرمج شنيدر ,العمارة ا الذكية ,برنامج يونيتي برو ,برنامج فيجيو سايتيكت، آلي البرمجة **M340.**

RESUME :
Le travail présent dans ce mémoire est basé essentiellement sur l'utilisation des automates programmables Schneider. Notre travail est la programmation de la commande et la supervision d'un bâtiment intelligent à l'aide de logiciel Unity Pro par l'utilisation des DFB et la supervision à l'aide de logiciel Vijeo Citect. Le bâtiment étudié est le siège de SCHNEIDER Electric situé à Cheraga à Alger. Sa gestion est assures par un automate M340.

Mots clés : automates programmables Schneider, Commande et supervision d'un bâtiment intelligent, logiciel Unity Pro, DFB, logiciel Vijeo Citect, automate M340, PM710, S80, ATV

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 an intelligent building with Unity Pro software through the use of DFB and supervision with Vijeo Citect software.
The intelligent building studied is Schneider Electric located in Cheraga Algiers. Its management is provided by a PLC M340.

Key word: Programmable logic controller Schneider, water pumping station, Unity Pro software, DFB, Vijeo Citect software, PLC M340.

REPUPLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

Schneider Electric

ECOLE NATIONALE POLYTECHNIQUE D'ALGER

Département de Génie Electrique

Spécialité : Automatique

Projet de fin d'études en vue de l'obtention du Diplôme d'Ingénieur d'Etat en Automatique

Commande et supervision du bâtiment intelligent Schneider Electric (Annexe)

Réalisé par :

Mr MEDJAOURI Anas

Mr ARIM Mohamed

Proposé et dirigé par :

Pr E.M.BERKOUK

Mr A.YOUNSI

Juin 2010

Ecole Nationale Supérieure Polytechnique, 10 Av. Hassan Badi, El Harrach, Alger, Algérie

TABLE DES MATIERES

ANNEXE A

LES A.P.I

Introduction :

Tous les secteurs de l'industrie, de la production d'électricité à la peinture pour automobiles, l'emballage alimentaire utilisent les automates programmables industriels pour améliorer la production. Dans ce chapitre, nous présenterons les différents aspects de ces outils puissants et polyvalents. Nous nous intéresserons, également aux solutions Schneider pour l'automatisation de l'industrie en général, et à l'automate programmable industriel Modicon M340 en particulier.

1. Place des API dans les systèmes automatisés de production :

Le système automatisé de production (SAP) est une notion assez large qui inclut des systèmes de contrôle, de conditionnement, et d'analyse. Ce dernier reçoit un flux de matières ou de produits et génère un flux de produits plus élaborés, de plus il doit gérer l'alimentation en énergie ainsi que des flux auxiliaires. Tout cela, ajouté à des exigences sans cesse accrues de qualité, sécurité, flexibilité, entraîne un accroissement des besoins, en particulier la manipulation d'un grand nombre de variables et la gestion de véritables flux de communication.

Ainsi, les systèmes câblés devenant trop volumineux et trop rigides pour de telles applications, explique que l'on se tourne donc vers des solutions utilisant les techniques de traitement de l'information par processeurs programmables.

La solution reposant sur un processeur central unique s'étant vite révélée peu économique du point de vue du câblage et complexe quant à la maintenance voire dangereuse en cas d'incident, l'utilisation de **processeurs spécialisés et interconnectés** s'est aujourd'hui largement imposée.

L'architecture décentralisée qui en résulte facilite la conception et l'installation en permettant de fractionner les études, la mise en place, les tests ; elle améliore aussi la maintenance (modification aisée des programmes, de parties du système automatisé) et se traduit par plus de flexibilité et de disponibilité. Elle entraîne toutefois, du fait des multiples sous-ensembles fonctionnels, un fort accroissement des besoins de communication et de gestion.

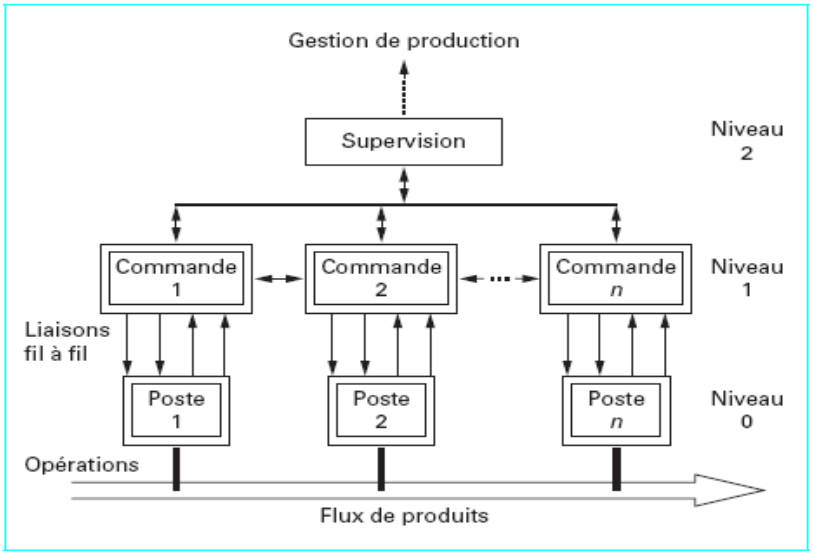


Figure A.1: Système automatisé de production

Dans ces systèmes de traitement de l'information, les API occupent une place de choix. Les équipements notés « commande » sont souvent des automates. Ils constituent de plus en plus un maillon fiable et efficace entre le calculateur et l'appareillage de terrain.

2. Définitions générales :

Il nous faut définir l'API de manière très précise tout en sachant que dans ce domaine et dans beaucoup d'autres touchant à l'informatique, les frontières sont floues et changeantes.

Un système est considéré comme automate programmable si :

- Il est construit autour d'un processeur numérique ;
- Il peut être relié à plusieurs signaux physiques ;
- Il fonctionne grâce à une protection adaptée aux milieux industriels ;
- Il est doté d'un logiciel de programmation ;
- Il est doté de possibilité d'échanges avec d'autres processeurs.

Ceci amène à la définition assez lourde de la norme française EN 61131-1 :
« Système électronique fonctionnant de manière numérique, destiné à être utilisé dans un environnement industriel, qui utilise une mémoire programmable pour le stockage interne des instructions orientées utilisateur aux fins de mise en oeuvre de fonctions spécifiques, telles que des fonctions de logique, de mise en séquence, de temporisation, de comptage et de calcul arithmétique, pour commander au moyen d'entrées et de sorties Tout-ou-Rien ou analogiques divers types de machines ou de processus. L'automate programmable et ses périphériques associés sont conçus pour pouvoir facilement s'intégrer à un système d'automatisme industriel et être facilement utilisés dans toutes leurs fonctions prévues. »

3. Architecture des automates :

3.1. Aspect extérieur :

Les automates peuvent être de type **compact** ou **modulaire.**

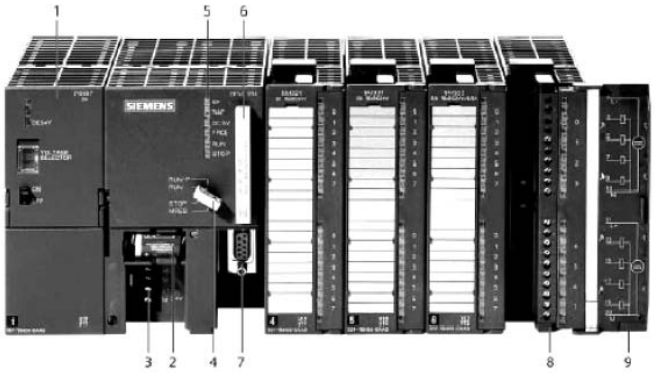
De type compact, on distinguera les modules de programmation (LOGO de Siemens, ZELIO de Schneider, MILLENIUM de Crouzet ...) des microautomates. Il intègre le processeur, l'alimentation, les entrées et les sorties. Selon les modèles et les fabricants, il pourra réaliser certaines fonctions supplémentaires (comptage rapide, E/S analogiques ...) et recevoir des extensions en nombre limité. Ces automates, de fonctionnement simple, sont généralement destinés à la commande de petits automatismes.

De type modulaire, le processeur, l'alimentation et les interfaces d'entrées / sorties résident dans des unités séparées (**modules**) et sont fixées sur un ou plusieurs **racks** contenant le "fond de panier" (bus plus connecteurs).

Ces automates sont intégrés dans les automatismes complexes où puissance, capacité de traitement et flexibilité sont nécessaires.

Automate compact (Allen-bradley)

Automate modulaire (Modicon)



Automate modulaire (Siemens)

Figure A.2 : Exemples d'automates

1 Module d'alimentation
2 Pile de sauvegarde
3 Connexion au 24V cc
4 Commutateur de mode (à clé)
5 LED de signalisation d'état et de défauts
6 Carte mémoire
7 Interface multipoint (MPI)
8 Connecteur frontal
9 Volet en face avant

3.2. Structure interne :

En général un automate programmable se constitue essentiellement d'une unité centrale, un module d'entrées/sorties, un module d'alimentation, un module de stockage et de liaisons et des auxiliaires.

Cette architecture et représentée dans la figure suivante :

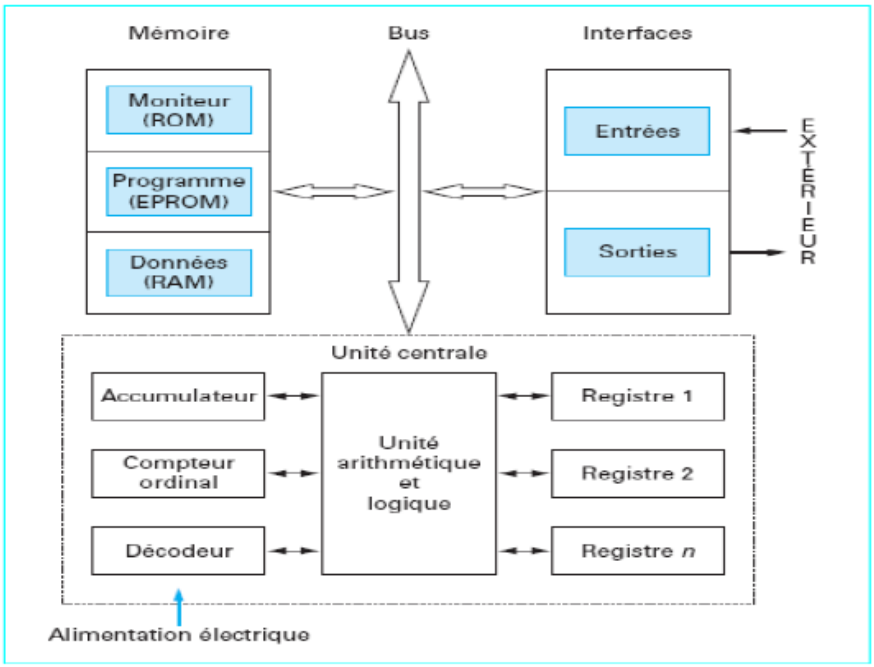


Figure A.3: Architecture interne d'un API

3.2.1. Module d'alimentation :

Ce module permet l'alimentation en tension continue nécessaire au bon fonctionnement de l'automate programmable ainsi que le circuit de charge. Il convertit la tension du réseau (AC 220) en tension de service (DC 24V, 15V ou 5V). Ce module doit posséder de bonnes performances face aux microcoupures du réseau, ainsi qu'un transformateur d'isolement pour lutter contre les perturbations du même réseau.

3.2.2. Unité centrale :

L'unité centrale (CPU) est l'élément le plus important dans l'automate programmable, elle peut être considérée comme le cerveau du système. Elle est constituée de deux composants principaux:

- Le Processeur.
- Les mémoires.

3.2.2.1 Le processeur :

La principale fonction du processeur est de commander et gouverner les différentes activités du système. Il effectue cette tache en interprétant et en exécutant un ensemble de programmes système. Ces derniers forment un groupe de programmes superviseurs stockés de façon permanente dans le processeur. Grâce à ces programmes superviseurs le processeur peut ainsi exécuter toutes ses taches de contrôle, ainsi que divers fonctions domestiques.

Ces programmes appelés aussi « **le pouvoir exécutif »** assurent la communication entre l'API et l'utilisateur par le biais de dispositifs de programmation. Ils supportent aussi d'autres périphériques de communication tels que la surveillance des appareils de terrain, la lecture des

données de diagnostic, l'alimentation, les modules d'entrées/sortie, les mémoires, et la communication avec les interfaces opérateurs.

3.2.2.2. Les mémoires :

Tout système bâti autour d'un processeur possède un ou plusieurs types de mémoires. La mémoire système dans un API est composée de deux majeures parties :

- La mémoire exécutive : assure le stockage des programmes superviseurs.
- La mémoire d'application : est une zone de stockage dédiée aux programmes d'instructions utilisateur.

Les exigences de stockage et de récupération pour les programmes superviseurs et les programmes d'application ne sont pas les mêmes, par conséquent ils ne sont pas toujours stockés dans le même type mémoire. Ainsi on aura l'organisation suivante :

- ROM ou PROM : Ce sont des mémoires mortes dont l'utilisateur ne peut que lire le contenu (ROM) et éventuellement les programmer à l'aide d'outils spéciaux (PROM). On y retrouve dans notre cas les programmes superviseurs.
- EPROM : C'est une mémoire reprogrammable qui permet de stocker les programmes mis au point et utilisables.
- RAM : C'est une mémoire vive (volatile) secourue en général par une batterie, elle stocke les données système lors du fonctionnement.

3.2.3. Modules d'entrées/sorties (E/S) :

Les modules d'E/S assurent le rôle d'interface de la partie commande, ils se situent entre la CPU et le processus.

Pour ce faire ils doivent :

- Regrouper les variables de même nature pour diminuer la complexité et le coût ;
- Assurer le dialogue avec la CPU ;
- Traduire les signaux industriels en information API et inversement.

Plusieurs types de modules sont disponibles sur les marchés comme :

- Modules d'E/S tout ou rien (TOR) : Ces modules traitent une information qui ne peut prendre que deux états (vrai ou faux, 0 ou 1), ils constituent l'interface entre l'API et les différents capteurs et pré-actionneurs présents.
- Modules d'E/S analogique : Dans ce cas, le signal traité est analogique et prend des valeurs comprises dans une plage bien déterminée. Ces modules sont munis de convertisseur analogique/numérique pour les entrées et respectivement de convertisseur numérique/analogique.
- Module spécialisés : l'information traitée est contenue dans des mots codés sous frome binaire ou bien hexadécimale. C'est le type d'information délivrée par un ordinateur ou un module intelligent.

3.2.4 Modules de communication :

Les modules de communication comprennent les consoles et les boitiers tests.

- Les consoles : Les consoles permettent la programmation, le paramétrage et les relevés d'informations, ils peuvent également afficher le résultat de l'autotest comprenant l'état des modules d'entrées et de sorties, l'état de la mémoire, de la batterie, etc. Ils sont équipés (pour la plupart) d'un écran à cristaux liquides.
- Les boitiers tests : Les boîtiers de tests quand a eux sont destinés aux personnels d'entretien ; ils permettent de visualiser le programme ou les valeurs des paramètres tes que l'affichage de la ligne de programme à contrôler, la visualisation de l'état des entrées et des sorties…).

3.2.5 Auxiliaires :

Il s'agit principalement :

- D'un ventilateur : qui est en général indispensable dans les châssis comportant de nombreux modules ou dans le cas où la température ambiante est susceptible de devenir assez élevée (plus de 40 °C) ;
- Du support mécanique : Il peut s'agir d'un rack (structure métallique accueillant des cartes avec généralement un raccordement arrière), l'automate se présentant alors sous forme d'un ensemble de cartes, d'une armoire, d'une grille et des fixations correspondantes ;
- D'indicateurs d'état concernant la présence de tension, l'exécution du programme (mode RUN), la charge de la batterie, le bon fonctionnement des coupleurs…

4. Langages de programmation :

On retrouve cinq langages qui peuvent être utilisés pour la programmation des automates programmables industriels. Ces langages peuvent être divisés en deux catégories :

- **Langages graphiques :**
  - SFC « Sequential Funiculite Chart » ou GRAFCET.
  - LD « Ladder Diagram » ou schéma à relais.
  - FBD « Function Block Diagram » ou schéma par bloc.
- **Langages textuels :**
  - ST « structured text » ou texte structuré.
  - IL « Instruction List » ou liste d'instructions.

4.1 Les langages graphiques :

4.1.1 Le GRAFCET :

Le GRAFCET ou Graphe Fonctionnel de Commande Etape Transition est une méthode de représentation graphique permettant de décrire le cahier de charge d'un automatisme. Il est adapté aux systèmes à évolution séquentielle ; il est défini par un ensemble d'éléments graphiques de base traduisant le comportement de la partie commande vis-à-vis de ses entrées et ses sorties.

Un programme GRAFCET décrit un procédé comme une réceptivité .Celle-ci est une condition logique qui doit être vraie pour franchir la transition et passer à l'étape suivante. Des actions sont associées aux étapes du programme.

Le format graphique d'un programme GRAFCET est le suivant :

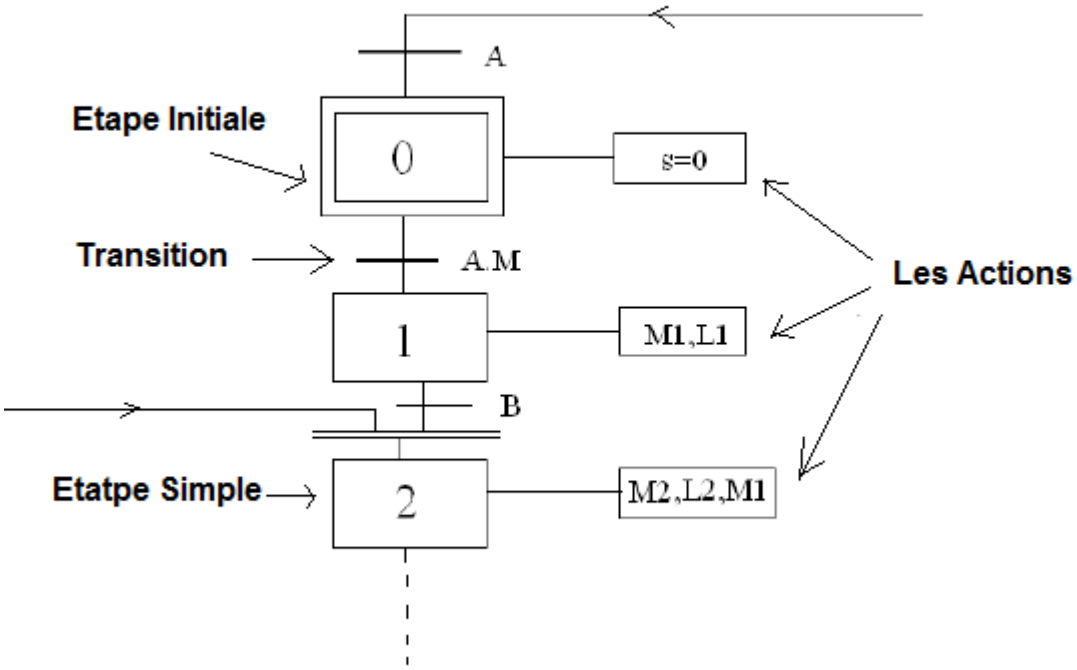


Figure A.4: Exemple de GRAFCET

Une étape représentée par un carré qui a un numéro identificateur et les actions associées sont indiquées dans un rectangle relié à la partie droite du carré (l'étape initiale est représentée par un carré double).

Une liaison orientée représentée par une ligne, parcourue par défaut de haut en bas ou de gauche à droite.

Une transition entre deux étapes et à laquelle est associée une réceptivité inscrite à sa droite, est représentée par une barre perpendiculaire aux liaisons orientées qui relient ces étapes.

4.1.2 Ladder Diagram :

Le LD est une représentation graphique qui traduit directement des équations booléennes en un circuit électrique et ce en combinant des contacts et des relais à l'aide de connexions horizontales et verticales ; les contacts représentent les entrées (contact normalement ouverts, contacts normalement fermés, …) et les relais les sorties (relais directs, relais inversés,…). Les diagrammes LD sont limités sur la gauche par une barre d'alimentation et par la masse sur la droite.

Par exemple la fonction logique : $s = a\,.\left(c + \bar{d}.\,b\right)$ est réalisée par le diagramme suivant:

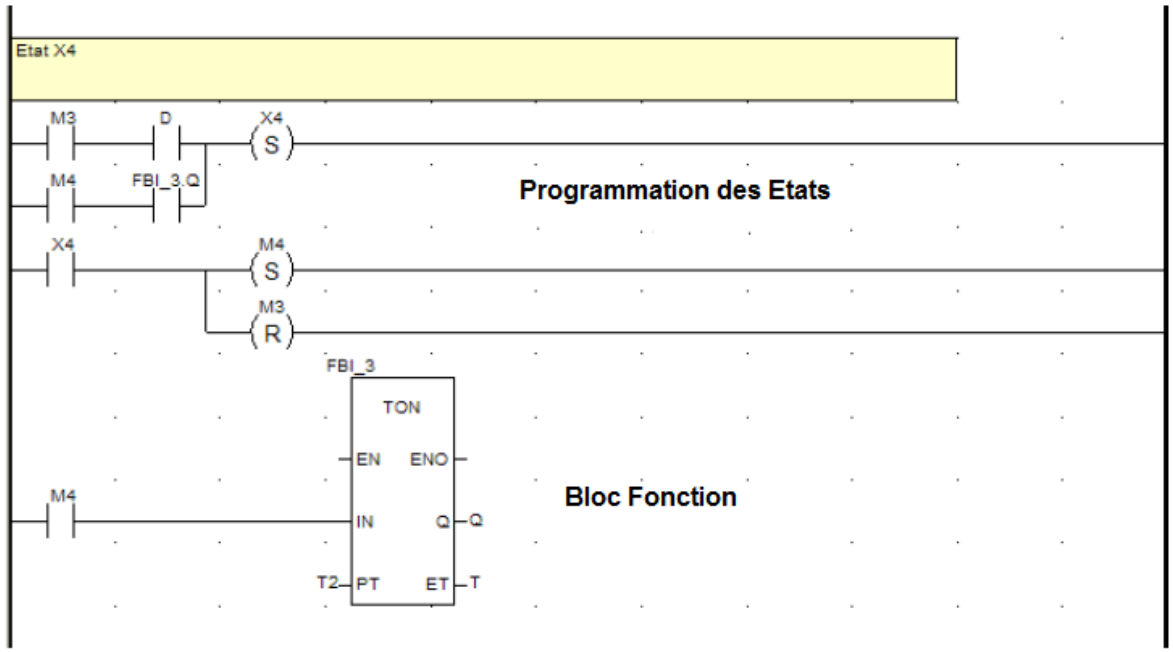


Figure A.5 : Exemple d'un programme en Ladder

4.1.3 Bloc de fonction :

C'est un langage graphique qui permet la construction d'équations complexes à partir des opérateurs standards, ou de blocs fonctionnels ; il se compose de réseaux de fonctions préprogrammées ou non, représentées par des rectangles connectés entre eux par des lignes.

La programmation avec le FBD est très souple et facile à apprendre, la plupart des fonctions nécessaires (les fonctions arithmétique et logique, les fonctions de temporisation, des blocs fonctionnels PID…) sont déjà disponibles dans la bibliothèque. Il suffit juste de les connecter et de bien paramétrer les entrées et les sorties, c'est-à-dire respecter le type des variables lors de la connexion.

Par exemple, pour réaliser la fonction arithmétique suivante: $w = 20.\frac{(x+y)}{z}$.
On aura besoin de deux blocs : un pour l'addition, un pour la multiplication et un autre pour la division.

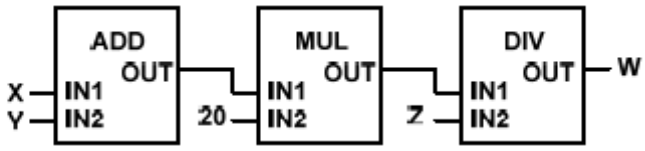


Figure A.6: Exemple d'un programme en Fonction Bloc

4.2 Les langages textuels :

4.2.1 Langage Structuré :

Le langage ST (Structured Text) est un langage de programmation textuel de haut niveau dédié aux applications d'automatisation ; il est utilisé principalement pour décrire les procédures complexes et difficilement modélisables avec les langages graphiques. Il peut aussi être utilisé en tant que sous programme avec d'autres langages de programmation.

Il utilise les mêmes énoncés que les langages de programmation de haut niveau (Pascal, C, C++…) comme: les assignations, les appels de fonction, les énoncés de contrôle (IF, THEN, ELSE, CASE) ou d'itération (FOR, WHILE, REPEAT), en plus des opérations arithmétiques et logiques.

Par exemple pour le calcul de la distance entre deux points dans un plan à deux dimensions :

```
FUNCTION Ecart: REAL
VAR_INPUT
X1, X2, Y1, Y2 : REAL;
END_VAR
BEGIN
RESULT := SQRT((X1-X2)^2 + (Y1-Y2)^2);
END_FUNCTION
```

Figure A.7: Exemple d'un programme en langage structuré

4.2.2 Liste d'Instructions :

Le langage IL est un langage textuel de bas niveau (proche du langage machine), qui utilise un jeu d'instructions simples. Il trouve sa puissance dans les applications de petites tailles, et dans la création de sous programme ou procédure, car il permet un contrôle totale et une optimisation parfaite du code ; par contre pour les grandes applications il est très difficile de programmer avec le IL ; les programmes dans ce langage peuvent être traduit ou déduit des autres langages.

Le IL a la même structure que l'assembleur ; il utilise un ou plusieurs registres de travail. Les valeurs intermédiaires nécessaires pour l'exécution d'une instruction donnée seront mémorisées dans ces registres le temps de leur utilisation et il possède un jeu d'instructions assez riche pour décrire toutes les opérations arithmétiques et logiques, les opérations de comptage et temporisation, la comparaison et le transfert…

On prend l'exemple suivant :

```
(* Calcul de K3=fct(y+K2*H/2)*)
LD K2
MUL H
DIV 2.0
ADD Y
ST Y2
CAL RK_1 (X:=Y2,R0:=R,C0:=C,Y0=>K3 )
```

Figure A.8 : Exemple de programme en Liste d'Instructions**.**

5. La gamme Schneider Electric en automates programmables :

5.1. ZELIO LOGIC :

C'est un module logique programmable destiné à la réalisation de petits automatismes de moins de 40 E/S pour des applications telles que :

- Comptage de pièces.
- Commande de pompes et de compresseurs.
- Convoyeurs.
- Panneaux d'affichage.
- Volets roulants.

Sa programmation, très intuitive, s'effectue directement sur la face ou à l'aide du logiciel Zelio Soft. Existe en version sans afficheur ni touches.

Les principales caractéristiques du ZELIO sont :

- Gamme compacte : 3 modèles monoblocs de 10, 12, 20 E/S (avec ou sans afficheur et touches).
- Gamme modulaire : 2 bases de 10 ou 26 E/S extensibles jusqu'à 40 E/S par modules d'extension de 6, 10 ou 14 E/S et un module de communication Modbus.
- Alimentation : 100/240 VCA ou 24 VCC.
- Fixation sans accessoire avec nez modulaire de 45 mm.
- Blocs fonctions prêts à l'emploi.
- Logiciel de programmation sur PC.

Figure A.9: Automate Zelio

5.2. TWIDO:

Version compacte ou modulaire partage des options, des extensions d'E/S et un logiciel de programmation communs. Application : installations simples et unitaires, machines répétitives et compactes.

Les principales caractéristiques sont :

- Applications standards de 10 à 100 E/S (maximum 252 E/S), 2 types de bases :
  - Compactes de 6, 10 ou 24 E/S.
  - Modulaires de 20 ou 40 E/S.
- Extension d'E/S : 14 modules TOR et 4 analogiques.
- Raccordement : borniers à vis ou à ressort, précâblage : connecteur HE10, Twido Fast, déport des E/S, maître AS-interface.
- Options : mémoire, horodateur, communication (liaison série), réglage (afficheur).
- Compacité : 40 E/S pour 95x90x70 mm.
- Fonctionnalités : compteurs rapides, sorties impulsionnelles, 1 port RS 485 cmultiprotocoles : Modbus Maître/esclave, ASCII, traitement rapide sur événements, type de donnés (mots, double mots, arithmétique flottante), jusqu'à 14 boucles PID…

Figure A.10: Automate TWIDO

5.3. TSX MICRO :

C'est un Automate développé pour satisfaire au mieux les exigences d'adaptabilité et de maintenabilité des machines. Sa modularité et sa compacité répondent de manière économique à l'automatisation aussi bien de machines simples à quelques dizaines d'E/S que de machines plus complexes à 480 E/S. Pour simplifier le câblage des machines, le TSX MICRO supporte la connexion du bus AS-i. il se programme à l'aide du logiciel PL7 Micro sous Windows.

Les principales caractéristiques sont:

- Jusqu'à 248 E/S TOR « in rack ».
- 96 E/S par déport TSX Nano.
- 248 E/S sur bs AS.
- Fonctions de comptage rapide (50kHz), analogique et régulation.
- Sécurité machine.
- Communication Uni-Telway, ASCII, Modbus, Fipway, FIPIO agent, modem.

Figure A.11: Automate TSX

5.4. PREMIUM :

Cette plate-forme d'automatismes, grâce à son architecture distribuée permet de répartir les cartes d'E/S et les fonctions métiers au plus près de l'application. Idéale pour les machines et installations modulaires et réparties, la gamme TSX Premium se décline :

- En format standard (TSX).
- EN format co-processeur pour PC.

L'intégration logicielle et matérielle des métiers permet de diminuer les temps de développement et de mise en route de l'installation. Enfin TSX Premium s'intègre à tous les niveaux de l'architecture de communication. Il se programme à l'aide des logiciels Unity Pro ou PL7.

Ses principales caractéristiques sont :

- Jusqu'à 2048 E/S TOR « in rack ».

E/S à distance sur bus FIPIO et tiers.

- 8 coupleurs AS-i, 248 E/S sur bus AS-I.
- Fonction de comptage rapide (1 MHz), analogique et régulation intégrée par configuration.
- Sécurité machine.
- Commande de mouvement multiaxe.
- Communication Uni-Telway, ASCII, Modbus, Fipway,FIPIO, Ethernet.
- TCP-IP, Ethway.

5.5. Modicon M340 :

Robuste, puissant et compact, idéal pour les constructeurs de machines dans des applications telles que le packaging secondaire, la manutention, le textile, l'imprimerie, l'agroalimentaire, les machines à bois, la céramique, idéal aussi pour la gestion technique des bâtiments. L'intégration des variateurs de vitesse Altivar et Lexium, des affi cheurs Magelis et des modules de sécurité Preventa a été particulièrement poussée pour simplifier la mise en oeuvre et l'exploitation des solutions Telemecanique. Modicon M340 est également le compagnon de Modicon Premium et Modicon Quantum pour répondre aux exigences d'automatisation des procédés industriels et des infrastructures, au coeur des architectures Transparent Ready.

Ses principales caractéristiques :

- Jusqu'à 1024 entrées/sorties TOR.
- Jusqu'à 256 entrées/sorties analogiques.
- Jusqu'à 3 réseaux Ethernet Modbus/TCP et jusqu'à deux modules réseaux.
- Un port Ethernet Modbus/TCP 10BASE-T/100BASE-TX.
- Un bus machines & installations CANopen.
- Une liaison série Modbus.
- Une prise TER de type USB (pour connexion d'un terminal de programmation ou d'un terminal de dialogue IHM Magelis XBT GT/GK/GTW).

Figure A.12: Automate M340

Comme nous travaillerons sur un Modicon M340 lors de notre projet, nous détaillerons les modules ainsi que les caractéristiques de ce dernier dans l'annexe de l'application Annexe D.

5.6. QUANTUM :

La plate-forme d'automatismes TSX Quantum, grâce à son architecture modulaire s'adapte à l'ensemble des applications du process exigeant de la disponibilité, une grande puissance de calcul et un nombre important de d'E/S. Le TSX Quantum se programme à l'aide des logiciels Unity Pro ou Concept.

Ses principales caractéristiques :

- Jusqu'à 64000 E/S décentralisées sur plusieurs km.
- Option de redondance sur alimentation, CPU et communications.
- Fonctions e comptage, commande d'axe sur bus SERCOS.
- Communication ASCII, Modbus plus, Ethernet TCP-IP, bus tiers.[cata].

Figure A.13 : Automate QUANTUM

ANNEXE B

UNITY PRO

1. Exemples de processeurs avec nombre de racks gérés :

Type de processeurs	Nombre de racks gérés
Pour tous les automates M340 Version 01.00.	1 rack
Pour BMX P34 1000 Version 02.00	2 racks
Pour BMX P34 20X0	4 racks

Tableau B.1 : Nombre de rack gérés par les processeurs M340

Type de processeurs	Nombre de racks gérés
TSX 57 0244	1 rack
TSX 57 1x4	Jusqu'à 4 racks
TSX P57 204 TSX PCI 57 204 TSX P57 254 TSX P57 2634 TSX P57 304 TSX P57 354 TSX P57 3634 TSX P57 454 / TSX PCI 57 354 TSX P57 4634 TSX P57 554	Jusqu'à 16 racks

Tableau B.2 : Nombre de rack gérés par les processeurs PREMIUM

Type de processeurs	Nombre de racks gérés
140 CPU 311-10	Ne dépend pas du type de processeur
140 CPU 434-12A	
140 CPU 534-14A	
140 CPU 651-50\60\60S	
140 CPU 671-60\60S	

Tableau B.3 : Nombre de rack gérés par les processeurs M340

2. Jeux d'instructions :

2.1. Liste d'instruction (IL) :

Modificateur	Applicable sur les opérandes du type de données	Description
N	BOOL, BYTE, WORD, DWORD	Le modificateur N est utilisé pour inverser bit à bit la valeur d'un opérande.
C	BOOL	Le modificateur C est utilisé pour exécuter l'instruction associée, si la valeur de l'accu est 1 (TRUE).
CN	BOOL	Si le modificateur C est combiné avec le modificateur N, l'instruction associée est exécutée seulement si la

		valeur de l'accumulateur est un 0 booléen (FALSE).
(	Toutes	Le modificateur Parenthèse gauche (est utilisé pour repousser l'évaluation de l'opérande, jusqu'à ce que l'opérateur Parenthèse droite) apparaisse. Le nombre d'opérations Parenthèse droite doit être égal au nombre de modificateurs Parenthèse gauche. Il est possible d'imbriquer les parenthèses.

Tableau B.4: Modificateurs.

Opérateur	**Modificateur**	**Signification**	**Opérandes**	**Description**
LD	N (uniquement pour les opérandes du type de données BOOL, BYTE, WORD ou DWORD)	Charge la valeur de l'opérande dans l'accumulateur	Valeur littérale, variable, adresse directe avec type de données quelconque	Avec LD, la valeur d'un opérande est chargée dans l'accumulateur. Les données de l'accumulateur s'adaptent automatiquement au type de données de l'opérande. Cela s'applique également aux types de données dérivés.
ST	N (uniquement pour les	Enregistre la valeur de l'accumulateur dans	Variable, adresse directe avec type de	Avec ST, la valeur actuelle de l'accumulateur est enregistrée dans l'opérande. Le type de données de l'opérande doit correspondre au type

	opérandes du type de données BOOL, BYTE, WORD ou DWORD)	l'opérande.	données au choix	de données de l'accumulateur.

Opérateur	**Modificateur**	**Signification**	**Opérandes**	**Description**
S	-	Définit l'opérande sur 1 si le contenu de l'accumulateur est 1.	Variable, adresse directe du type de données BOOL	S permet de définir l'opérande sur 1 lorsque le contenu de l'accumulateur actuel est un 1 booléen.
R	-	Définit l'opérande sur 0 si le contenu de l'accumulateur est 1.	Variable, adresse directe du type de données BOOL	R permet de définir l'opérande sur 0 lorsque le contenu actuel de l'accumulateur est un 1 booléen.
AND	N, N(, (	ET logique	Valeur littérale, variable, adresse directe du type de données BOOL, BYTE, WORD ou DWORD	L'opérateur AND établit une liaison ET logique entre le contenu de l'accumulateur et l'opérande. Pour les types de données BYTE, WORD et DWORD, la liaison est effectuée par bit.

OR	N, N(, (	OU logique	Valeur littérale, variable, adresse directe du type de données BOOL, BYTE, WORD ou DWORD	L'opérateur OR établit une liaison OU logique entre le contenu de l'accumulateur et l'opérande. Pour les types de données BYTE, WORD et DWORD, la liaison est effectuée par bit.
XOR	N, N(, (	OU exclusif logique	Valeur littérale, variable, adresse directe du type de données BOOL, BYTE, WORD ou DWORD	L'opérateur XOR établit une liaison OU exclusif logique entre le contenu de l'accumulateur et l'opérande. Si plus de deux opérandes sont reliés, le résultat de l'opération est à l'état 1 pour un nombre impair d'états 1 et à l'état 0 pour un nombre pair d'états 1. Pour les types de données BYTE, WORD et DWORD, la liaison est effectuée par bit.
NOT	-	Négation logique (complément)	Contenu d'accumulateur du type de données BOOL, BYTE, WORD ou DWORD	NOT permet d'inverser le contenu de l'accumulateur par bit.

Tableau B.5 : Opérateur Logiques

Opérateur	Modificateur	Signification	Opérandes	Description
ADD	(	Addition	Valeur littérale, variable, adresse directe du type de données INT, DINT, UINT, UDINT, REAL ou TIME	ADD permet d'ajouter la valeur de l'opérande à la valeur du contenu de l'accumulateur.
SUB	(	Soustraction	Valeur littérale, variable, adresse directe du type de données INT, DINT, UINT, UDINT, REAL ou TIME	SUB permet de retirer la valeur de l'opérande du contenu de l'accumulateur.
MUL	(	Multiplication	Valeur littérale, variable, adresse directe du type de données INT, DINT, UINT, UDINT ou REAL	MUL permet de multiplier le contenu de l'accumulateur par la valeur de l'opérande. .
DIV	(	Division	Valeur littérale, variable, adresse directe du type de données INT, DINT, UINT, UDINT ou REAL	DIV permet de diviser le contenu de l'accumulateur par la valeur de l'opérande.
MOD	(	Division modulo	Valeur littérale, variable, adresse directe du type de	MOD permet de diviser la valeur du premier opérande par la valeur du deuxième et

			données INT, DINT, UINT ou UDINT	de sortir le reste de la division (modulo) comme résultat.

Tableau B.6: Opérateurs arithmétiques.

Opérateur	Modificateur	Signification	Opérandes	Description
GT	(	Comparaison : >	Valeur littérale, variable, adresse directe du type de données BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT ou TOD	GT permet de comparer le contenu de l'accumulateur au contenu de l'opérande. Si le contenu de l'accumulateur est supérieur au contenu de l'opérande, le résultat est un 1 booléen. Si le contenu de l'accumulateur est inférieur ou égal au contenu de l'opérande, le résultat est un 0 booléen.
GE	(	Comparaison : >=	Valeur littérale, variable, adresse directe du type de données BOOL, BYTE, WORD, DWORD, STRING, INT,	GE permet de comparer le contenu de l'accumulateur au contenu de l'opérande. Si le contenu de l'accumulateur est supérieur/égal au contenu de l'opérande, le résultat est un 1 booléen. Si le contenu de l'accumulateur est inférieur au

			DINT, UINT, UDINT, REAL, TIME, DATE, DT ou TOD	contenu de l'opérande, le résultat est un 0 booléen.
EQ	(	Comparaison : =	Valeur littérale, variable, adresse directe du type de données BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT ou TOD	EQ permet de comparer le contenu de l'accumulateur au contenu de l'opérande. Si le contenu de l'accumulateur est égal à celui de l'opérande, le résultat est un 1 booléen. Si le contenu de l'accumulateur n'est pas égal à celui de l'opérande, le résultat est un 0 booléen.
NE	(	Comparaison : <>	Valeur littérale, variable, adresse directe du type de données BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT ou TOD	NE permet de comparer le contenu de l'accumulateur au contenu de l'opérande. Si le contenu de l'accumulateur n'est pas égal à celui de l'opérande, le résultat est un 1 booléen. Si le contenu de l'accumulateur est égal à celui de l'opérande, le résultat est un 0 booléen.
LE	(	Comparaison : <=	Valeur littérale, variable, adresse directe du type	LE permet de comparer le contenu de l'accumulateur au contenu de l'opérande. Si le

			de données BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT ou TOD	contenu de l'accumulateur est inférieur ou égal à celui de l'opérande, le résultat est un 1 booléen. Si le contenu de l'accumulateur est supérieur à celui de l'opérande, le résultat est un 0 booléen.
LT	(	Comparaison : <	Valeur littérale, variable, adresse directe du type de données BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT ou TOD	LT permet de comparer le contenu de l'accumulateur au contenu de l'opérande. Si le contenu de l'accumulateur est inférieur à celui de l'opérande, le résultat est un 1 booléen. Si le contenu de l'accumulateur est supérieur ou égal à celui de l'opérande, le résultat est un 0 booléen.

Tableau B.7 : Opérateurs de comparaison

Opérateur	Modificateur	Signification	Opérandes	Description

CAL	C, CN (uniquement si le contenu de l'accumulateur est du type de données BOOL)	Appel d'un bloc fonction, d'un DFB ou d'un sous-programme	Nom d'instance d'un bloc fonction, d'un DFB ou d'un sous-programme	CAL permet d'appeler un bloc fonction, un DFB ou un sous-programme avec ou sans conditions.
NOM_DE_FONCTION	-	Exécution d'une fonction	Valeur littérale, variable, adresse directe (le type de données dépend de la fonction)	Le nom de fonction vous permet d'exécuter une fonction.
NOM_DE_PROCEDURE	-	Exécution d'une procédure	Valeur littérale, variable, adresse directe (le type de données dépend de la procédure)	Le nom de procédure vous permet d'exécuter une procédure.

Tableau B.8 : Opérateurs d'appel

Opérateur	**Modificateur**	**Signification**	**Opérandes**	**Description**

JMP	C, CN (uniquement si le contenu de l'accumulateur est du type de données BOOL)	Saut vers l'étiquette	ETIQUETTE	JMP permet d'effectuer un saut avec ou sans conditions vers une étiquette.
RET	C, CN (uniquement si le contenu de l'accumulateur est du type de données BOOL)	Retour dans l'unité organisationnelle supérieure suivante du programme	-	Des opérateurs RETURN peuvent être utilisés dans des blocs de fonction dérivés DFB et des SR (sous-programmes). Des opérateurs RETURN ne peuvent pas être utilisés dans le programme principal. Dans un DFB, un opérateur RETURN force le retour au programme qui a appelé le DFB. Le reste de la section de DFB contenant l'opérateur 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 opérateur RETURN force le retour au programme qui a appelé le SR. Le reste du SR contenant l'opérateur RETURN n'est pas exécuté. Le programme qui a appelé le SR seraexécuté après retour du SR.
)	-	Traitement d'opérations placées en attente	-	La parenthèse droite) permet de lancer l'édition de l'opérateur en attente. Le nombre d'opérations Parenthèse droite doit être égal au nombre de modificateurs Parenthèse gauche. Il est possible d'imbriquer les parenthèses.

Tableau B.9 : Opérateurs de structuration

2.2. LADDER DIAGRAM (LD):

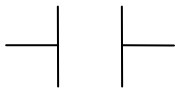
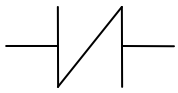
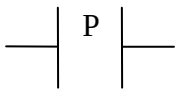
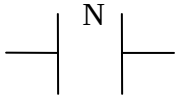
Désignation	Représentation	Description
A fermeture		Dans le cas de contacts à fermeture, l'état de la liaison de gauche est transféré vers la liaison de droite si l'état du paramètre booléen réel associé (indiqué par xxx) est ON. Sinon, l'état de la liaison de droite est OFF.
A ouverture		Dans le cas 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 qu'en même temps, l'état de la liaison de gauche est ON. Sinon, l'état 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, l'état de la liaison de droite est 0.

Tableau B.10 : Les contacts.

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	P	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	N	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	S	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.

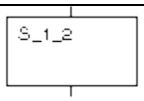
Bobine de Réinitialisation	R	Avec une bobine de réinitialisation, l'état de la liaison de gauche est copié sur la liaison de droite. Le paramètre booléen réel associé (indiqué par xxx) est défini sur OFF si l'état de la liaison de gauche est ON, sinon il reste inchangé. Le paramètre booléen réel associé 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 B.11: Bobines

2. 3. FBD :

Désignation	Représentation	Description
Le saut	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 conditionnel, l'objet saut est lié à une sortie FFB booléenne. Pour générer un saut inconditionnel, la valeur 1 est affectée à l'objet saut via la fonction AND.
Libellé	LABEL:	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 qu'entre les deux premiers points de trame sur la marge gauche de la section. Note : Les étiquettes de saut ne doivent "couper" aucun réseau, c'est-à-dire qu'une ligne imaginaire entre l'étiquette de saut et la marge droite de la section ne doit être coupée par aucun objet. Cela est également valable pour les liaisons.

2.4. SFC:

Type	Représentation	Description
Etape « normale »		Une étape devient active lorsque l'étape précédente devient inactive (un temps de retard éventuellement défini doit s'être écoulé) et la transition située en amont est vraie. Une étape devient généralement inactive lorsque le temps de retard éventuellement défini s'est écoulé et que la transition située en aval est vraie. Pour les convergences en ET, toutes les étapes précédentes doivent être vraies Chaque étape compte zéro ou plusieurs actions. Les étapes sans action sont considérées comme des étapes d'attente.
Etape initiale		L'état initial d'une séquence est caractérisé par l'étape initiale. A l'issue de l'initialisation du projet ou de la séquence, l'étape initiale est active. Généralement, aucune action n'est affectée aux étapes initiales. Pour les jetons uniques (conformes à la norme CEI 61131-3), seule une étape initiale est admise par séquence. Pour les jetons multiples un nombre d'étapes initiales pouvant être défini (de 0 à 100) est possible.
Macroétape		servent à appeler une macro-section et ainsi à établir une structure hiérarchique des commandes d'enchaînement.
Etape d'entrée		Chaque macro-section commence par une étape d'entrée.
Etape de sortie		Chaque macro-section se termine par une étape de sortie.

En plus de ces fonctions de contrôles, on peut importer toutes les fonctions et les blocs de fonctions de la bibliothèque.

3. Programmes des exemples d'application d'UNITY PRO :

3.1. Programmation en SFC :

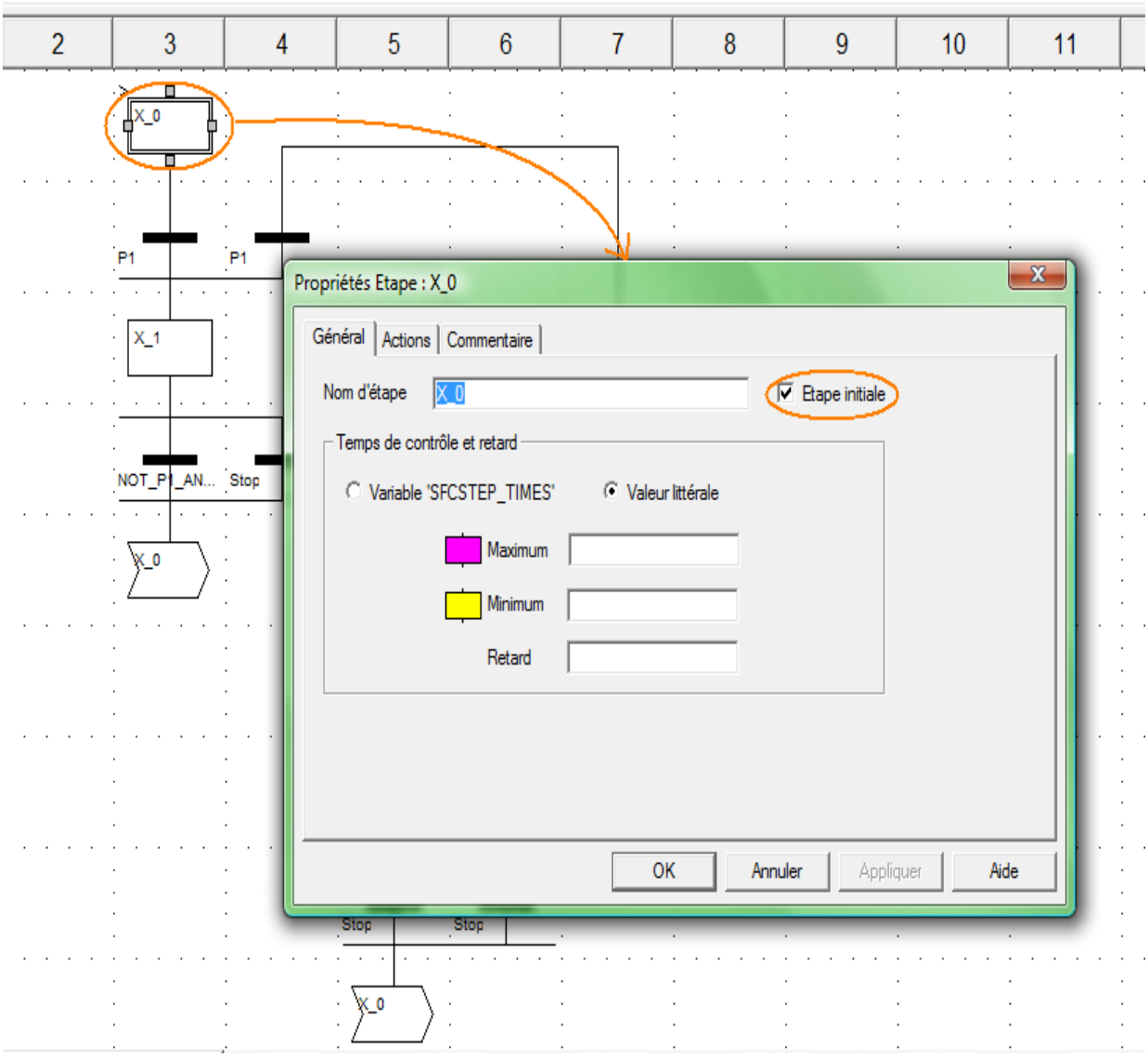


Figure B.1: Propriété de l'étape initiale X0.

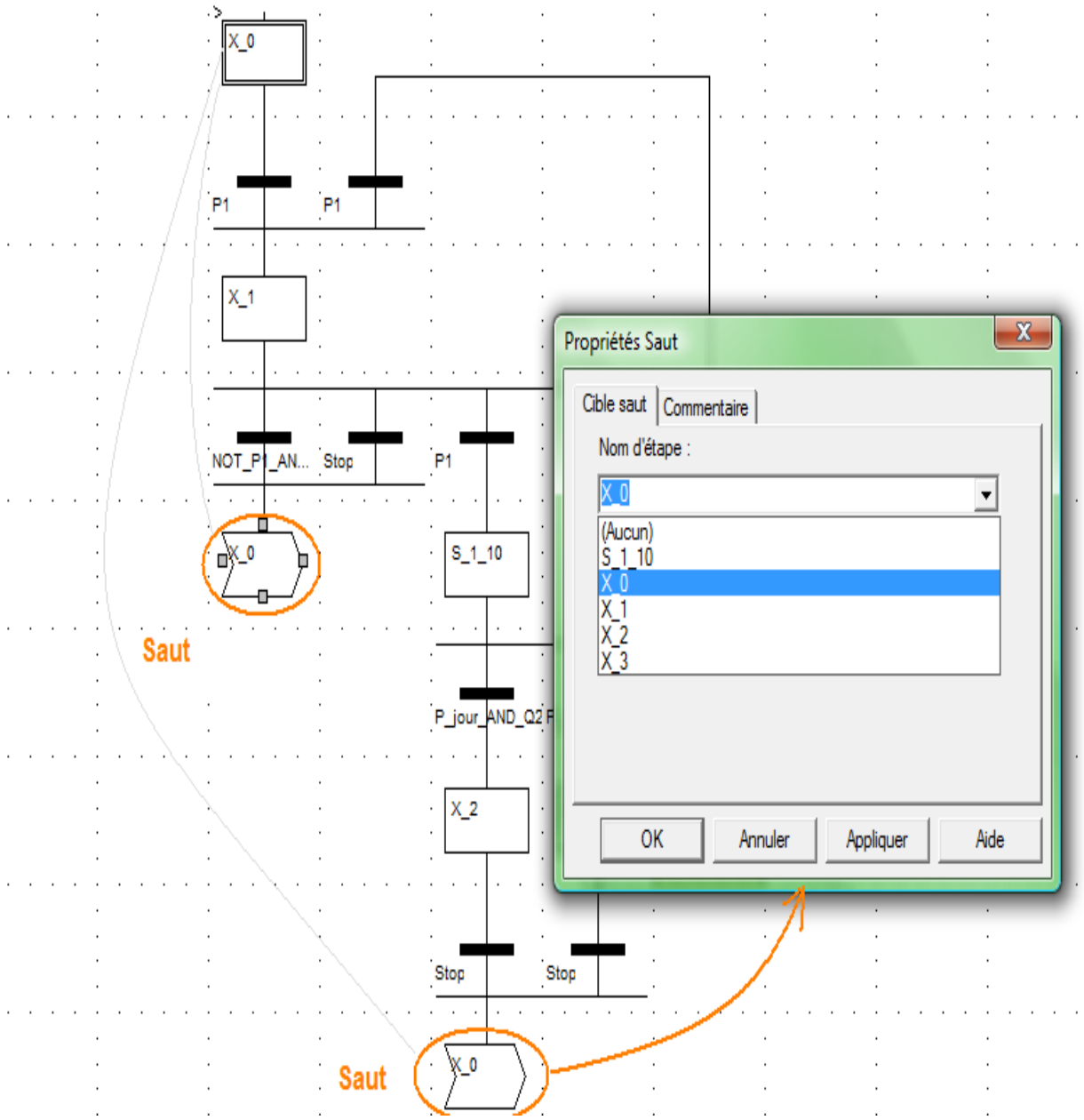


Figure B.2 : Propriété Saut

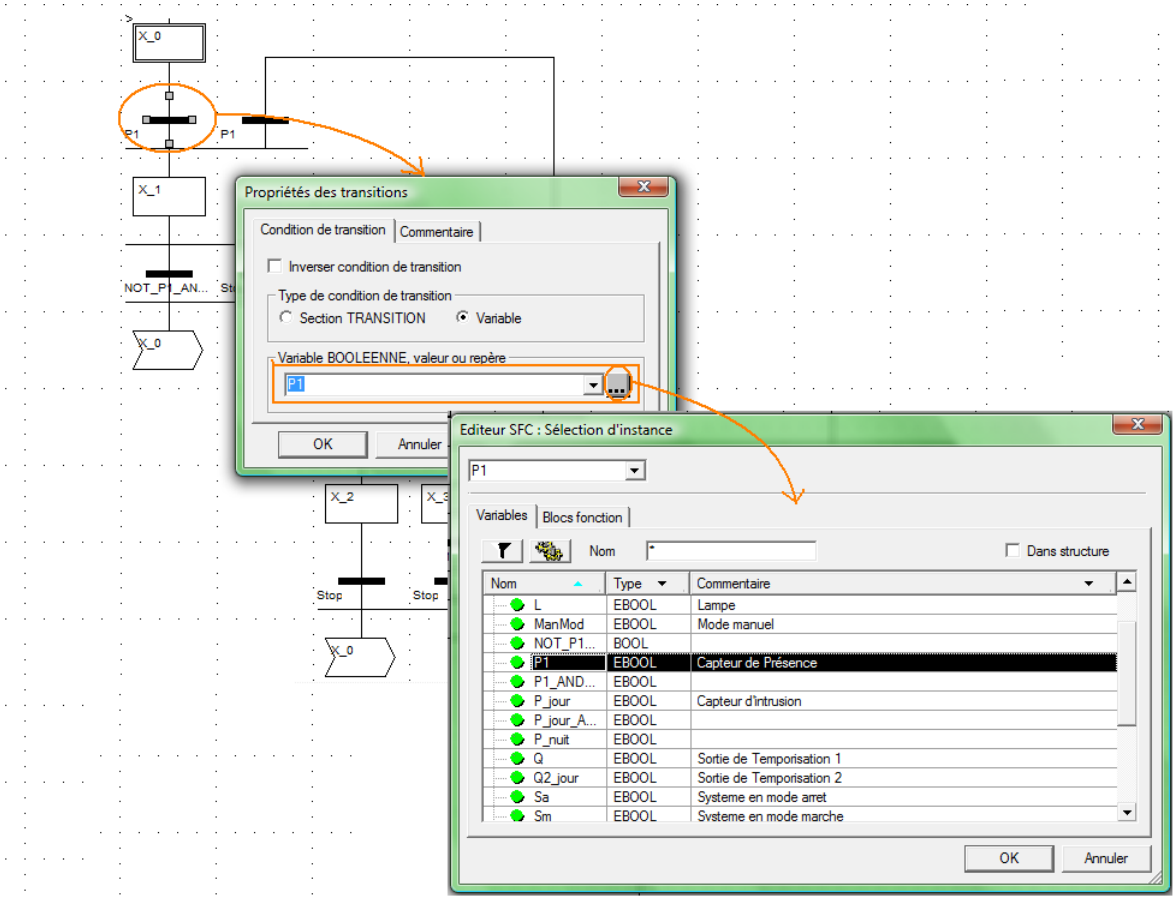
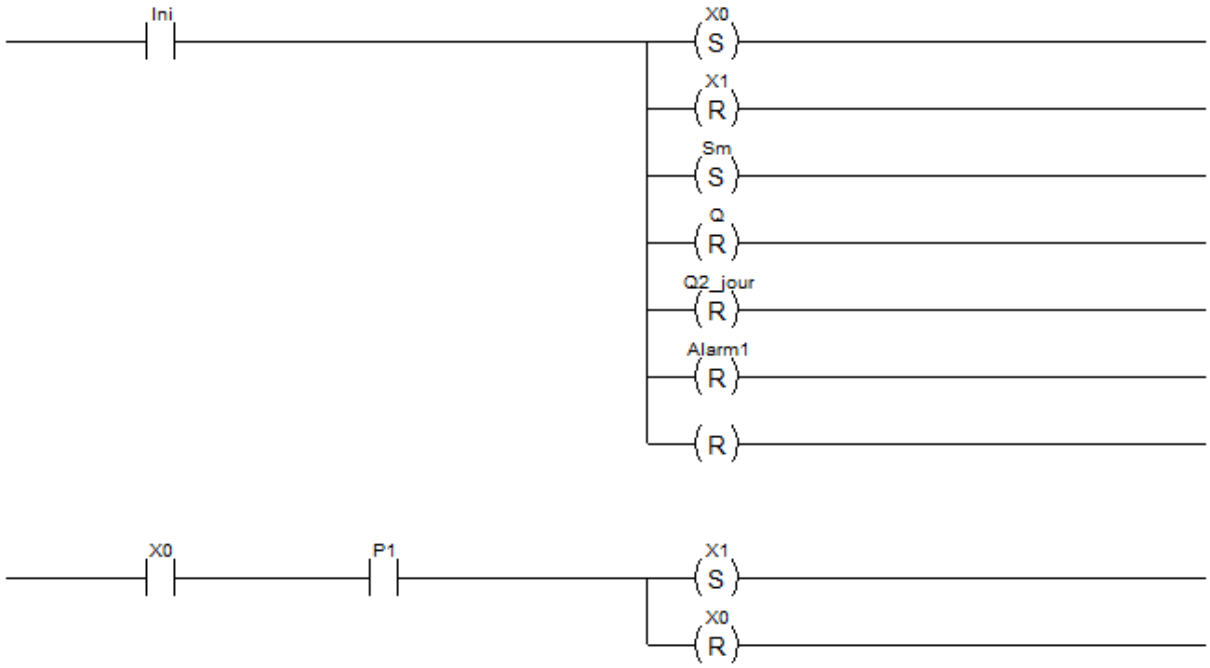


Figure B.3 : Propriété de l'étape et sélection d'instance

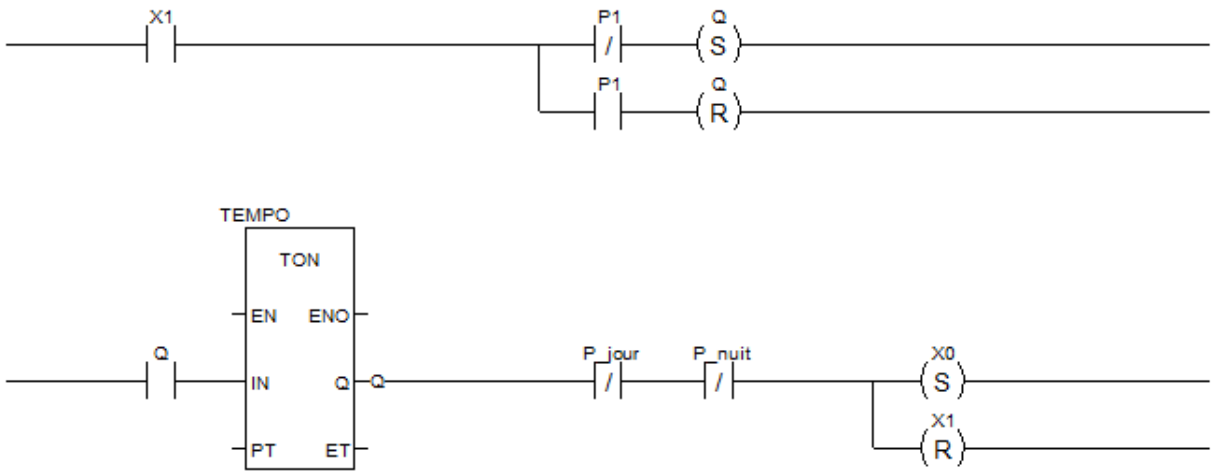
3.2. Le programmation en LADDER :

Les états

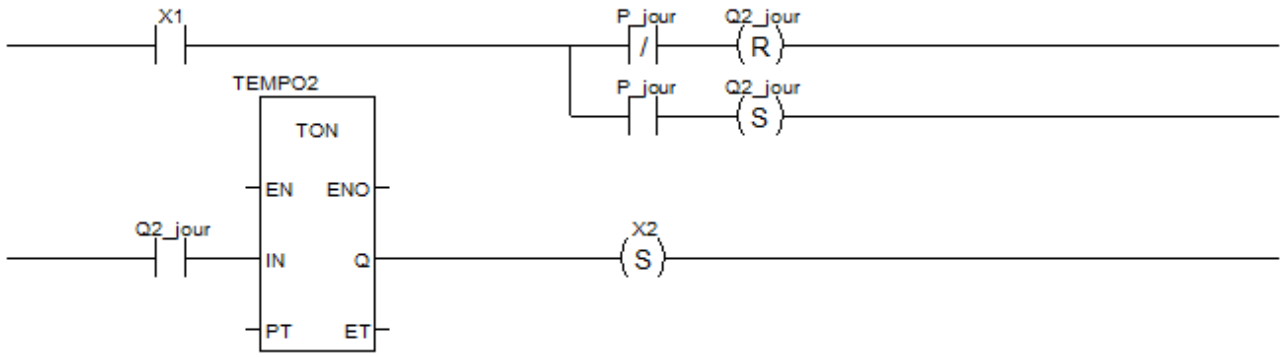
X0



X1

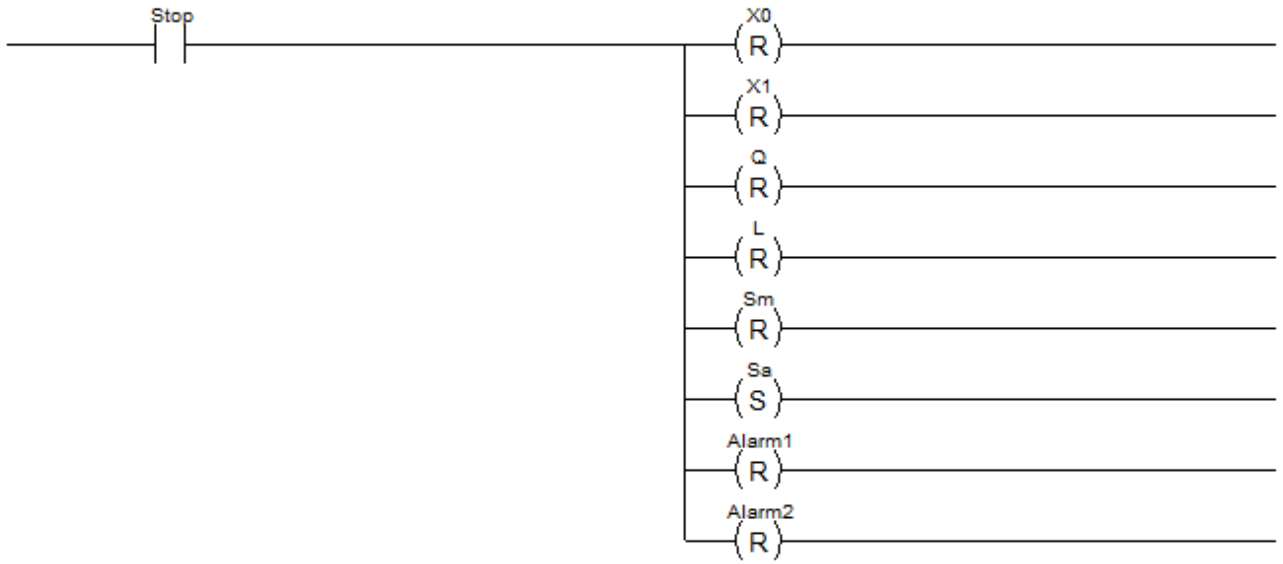


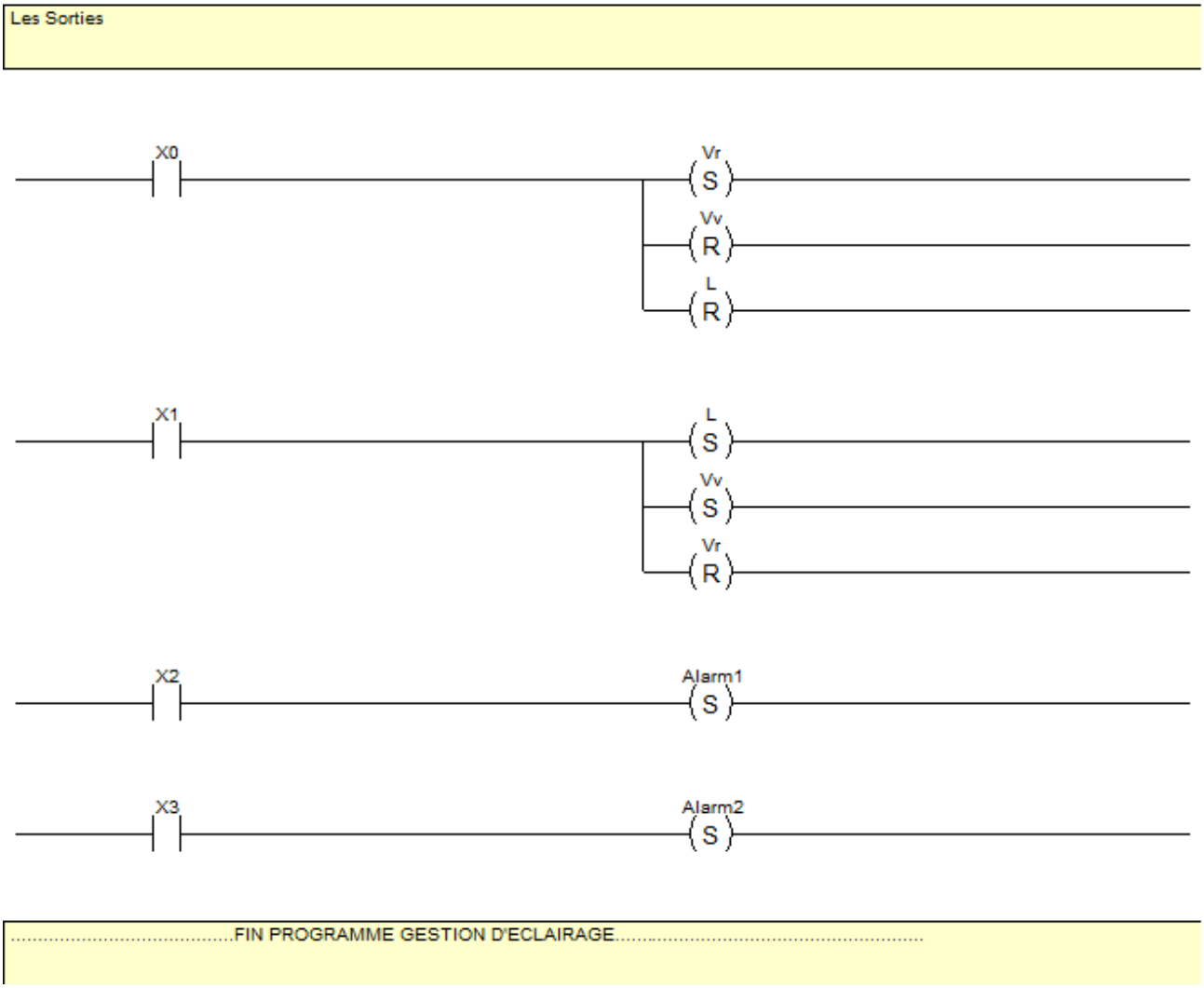
X2 , Controle d'accès au cours de la jounrée



X3 , Controle d'accès au cours de la nuit

Etap Stop





4. Exemple de Commande d'éclairage de la salle :

4.1. Programmation des FBD en langage Structuré ST

Les Programmes des deux Blocs :

Ces figures suivantes illustrent les programmes utilisés dans notre exemple.

```
if (Erreur=0 and Mode=True ) then

        if Presence=true
                then    Tempo:=false ;

                        if Lumiere=false

                                then Inter:=1;
                        end_if;

                        if (lumiere=true and Rideau=false )
                                then Inter:=0;
                        end_if;

                        if (Lumiere=true and Rideau=true)

                                then  Inter:=1;
                        end_if;

                end_if;


        if Presence=false
                then
                        Tempo:=true;
                        FBI_2 (IN := Tempo,
                               PT := T0,
                               Q => Q,
                               ET => Temps lampe On  );
                        if Q=True
                                then Inter:=0;
                                     Tempo:=false;
                        End_if;

                end_if;


end_if;
```

Figure B.4 : Programme de la section ST du Bloc Eclairage.

```
if (Defaut_Presence=false and Defaut_lumiere=false and Defaut_Rideau=false and Defaut_Disjoncteur=false
then
        Erreur:=0;
Else    Erreur:=1;

if      Erreur=1 then


                if Defaut_Presence=True    then Erreur_Presence:=true;     end_if ;

                if Defaut_lumiere=True then Erreur_lumiere:=true;       end_if;

                if Defaut_Rideau=True then Erreur_Rideau:=true;        end_if;

                if Defaut_Disjoncteur=True then Erreur_Disjoncteur:=true;   end_if;

                if Defaut_Alarme=True      then Erreur_Alarme:=true;        end_if;

end_if;
end_if;
```

Figure B.5: Programme de la section ST du Bloc Erreur.

ANNEXE C

VIJEO CITECT

Introduction :

Les Génies Vijeo Citect agissent comme des macros au sein d'un projet. Un Génie sert à associer plusieurs objets graphiques. Une pompe peut être composée d'une image de pompe, accompagnée d'un signal auto/manuel et d'un signal d'alarme. Toutes ces configurations sont regroupées dans un Génie.

La configuration est effectuée en associant un texte fixe à des paramètres. Les paramètres peuvent représenter un champ entier seul ou être associés à d'autres paramètres ou à du texte fixe pour représenter le contenu d'un champ.

Les Super Génies Vijeo Citect sont le plus souvent utilisés pour des écrans de contrôle contextuels. Un Super Génie est une combinaison d'objets graphiques regroupés sur une page ou un écran contextuel. Un écran contextuel de contrôle de boucle peut comprendre des curseurs, boutons, valeurs de tendance et autres configurations, définis comme un Super Génie et pouvant être réutilisés dans tout le projet.

Grace à ces deux outils vous n'aurez à créer et configurer un objet une seule fois, et vous pouvez le sauvegarder dans une bibliothèque et l'utiliser autant de fois que vous le désirez.

Nous allons lors de ce chapitre apprendre à créer un Génie et un Super Génie.

1. Création d'un Génie :

Nous allons créer un Génie « Disjoncteur » qui aura deux états, fermé et ouvert, que l'on va associer à des variables « Etat_disjoncteur_i » et nous verrons le schéma du disjoncteur ouvert ou fermé selon le cas de ces variables.

Pour ce faire, créez un nouveau projet et ajoutez un cluster et un serveur avec comme adresse l'adresse de l'automate, dans notre cas nous n'allons que simuler donc utilisez 127.0.0.1 comme adresse, et créez trois variables digitales internes, « Etat_disjonteur1 », « Etat_disjonteur2 » et « Etat_disjonteur3 » qui seront égales à « 1 » lorsque le disjoncteur i sera fermé.

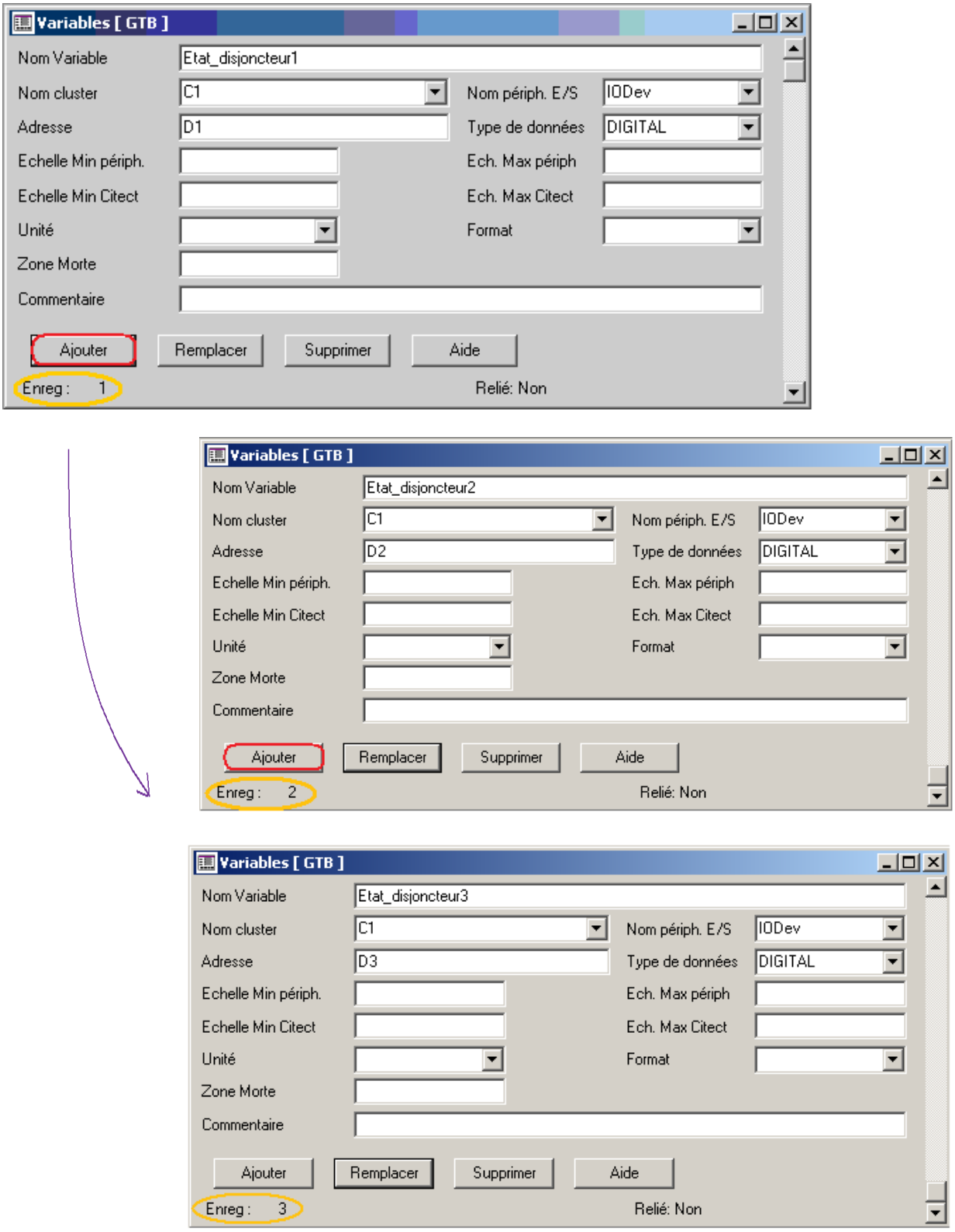
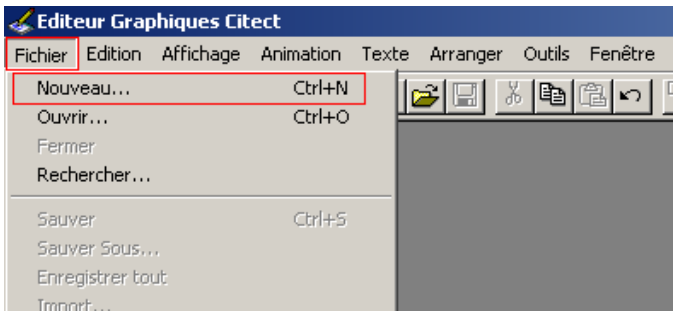


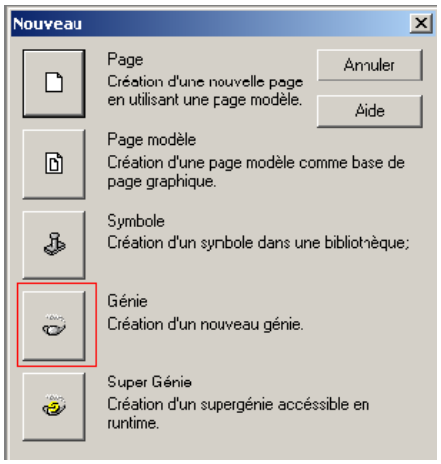
Figure D.1 : Déclaration des variables

Cela étant fait, allez dans l'éditeur graphique et cliquez sur fichier puis sur nouveau.à cliquez sur Génie.

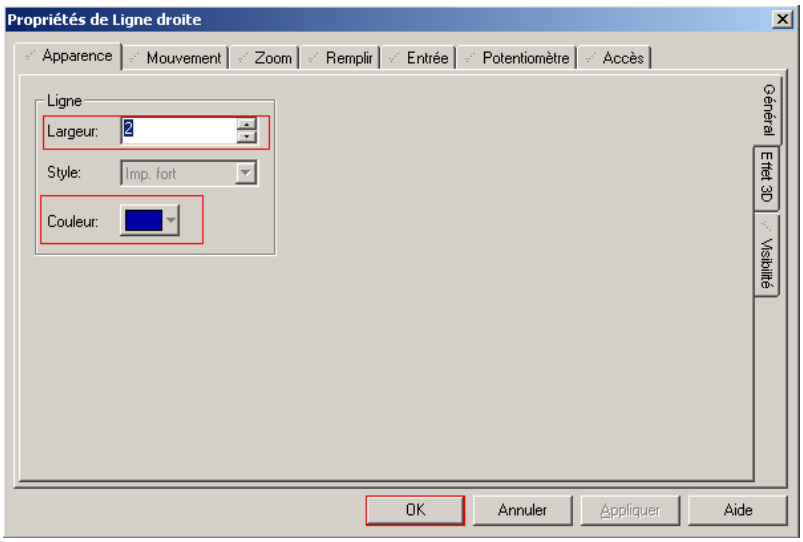


Nous arrivons à la page de création des Génies.

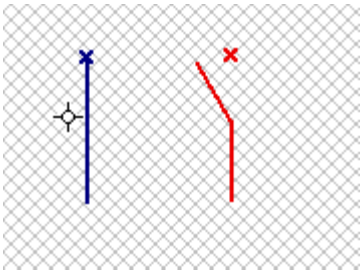
Dessinez votre objet, dans notre cas dessinons deux disjoncteurs, l'un d'eux ouvert et l'autre fermé. Utilisons pour cela l'outil trait qui se trouve sur la barre d'outils.



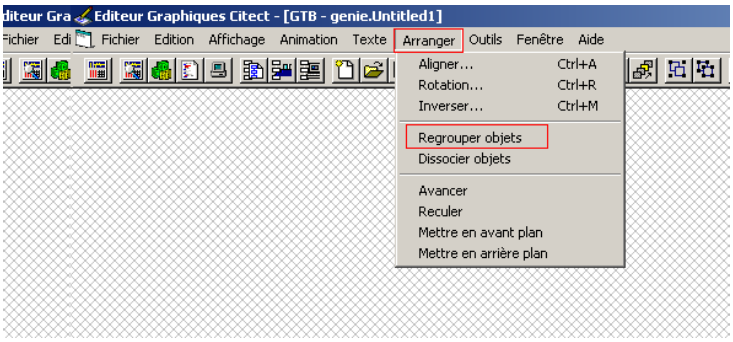
Dessinez donc vos lignes et à chaque fois que vous en finissez une, une boite de dialogue s'ouvre pour vous permettre de configurer le trait dessiné, mettez la largeur des traits à 2 à chaque fois, et mettez la couleur des traits à bleu pour le disjoncteur fermé et à rouge pour le disjoncteur ouvert et validez à chaque fois en appuyant sur le bouton OK.



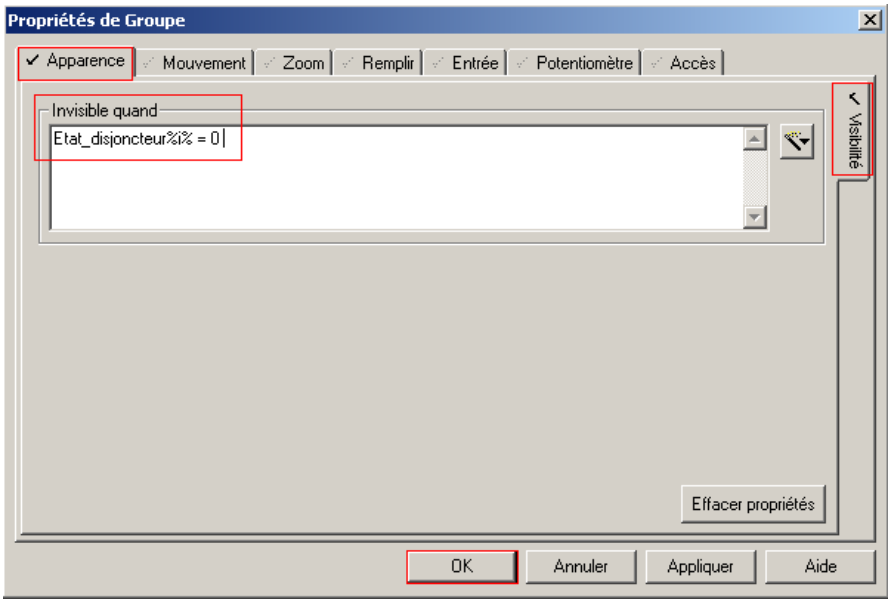
Dessinez deux disjoncteurs, l'un ouvert et l'autre fermé.



Sélectionnez tous les objets des deux disjoncteurs, un à un, et regroupez les en un seul objet, pour cela cliquez sur arranger puis sur Regrouper Objets.



Après cela double cliquez sur le disjoncteur fermé, une boite de dialogue apparaît à chaque fois, avec dans l'onglet Visibilité, mettez invisible quand « Etat_disjoncteur%i% = 0 », pour qu'il soit invisible lorsque le disjoncteur i sera ouvert puis validez en cliquant sur ok.

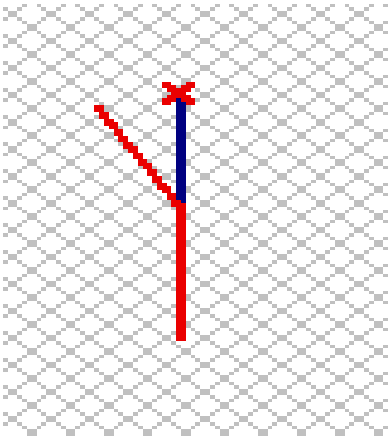


Faites de même pour le disjoncteur à l'état ouvert, en mettant « Etat_disjoncteur%i%= 1 » pour qu'il ne soit visible que lorsque le disjoncteur sera ouvert.

Nous avons mis « i » entre deux symboles « % » car tout ce qui sera entre ces deux symboles sera remplacé par une chaine de caractères choisis lorsque vous poserez le Génie dans la page graphique. Ainsi lorsque nous poserons le Génie dans la page graphique, un pop up apparaitra pour qu'on remplace cet élément par une chaine de caractère de notre choix. Nous remplacerons, dans notre cas, ces « i » par le chiffre du disjoncteur adéquat.

Une fois que vous avez fini, superposez les deux disjoncteurs et alignez les avec l'outil aligner, et regroupez les en un seul élément.

Vous devriez obtenir quelque chose comme ça.



Sauvegardez votre Génie avec comme nom par exemple, Disjoncteur, en cliquant ok pour le sauvegarder on vous propose de créer une librairie pour le projet. Créez la librairie « Genies » pour notre exemple, puis appuyez sur ok pour valider.

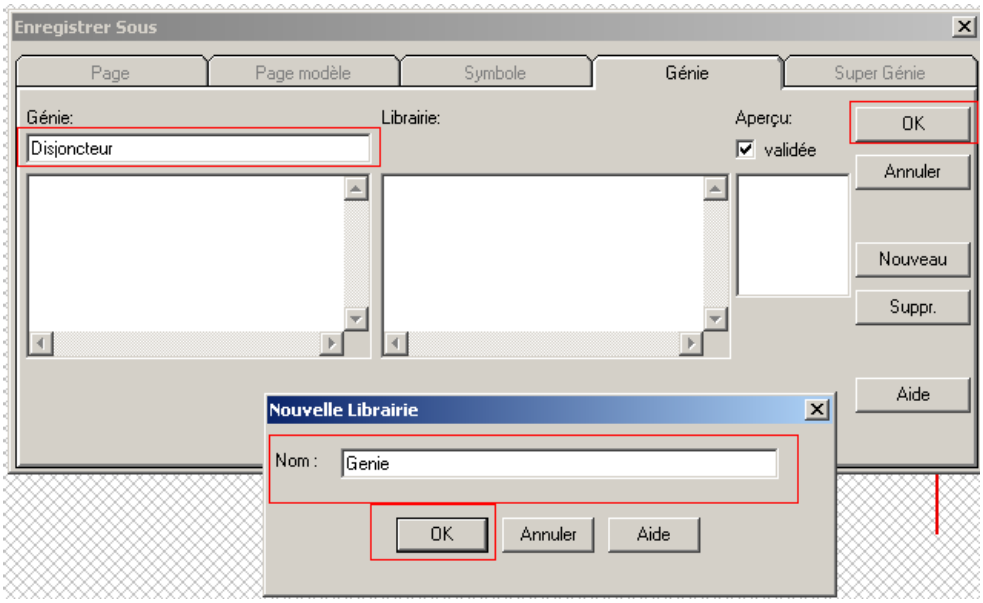
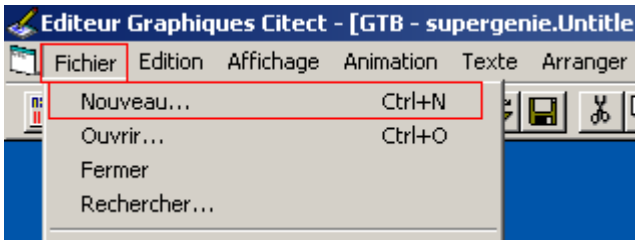


Figure : Enregistrement du Génie

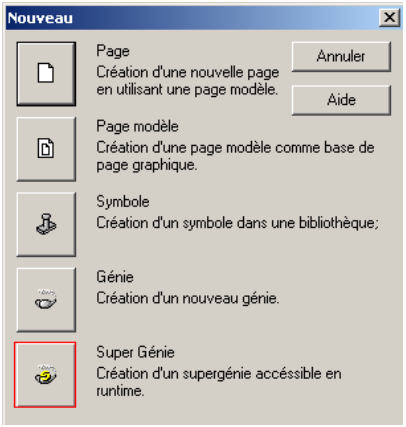
Notre Génie est donc créé, nous allons maintenant créer un Super Génie que nous allons intégrer à notre Génie pour qu'un Pop Up apparaisse quand on clique sur le Génie et qui nous propose d'ouvrir ou de fermer notre disjoncteur selon le cas actuel du disjoncteur.

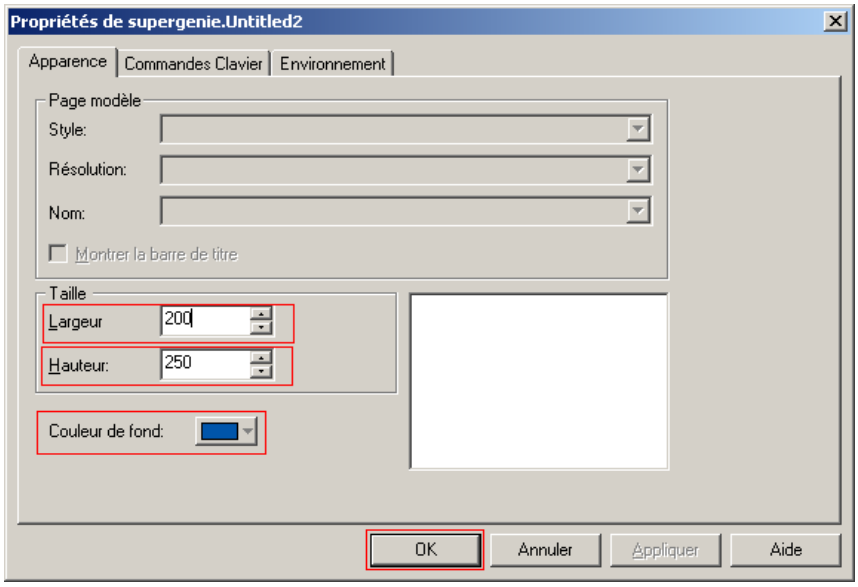
2. Création d'un Super Génie :

Cliquez sur Fichier puis sur Nouveau :



Là choisissez Super Génie :

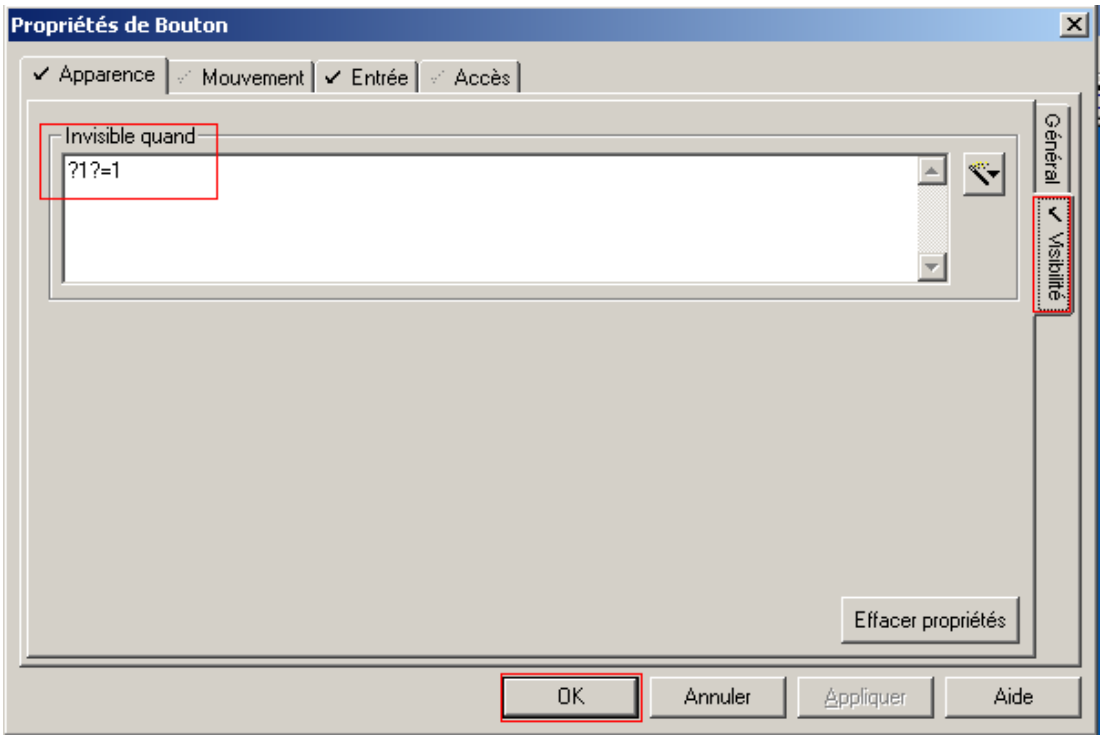


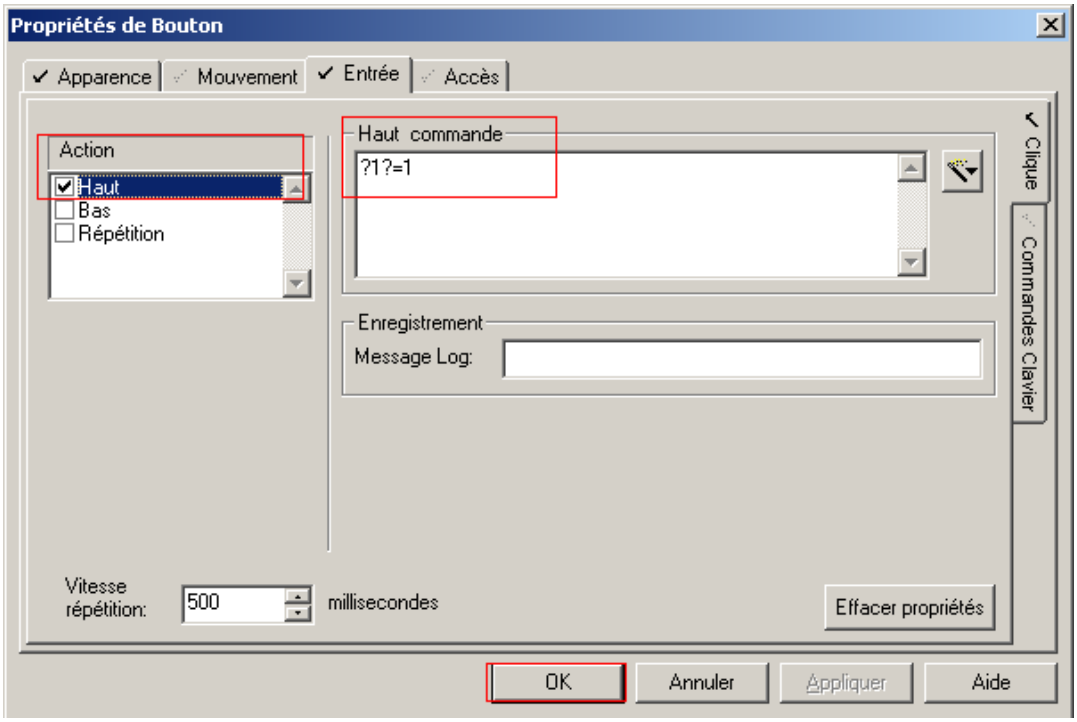


Cliquez avec le bouton droit et choisissez Propriétés Page, une fenêtre apparaît ou vous pourrez choisir la largeur et la hauteur de votre Pop Up et la couleur de fond, quand vous aurez fini validez en cliquant sur OK.

Créez deux boutons, un pour fermer et l'autre pour ouvrir les disjoncteurs.

Dans les options de visibilité du bouton fermer mettez invisible quand « ?1 ? =1 », « ?1 ? » est une variable à laquelle nous associerons une variable lors de l'appel du Super Génie, dans notre cas nous lui donnerons comme variable « Etat_disjoncteur%i% », comme cela le bouton Fermer ne sera pas visible lorsque le disjoncteur sera déjà fermé.





Allez maintenant dans Entrée, et cochez Action Haut et mettez dans Haut commande, « ?1 ?=1 » pour que le disjoncteur se ferme lorsqu'on cliquera sur le bouton fermer.

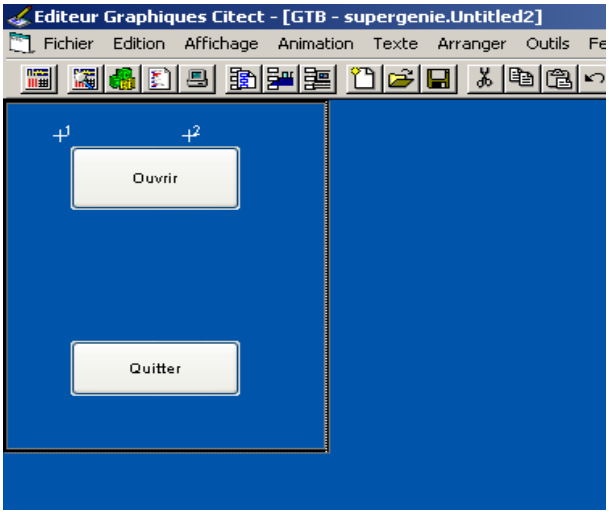
Faites le contraire pour le bouton ouvrir et superposez les pour qu'on ne puisse voir qu'un seul des deux au même endroit.

Créez maintenant un bouton qui nous permettra de fermer le Pop Up une fois ouvert.

Pour cela créez un bouton que vous appellerez « Quitter » avec comme commande haute la commande « WinFree() » qui permet de fermer la fenêtre en cours.

Créez aussi un cadre autour de la fenêtre, de largeur 2 avec un effet 3D rabaissé, pour que le pop up ait un cadre visible d'une autre couleur.

Vous devriez avoir ceci :



Sauvegardez votre Super Génie, sous le nom «Commande_disjoncteur » par exemple. Il est prêt à l'emploi.

3. Appel d'un Super Génie :

Nous voulons faire apparaître le Pop Up lorsque l'utilisateur cliquera sur le disjoncteur du Génie.

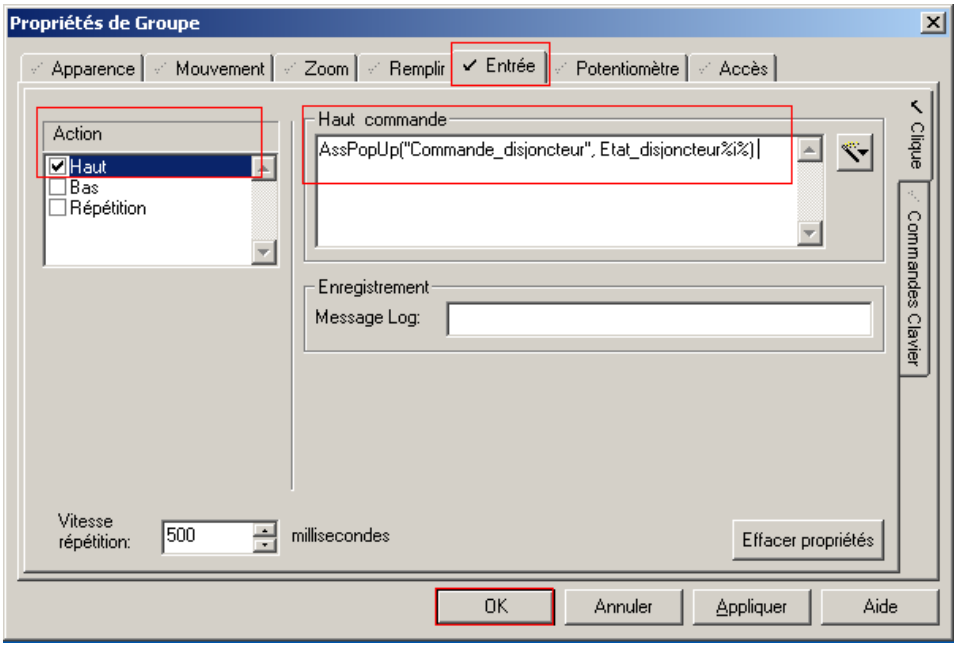


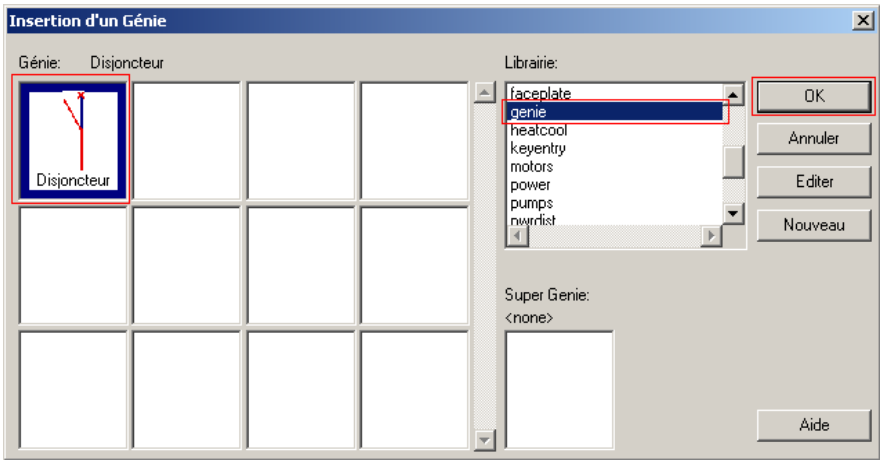
Figure : Appel d'un Super Génie

Pour cela retournez au Génie et double cliquez sur le disjoncteur, dans la boite de dialogue qui apparaît, cliquez sur Entrée et mettez comme fonction, « AssPopUp(''Commande_disjoncteur'', Etat_disjoncteur%i%) », la fonction AssPopUp appelle le Super Génie et attribue aux variables entre « ? », les variables données sous la forme (AssPopUp(''Super Génie'', variable ?1?, variable ?2?, …), ainsi nous avons appelé notre Super Génie et nous avons attribué à notre variable ?1?, l'Etat du disjoncteur « i ».

4. Appel d'un Génie dans une page de projet :

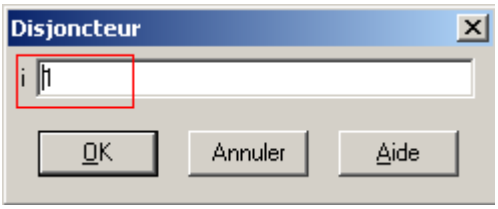
Afin d'intégrer un Génie au projet, cliquez sur le bouton Génie dans la barre d'outil.

Là, une boite de dialogue apparaît, choisissez votre Génie que vous trouverez dans la Librairie « genie » que vous aviez créé, et choisissez le Génie Disjoncteur.

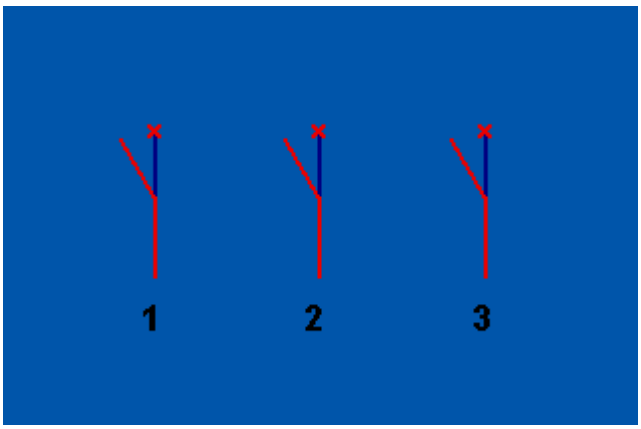


Là une autre boite de dialogue apparaît qui vous demandera de par quoi remplacer le paramètre « i », placez trois Génies comme ça avec comme i, 1,2 et 3

.



On aura ceci :



Exécutez le projet, vous verrez que lorsqu'on clique sur le disjoncteur le Pop Up apparaît et nous pouvons modifier l'état du disjoncteur grâce aux boutons, et vous pourrez fermer le Pop Up grâce au bouton Quitter.

ANNEXE D

L’APPLICATION

1. Plate-forme d'automatisme 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 d'automatisation des procédés industriels et des infrastructures.

Figure D.1 : Automate Modicon M340

1.1. Processeurs Modicon M340 :

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

- Modules d'entrées/sorties "Tout ou Rien".
- 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 d'E/S et nombre et type de ports de communication. De plus, selon le modèle, ils proposent au maximum et d'une manière non cumulative :

- De 512 à 1024 entrées/sorties "Tout ou Rien".
- 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 d'un terminal de programmation).

Chaque processeur est fourni avec une carte mémoire permettant :

- La sauvegarde de l'application (programme, symboles et constantes).

- L’activation d’un serveur Web de base du port Ethernet intégré de classe.

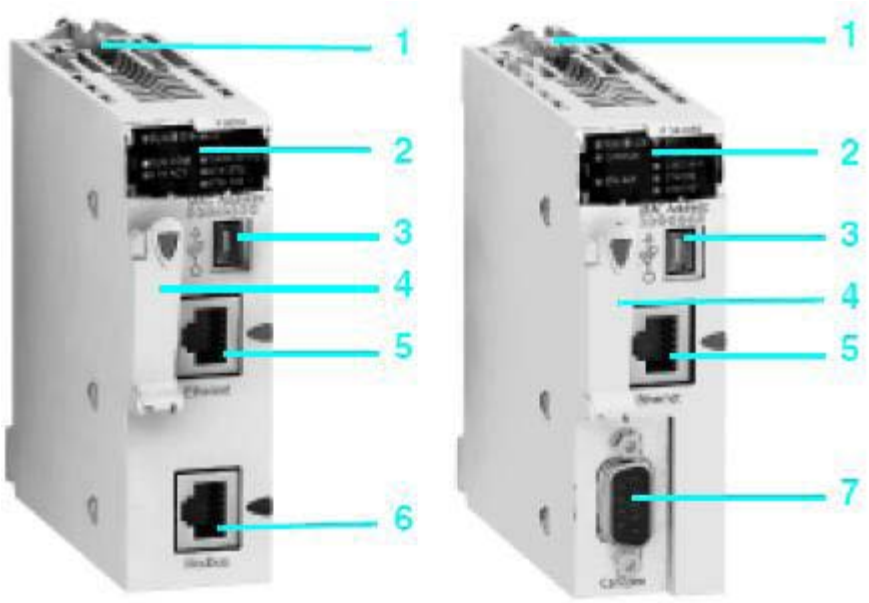


Figure D.2 : Processeurs : BMX P34 2020 et BMX P34 2030

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 d’entrées/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 d’un terminal de dialogue opérateur Magelis XBT GT/GK/GTW).

4 Un emplacement équipé de sa carte mémoire Flash pour la sauvegarde de l’application. Un voyant, situé au dessus de cet emplacement indique la reconnaissance ou l’accès 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 l'attribution de l'adresse IP. Cette attribution est définie selon 3 modes :

- Adresse fixée par la position des 2 commutateurs.
- Adresse fixée par les paramètres de l'application.
- Adresse fixée par le serveur BOOTP du réseau Ethernet TCP/IP.

1.2. 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. L'utilisation des données non localisées supprime les contraintes de gestion de la localisation mémoire du fait de l'attribution 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 l'automate 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 "backup" de cette mémoire par programme utilisateur.

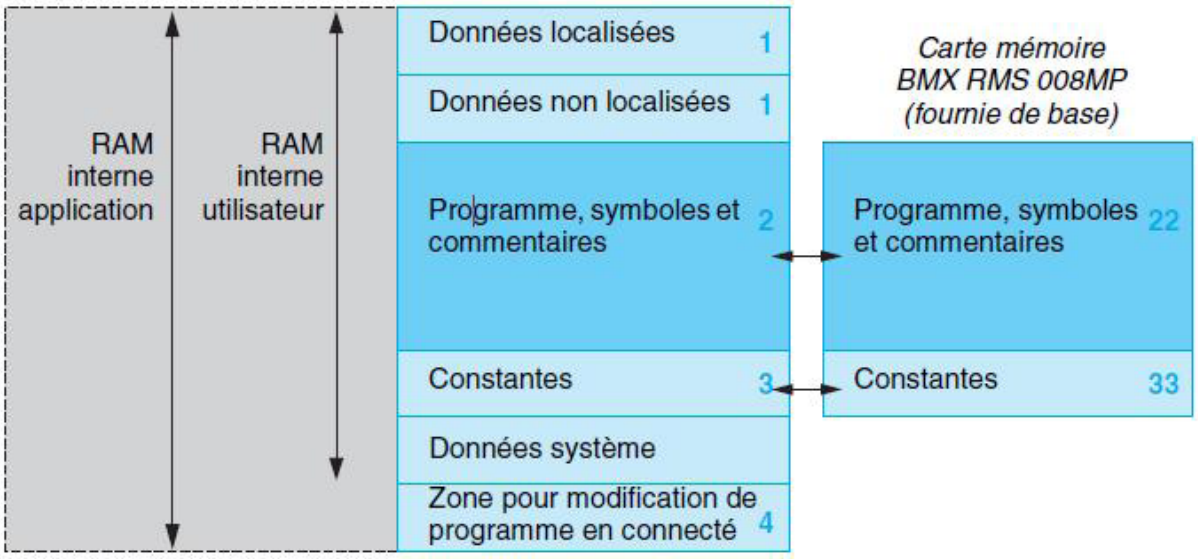


Figure D.3 : Structure de la mémoire

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.

1.3. Module d'alimentation :

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 100…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).

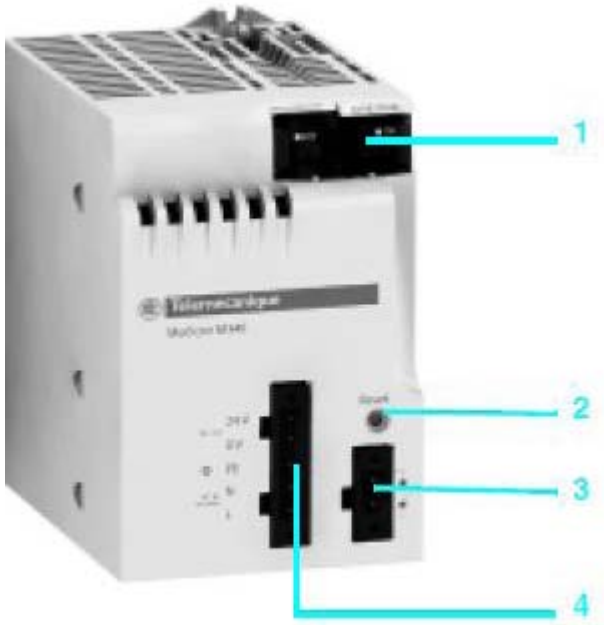


Figure D.4 : Module d'alimentation

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 d'alimentation CC ou AC.
- De la terre de protection.
- De la tension c 24 V dédiées à l'alimentation des capteurs d'entrées (uniquement avec modules alimentation courant alternatif BMX CPS 2000/3500).

1.4. Configuration monorack :

Présentation :

Le rack BMX XBP pp00 constitue l'élément de base de la plate-forme d'automatisme 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 "Tout ou Rien", entrées/sorties analogiques et métiers). Ces racks peuvent être fixés sur panneau, platine ou profilé DIN :

- Dans des armoires.
- Dans des bâtis de machines, …

- 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.
  - L'embrochage 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 d'extension. 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 l'encrage des ergots des modules.

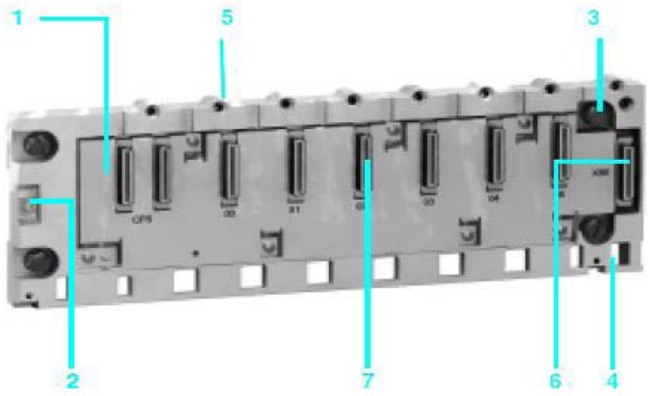


Figure D.5 : Profilé à 6 emplacements BMX XBP 0600

1.5. 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:

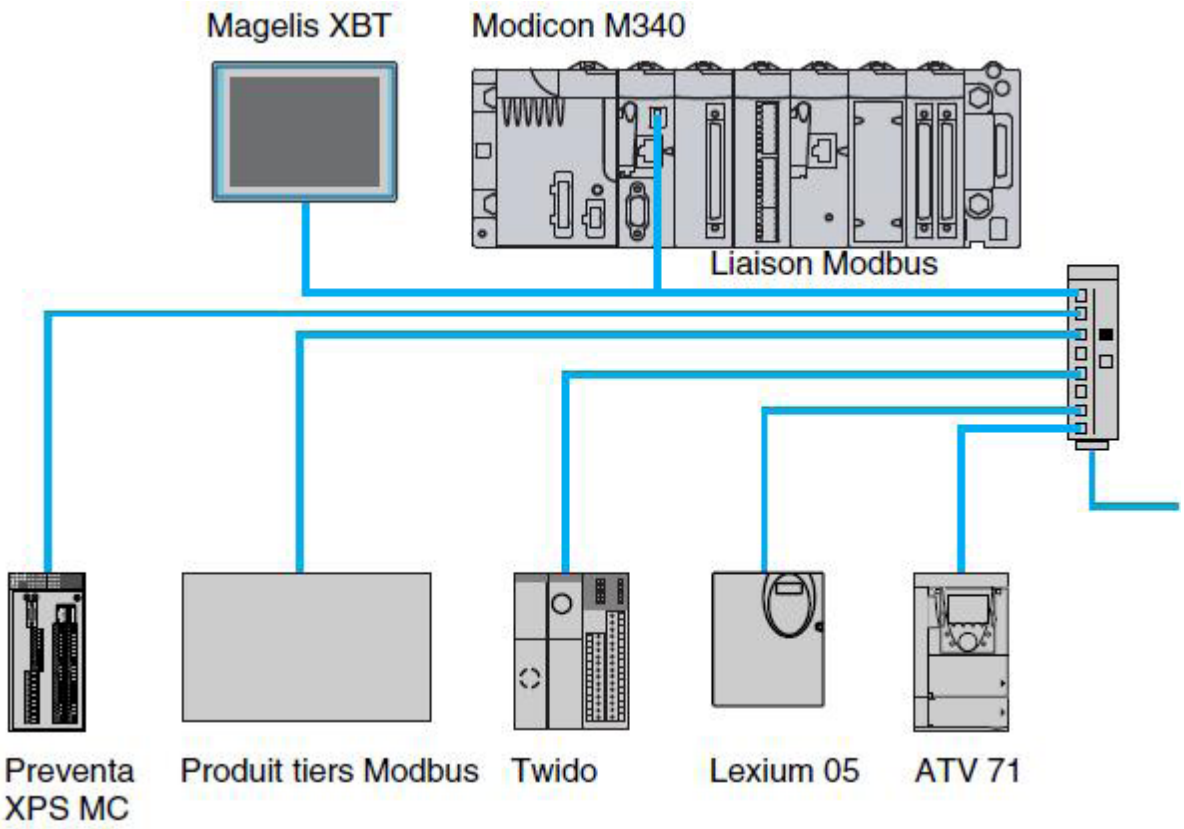


Figure D.6 : présentation du réseau Modbus

- Question/réponse, les demandes du maître sont adressées à un esclave donné. La réponse est attendue en retour de la part de l'esclave 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.

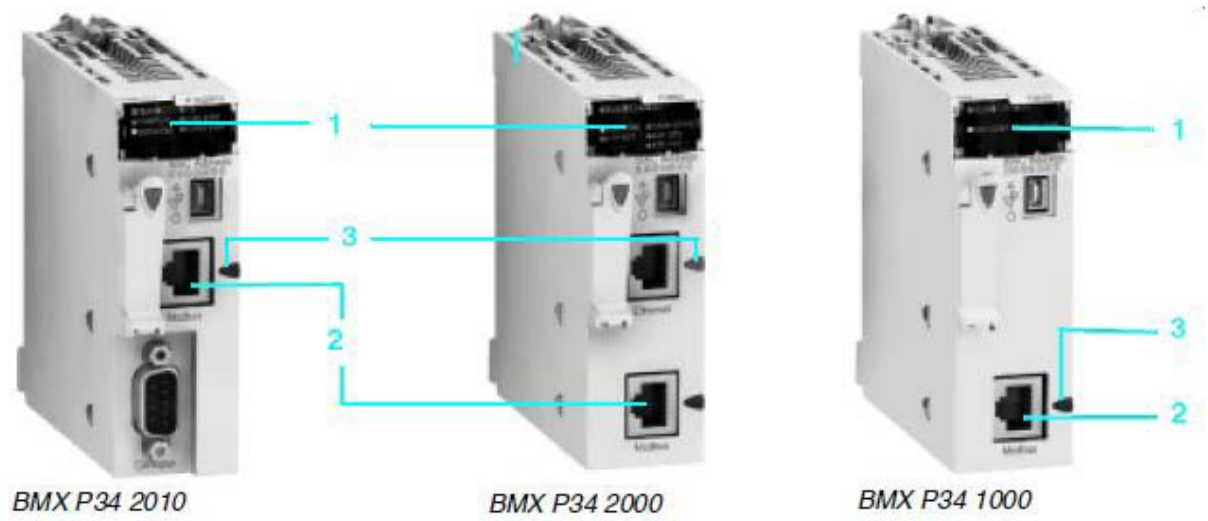


Figure D.7 : description de la communication M340

Système de câblage :

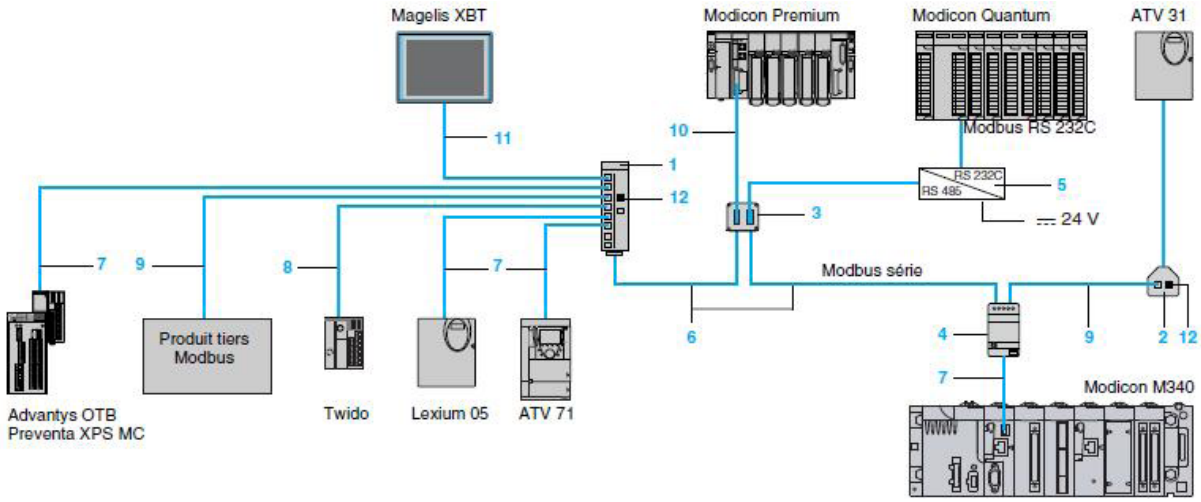


Figure D.8 : système de câblage

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 Q. indiv*	**VW3 A8 306 RC**

Tableau D.1 : Eléments de dérivation et d'adaptation pour liaison série RS

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

1.6. 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.

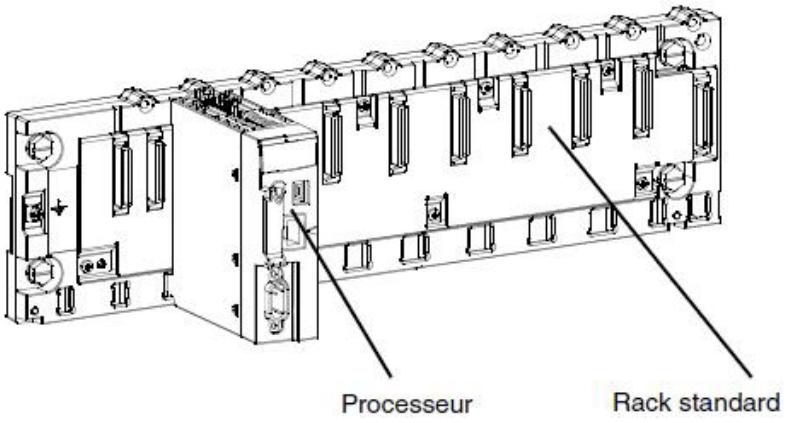


Figure D.9 : un processeur BMX P34 2010 monté dans un rack BMX XBP 0800

Le tableau ci-dessous présente la procédure d'installation d'un processeur sur un rack.

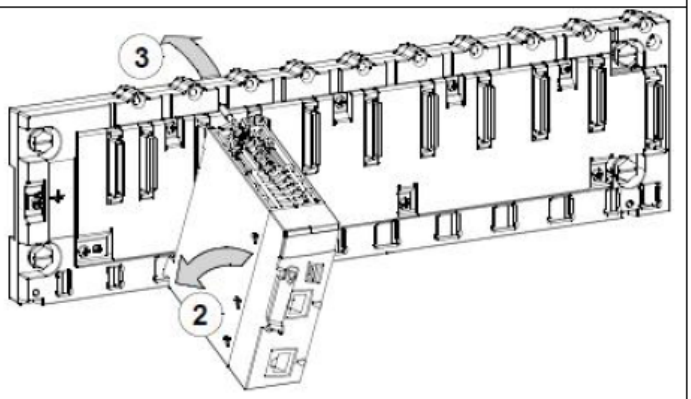
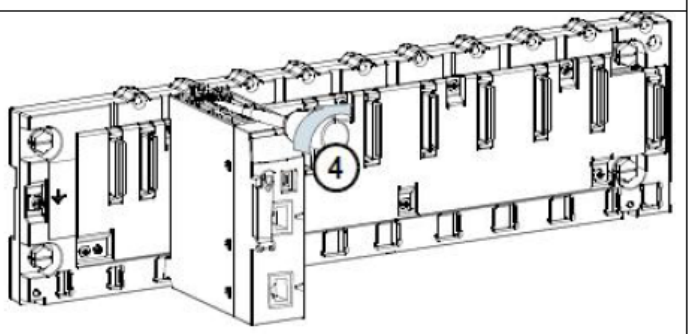
Etape	Action	Illustration
1	Vérifiez que l'alimentation est hors tension et qu'une carte mémoire adaptée est utilisée.	
2	Placez les deux ergots situés à l'arrière du module (dans la partie inférieure) dans les emplacements correspondants du rack.	
3	Faites pivoter le module vers le haut du rack de façon 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

2. Mise en œuvre et configuration du variateur ATV 71 sous Unity Pro :

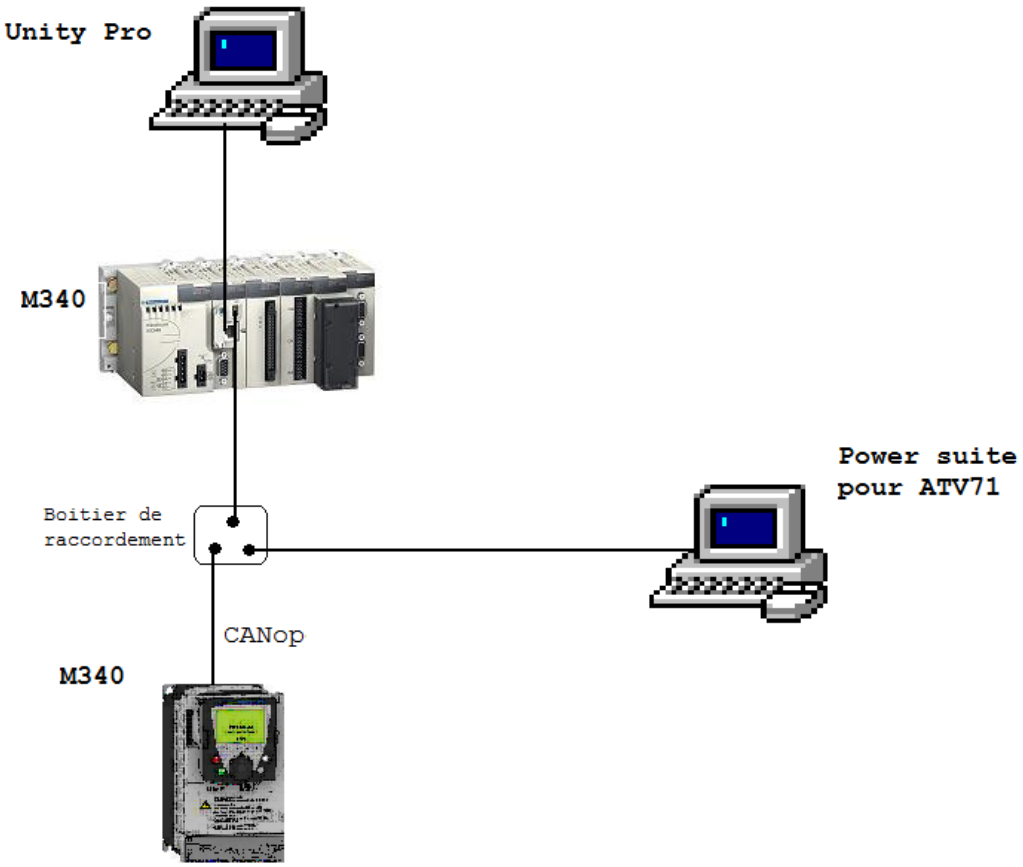


Figure D.10 : l'architecture utilisée dans l'application incluant un variateur ATV 71

2.1. Configuration requise :

En ce qui concerne la configuration logicielle requise présentée dans le guide de mise en route, PowerSuite est utilisé pour la configuration et le réglage de l'ATV 71.

PowerSuite pour Lexium 05 permet la mise en ligne de l'axe et garantit une méthode simple de configuration des paramètres d'un variateur Lexium 05. Il en va de même pour PowerSuite pour ATV 71, mais pour un variateur ATV 71.

Il est néanmoins possible de fonctionner sans PowerSuite dans certains cas en utilisant l'interface utilisateur du panneau avant du variateur ATV 71

Matériel	**Version minimale du logiciel**	**Version minimale du micro logiciel**
Modicon M340	Unity Pro V3.0	-
ATV 71	PowerSuite pour ATV 71 V2.00	**V1.1**

Tableau D.4 : les versions de matériel et de logiciel utilisées dans l'architecture permettant l'utilisation des MFB dans Unity Pro.

Matériel	Marque	Référence
Automate Modicon M340	Télémécanique	BMX P34 2030
Alimentation Modicon M340	Télémécanique	BMX CPS 2000
Rack Modicon M340	Télémécanique	BMX XBP 0800
Boîtier de raccordement CANopen entre le Modicon M340 et le variateur ATV 71	Télémécanique	VW3CANTAP2
Câble de programmation RJ45 avec adaptateur RS485/RS232 entre le boîtier de raccordement et le variateur	Télémécanique	ACC2CRAAEF030
Variateur ATV 71	Télémécanique	ATV71H075N2Z

Tableau D.5 : répertorie le matériel utilisé dans l'architecture permettant la mise en œuvre des MFB ATV 71 dans Unity Pro

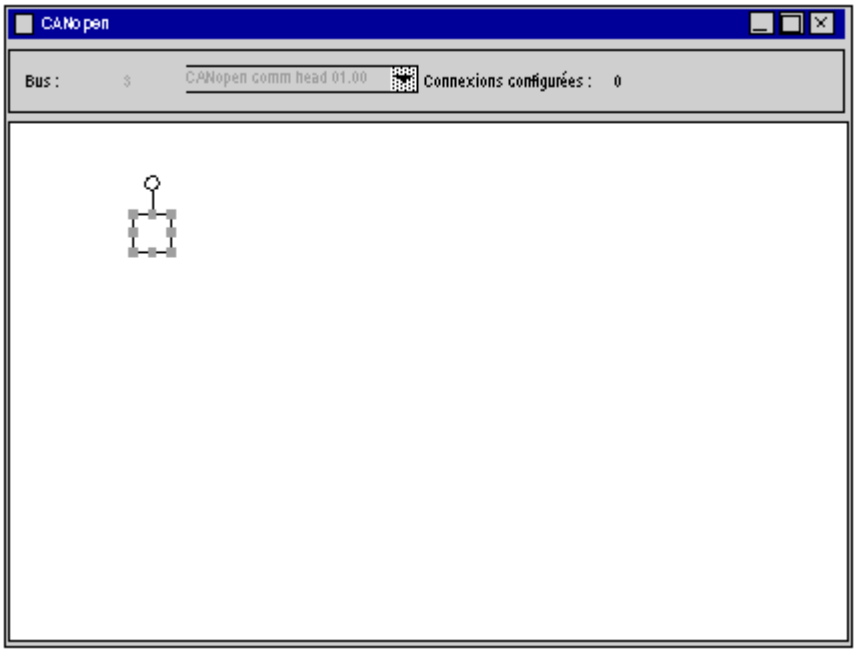
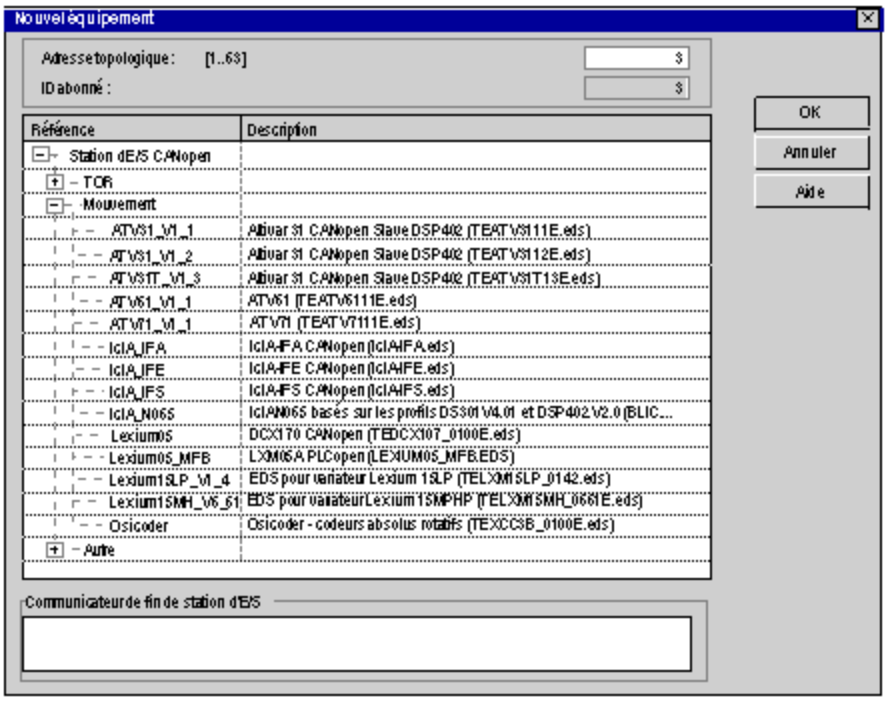
2.2. Configuration du bus CANopen ATV 71 :

Méthodologie de mise en œuvre d'un bus CANopen utilisant Modicon M340 :

- Configuration du port CANopen de l'UC
- Déclaration de l'esclave choisi dans le catalogue matériel (voir paragraphe ci-dessous),
- Configuration de l'esclave,
- Possibilité de configuration sous Unity Pro,
- Contrôle du bus CANopen dans le navigateur de projet.

Configuration de l'esclave CANopen

Etape	Action
1	Dans le navigateur de projet de Unity Pro, développez complètement le répertoire Configuration, puis cliquez deux fois sur CANopen. Résultat : La fenêtre CANopen s'affiche.

	
2	Sélectionnez Edition ® Nouvel équipement. Résultat : La fenêtre Nouvel équipement s'affiche : 

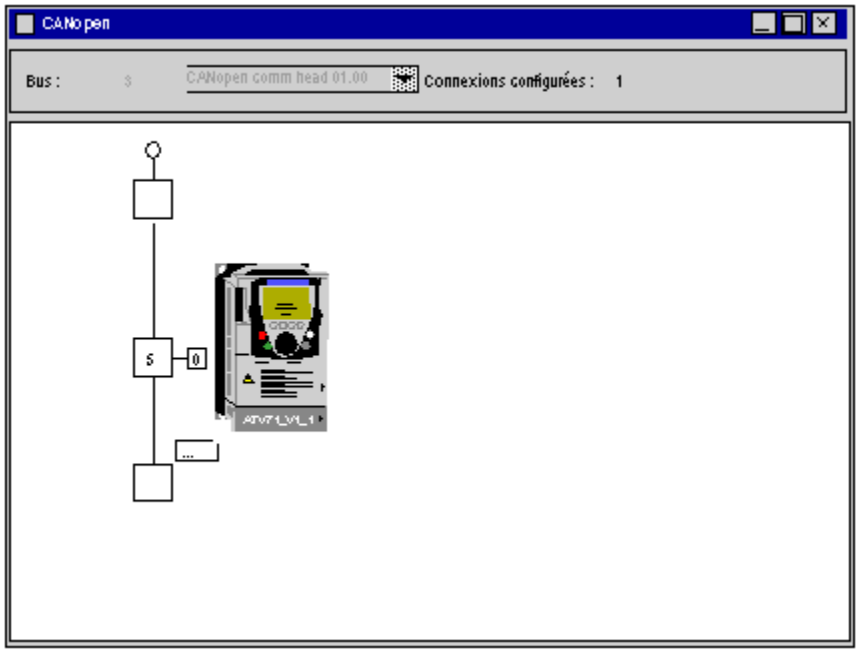
3	Définissez 5 dans l'adresse topologique. Pour l'équipement esclave, choisissez ATV71_V1_1.
4	Cliquez sur OK pour confirmer votre sélection. Résultat : La fenêtre CANopen s'affiche avec le nouvel équipement sélectionné. 
5	Sélectionnez Edition → Ouvrir le module. Si le MFB n'a pas encore été sélectionné, choisissez-le dans la zone Fonction.
6	Il vous sera demandé de valider vos modifications lors de la fermeture des fenêtres Equipement et CANopen.

Tableau D.6 : la procédure de configuration de l'esclave CANopen

3. Unité de mesure PM710 :

Le PM710 est une unité centrale de mesure pour les réseaux HT et BT.

Figure D.11 : Unité de mesure PM710

Le PowerLogic Power Meter série 700 PM710 offre toutes les fonctionnalités de mesure nécessaires pour surveiller une installation électrique dans un seul 96 x 96 mm unité ne s'étend que sur 50 mm derrière la surface de montage.

Avec son grand écran, il est possible de suivre tous les trois phases et le neutre en même temps. L'éclat d'affichage anti-grandes caractéristiques de 11 mm de haut et de caractères rétro-éclairage puissant pour une lecture facile même dans des conditions d'éclairage extrêmes et des angles de vision.

Le Power Meter série 700 est disponible en trois versions:

- PM700, version de base, avec une DHT et lectures min / max
- PM700P, version de base plus deux sorties d'impulsions pour comptage d'énergie
- PM710, version de base, plus un port RS 485 pour la communication Modbus.

Caractéristiques:

- Besoin de seulement 50 mm en arrière de la surface de montage
- Large écran rétro-éclairé avec les graphiques à barres intégré
- Utilisation intuitive

Il est possible de lire la puissance et de la demande actuelle, THD et min / max en version de base.

Classe d'énergie 1 tel que défini par la CEI 61036.

4. Relais de protection numérique Sepam S80 :

Le Sepam S80 est un relais de protection numérique pour la protection de courant et de tension, pour tout système de distribution.

Figure D.12 : Relais de protection Sepam S80

Le Sepam série 80 comprend 16 types de Sepam. Un type de Sepam est dédié à une seule application.

Caractéristiques:

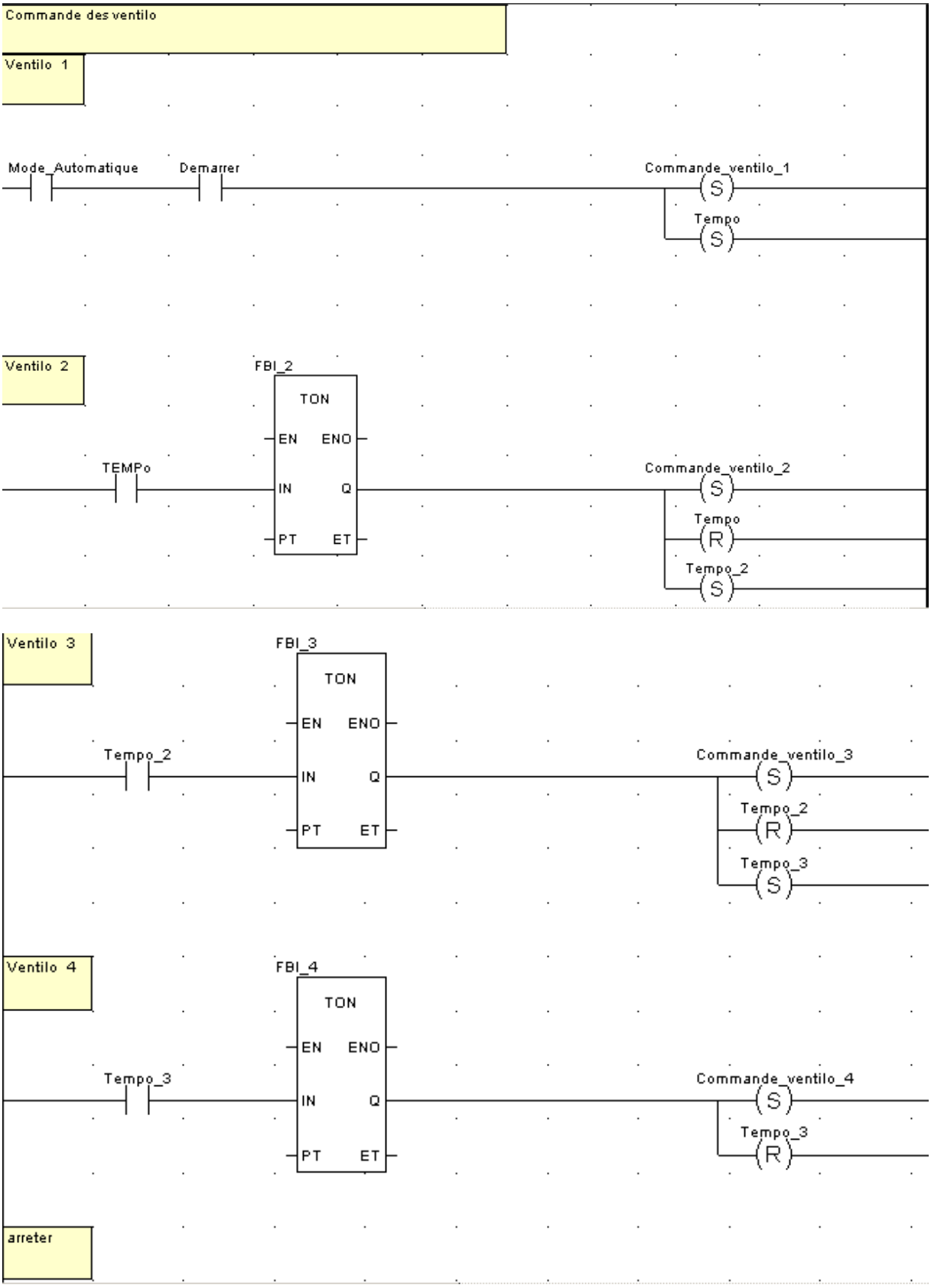
- 3 types de UMI
- 12 entrées analogiques:
- 8 entrées courant et de tension 4 entrées
- 4 entrées courant et de tension 8 entrées
- 42 entrées logiques
- 23 sorties relais
- 2 ports de communication Modbus
- 16 entrées capteur de température
- catridge mémoire amovible avec les paramètres et les paramètres de commissioning rapide après le remplacement

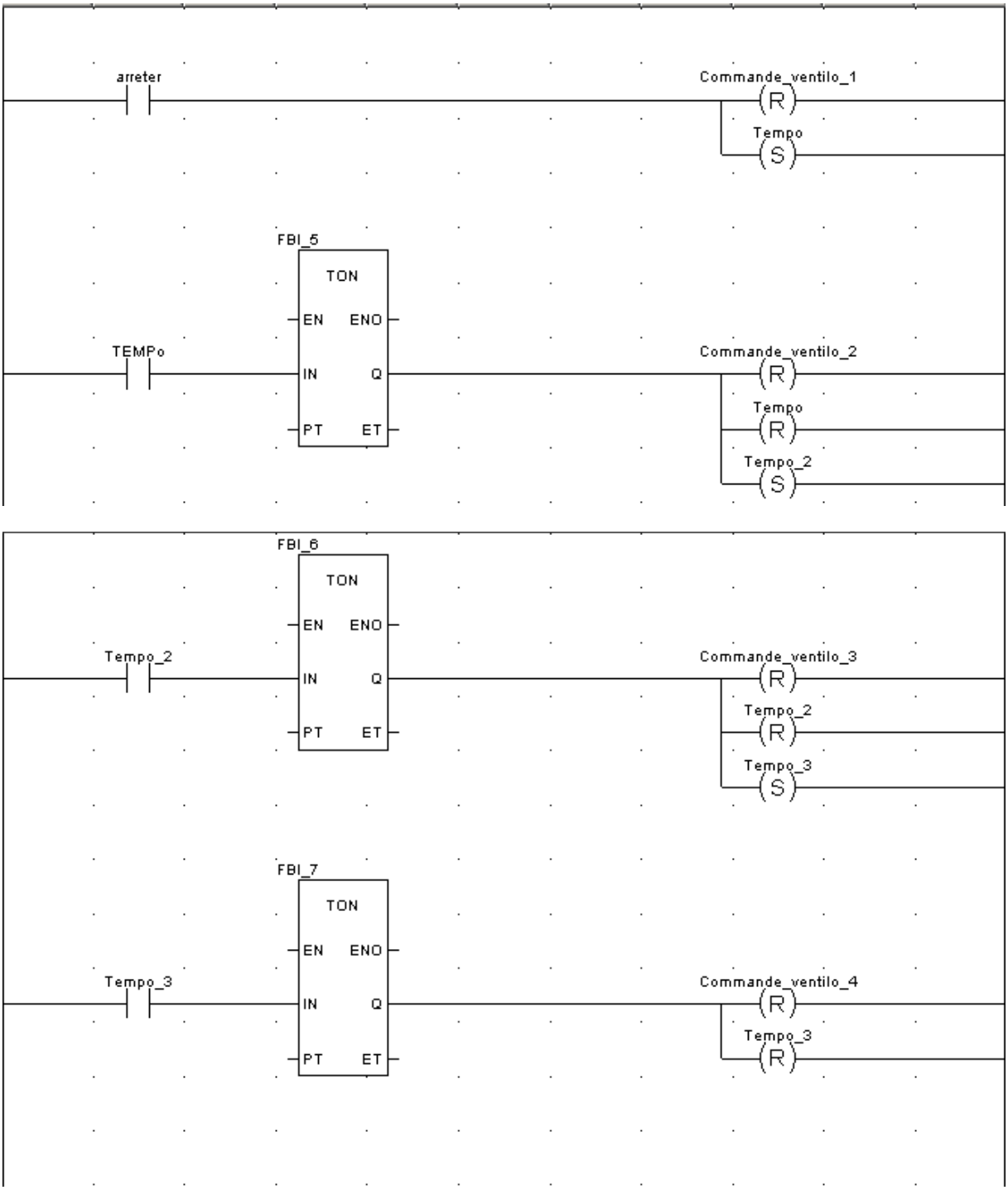
Il a un logiciel de programmation des fonctions spécifiques qui s'appelle Logipam.

5. Programmes Unity Pro:

5.1. Climatisation:

Démarrage des ventilateurs :


Commande des ventilo
Ventilo 1
Mode_Automatique
Demarrer
Commande_ventilo_1
S
Tempo
S
Ventilo 2
FBI_2
TON
EN
ENO
TEMPo
IN
Q
PT
ET
Commande_ventilo_2
S
Tempo
R
Tempo_2
S
Ventilo 3
FBI_3
TON
EN
ENO
Tempo_2
IN
Q
PT
ET
Commande_ventilo_3
S
Tempo_2
R
Tempo_3
S
Ventilo 4
FBI_4
TON
EN
ENO
Tempo_3
IN
Q
PT
ET
Commande_ventilo_4
S
Tempo_3
R
arreter



Commande des volets :

```
                                    (**********Programme commande des vloet*****)

                    (**********************Dans le mode manuel il a besoin d'une consigne

if mode manuel= true  then

if (t interne<Consigne t) then set(V rouge);
                                  Reset(V vert);

end_if;


if (t interne>Consigne t) then


set(v Vert);

reset(v rouge);

diff:=t interne-consigne t;

                  if (diff>1 and diff <2) then angle volet:=30; end_if;

                  if (diff>2 and diff <3) then angle volet:=45; end_if;

                  if (diff>3 and diff <5) then angle volet:=60; end_if;

                  if (diff>5)             then angle volet:=80; end_if;

end_if;


end_if;
```

```
                (******dans le mode automatique il va considérer la température externe comme une Consigne*


if (Mode Automatique=true and Mode Manuel=false) then (*************Mode manuel est prioritaire****)

if (t externe>t interne) then set(v vert);  (****pour montrer que le système est en marche**)
                         reset(v rouge);    (*le v_rouge si la climatisation n'est pas nécessaire***)

diff:=t externe-t interne;


                if (diff>1 and diff <3) then angle volet:=30; end_if;

                if (diff>3 and diff <5) then angle volet:=45; end_if;

                if (diff>5 and diff <7) then angle volet:=60; end_if;

                if (diff>7)             then angle volet:=80; end_if;

end_if;
```

5.2. Gestion d'electricité :

```
For i:=1 to 11 do

if (courant batiment[i] > consigne courant[i]) then Reset( disjoncteur batiment[i]);
                                                    Set (Alarme courant);
end_if;

if (tension batiment[i] > consigne tension[i]) then Reset( disjoncteur batiment[i]);
                                                    Set (Alarme tension);
end_if;

end_for;
```

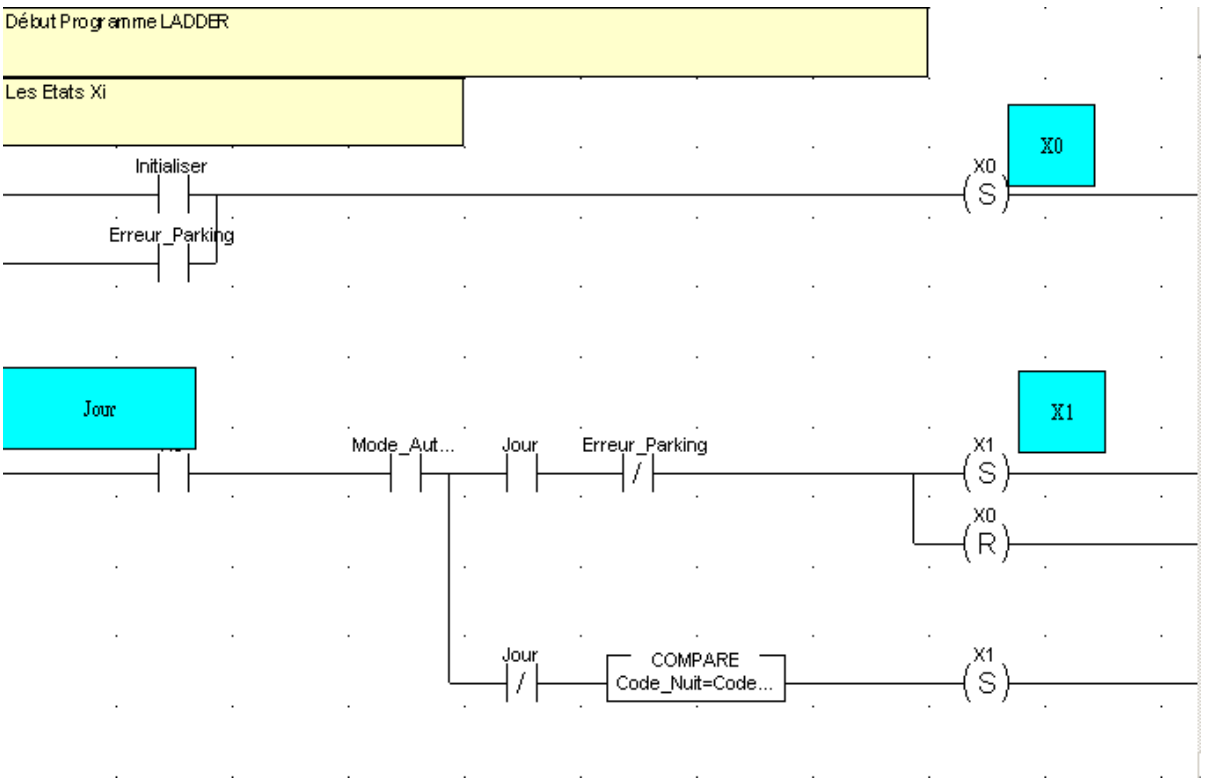
5.3. Parking :

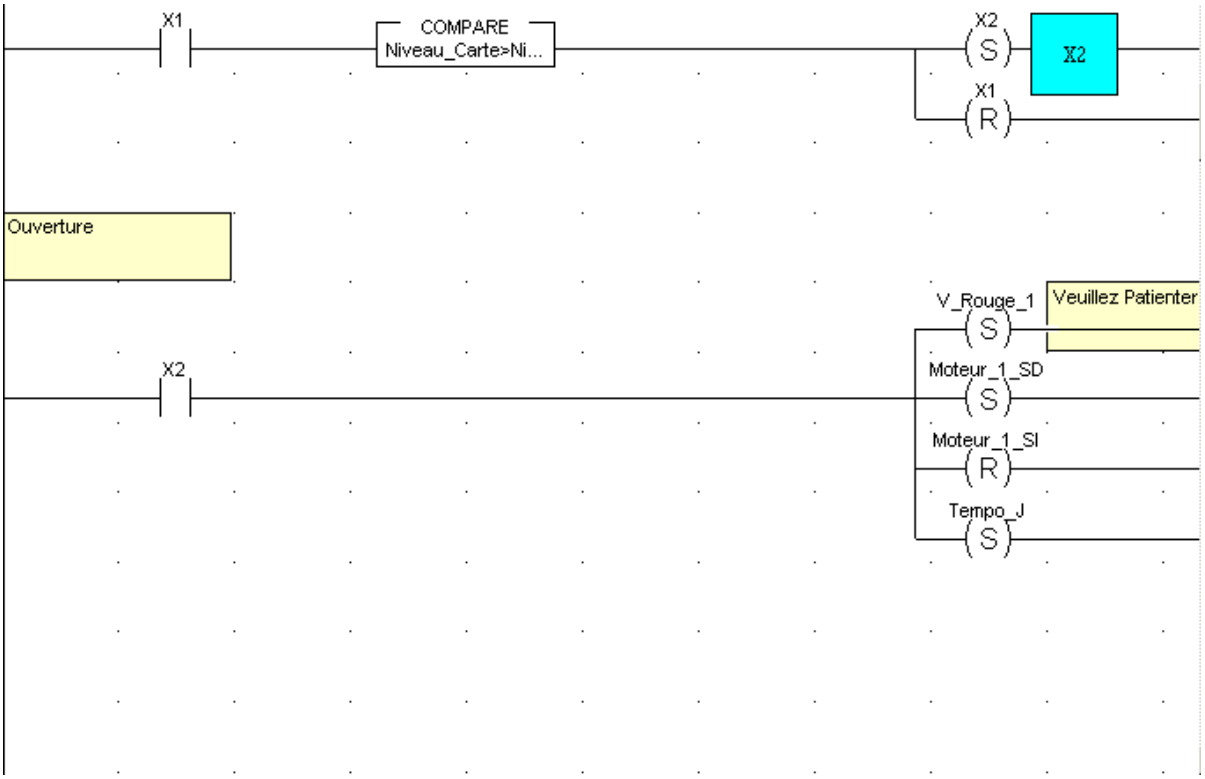
Nombre de voitures :

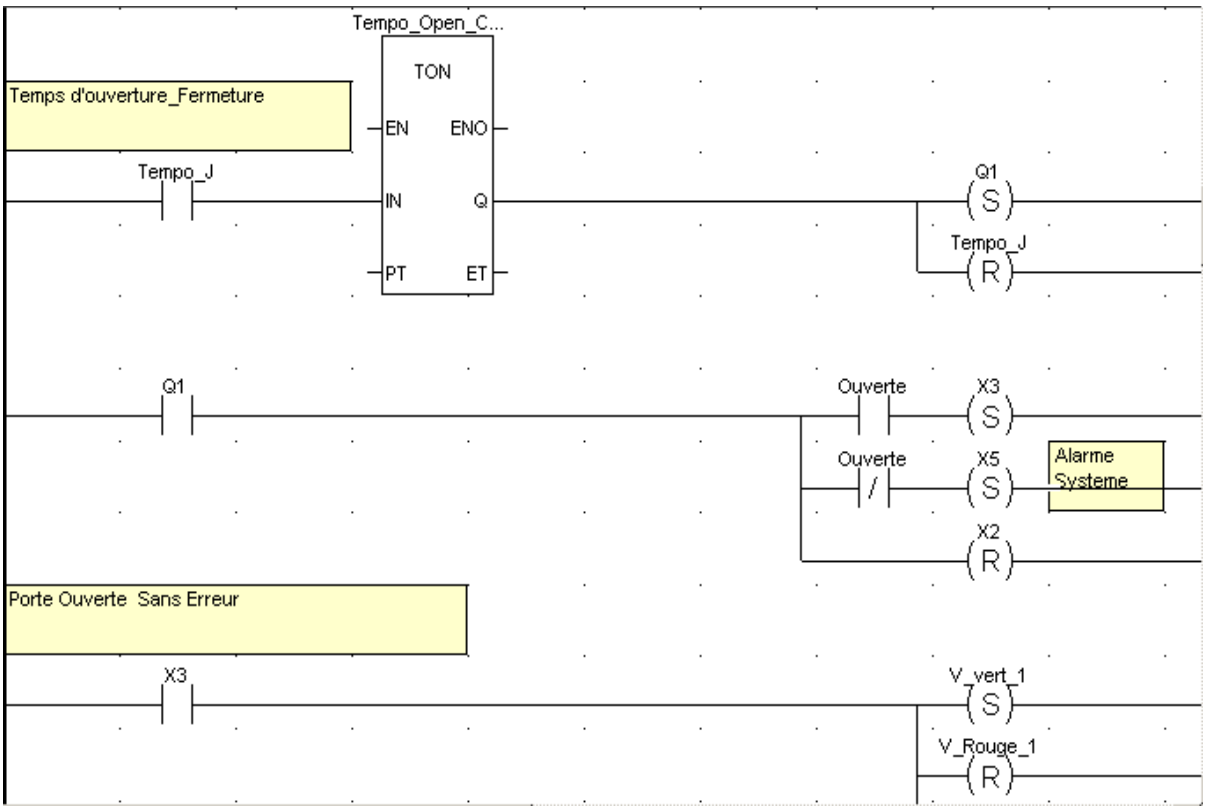
```
N1:=0;
N2:=0;

                                                (* Incrémentation du compteur*)

For i:=1 to 20 do

                                                (*Vérification de Présence de voiture

if Capteur_Presence[i]= true

then

        if (Detecteur_Co2[i]<Seuil_Co2)         (* Vérification voiture mal garrée*)

        then Position[i]:=1;

                        if (i<7) then N1:=N1+1;
                        else N2:=N2+1;
                        end_if ;

        else position[i]:=2 ;

        end_if;

else  Position[i]:=0;

end_if;


end_for;
```

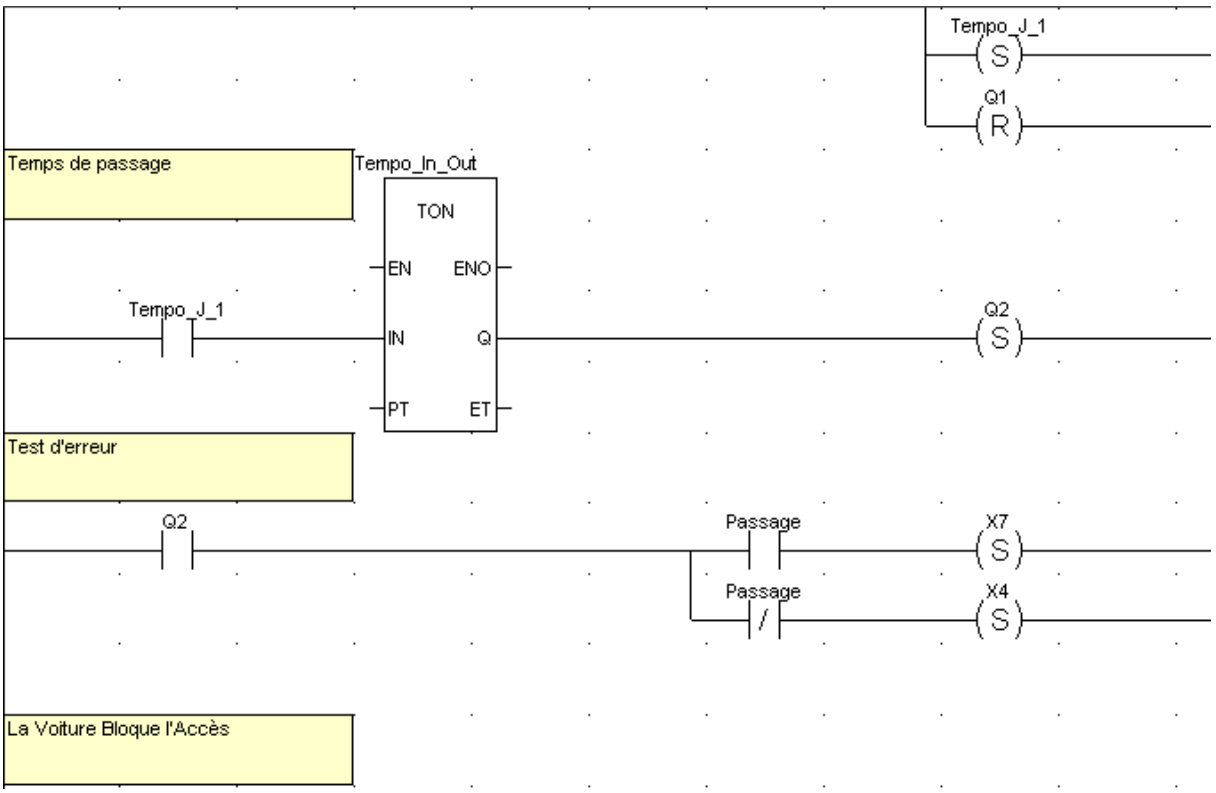
```
Nbr_Voiture_Parking_1:=N1;
Nbr_Voiture_Parking_2:=N2;
Nbr_V_Mal_G_1:=6-Nbr_Voiture_Parking_1;
Nbr_V_Mal_G_2:=14-Nbr_Voiture_Parking_2;
```

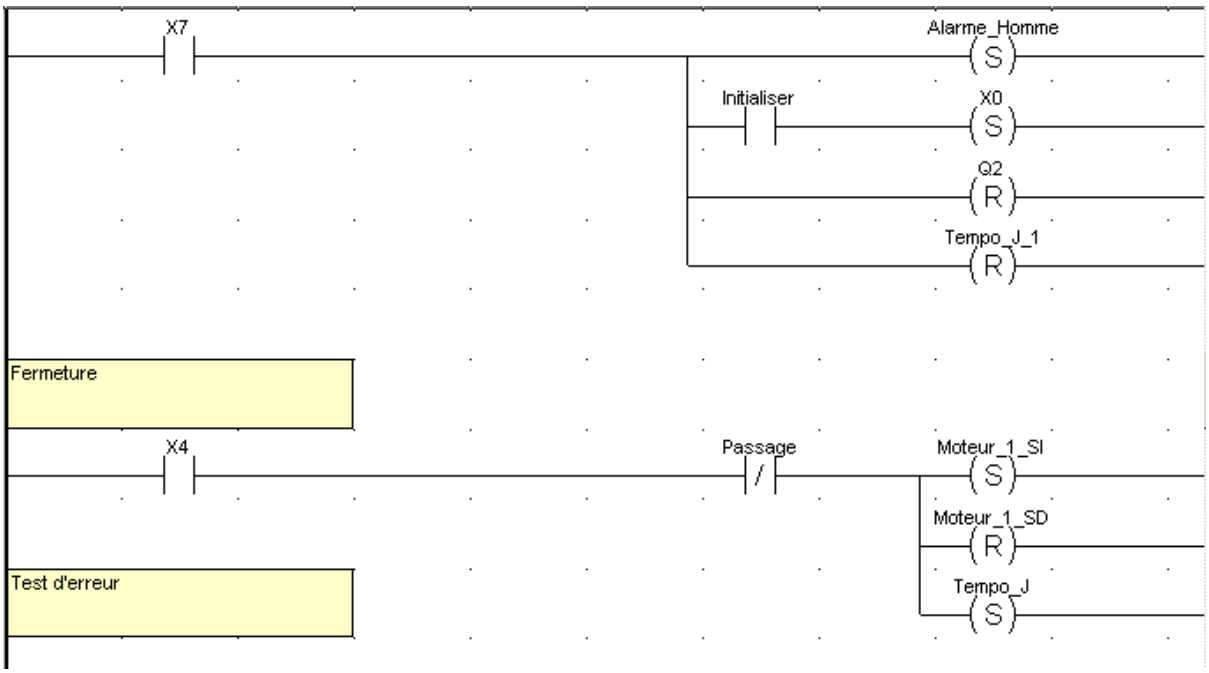
Contrôle d'accès parking :

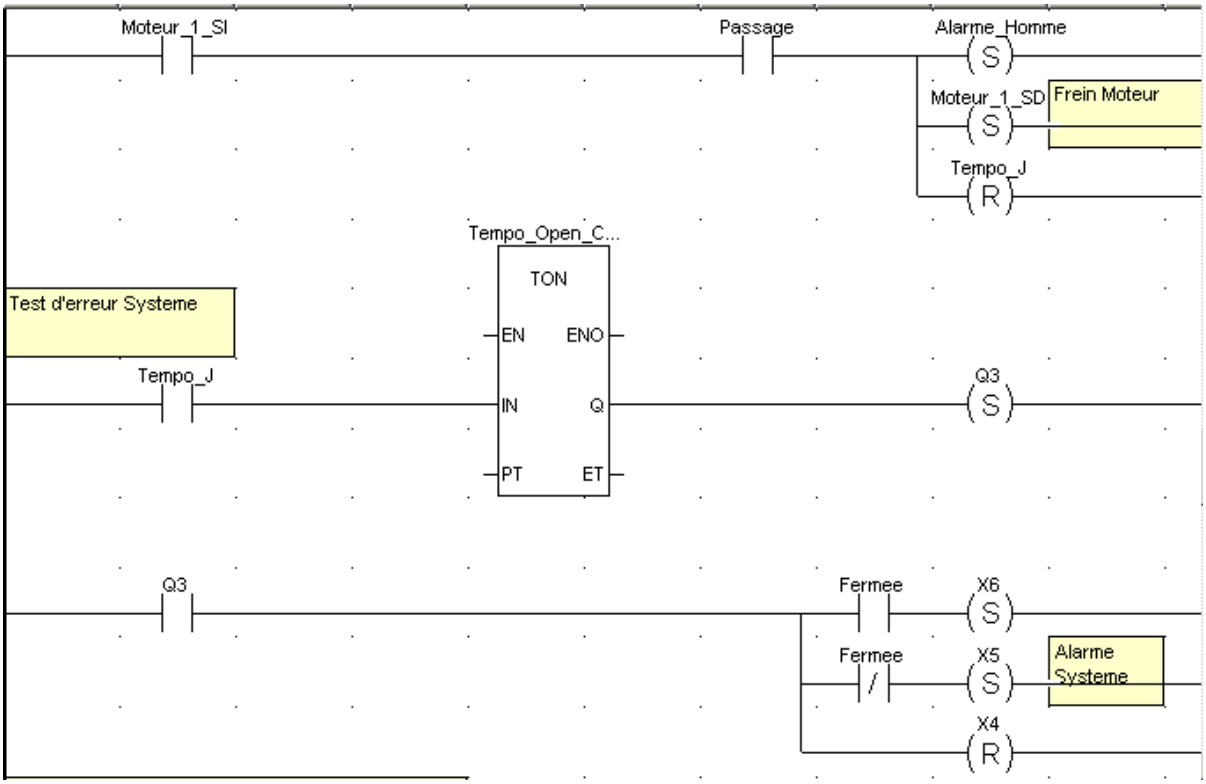


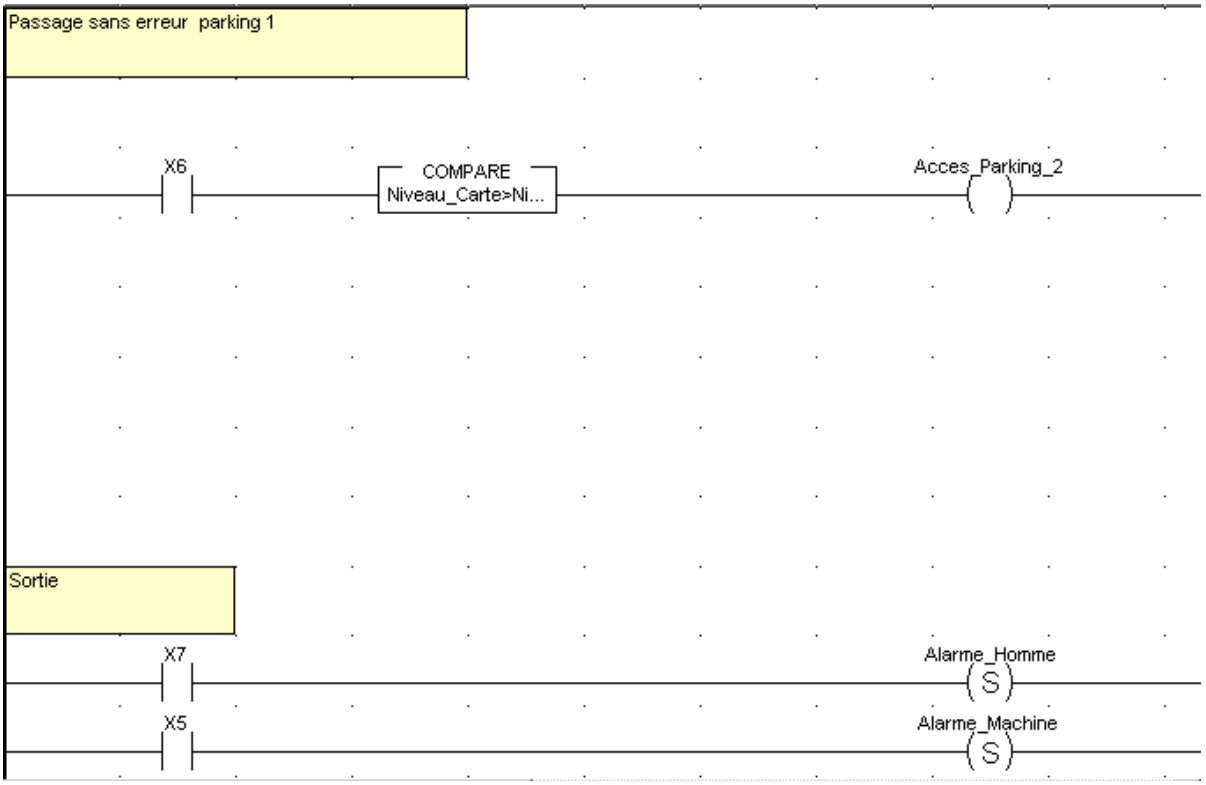

X1
COMPARE
Niveau_Carte>Ni...
X2
S
X2
X1
R
Ouverture
V_Rouge_1
Veuillez Patienter
S
X2
Moteur_1_SD
S
Moteur_1_SI
R
Tempo_J
S

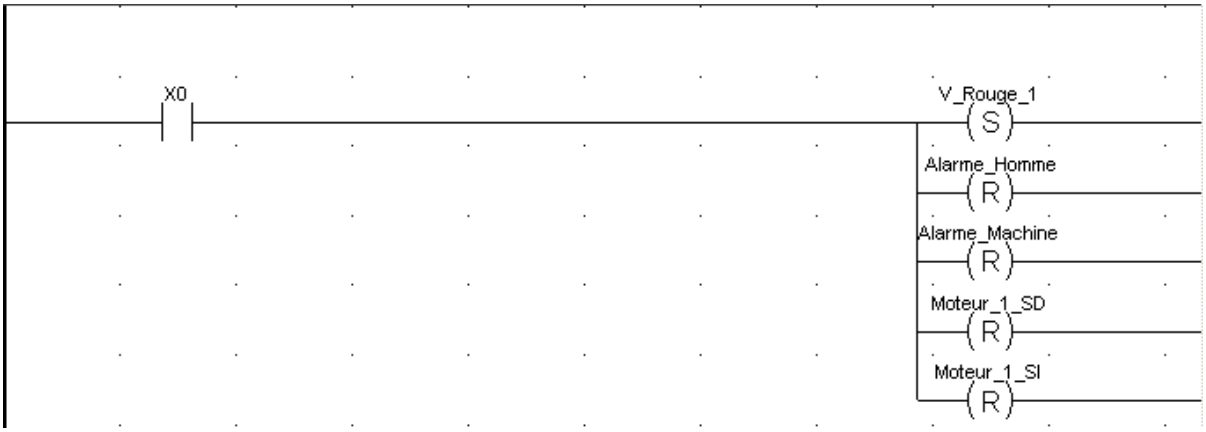

Tempo_Open_C...
TON
Temps d'ouverture_Fermeture
EN
ENO
Tempo_J
IN
Q
Q1
S
Tempo_J
PT
ET
R
Q1
Ouverte
X3
S
Ouverte
X5
S
Alarme
Systeme
X2
R
Porte Ouverte Sans Erreur
X3
V_vert_1
S
V_Rouge_1
R


Tempo_J_1
S
Q1
R
Temps de passage
Tempo_In_Out
TON
EN
ENO
Tempo_J_1
IN
Q
Q2
S
PT
ET
Test d'erreur
Q2
Passage
X7
S
Passage
X4
S
La Voiture Bloque l'Accès


X7
Alarme_Homme
S
Initialiser
X0
S
Q2
R
Tempo_J_1
R
Fermeture
X4
Passage
Moteur_1_SI
S
Moteur_1_SD
R
Test d'erreur
Tempo_J
S

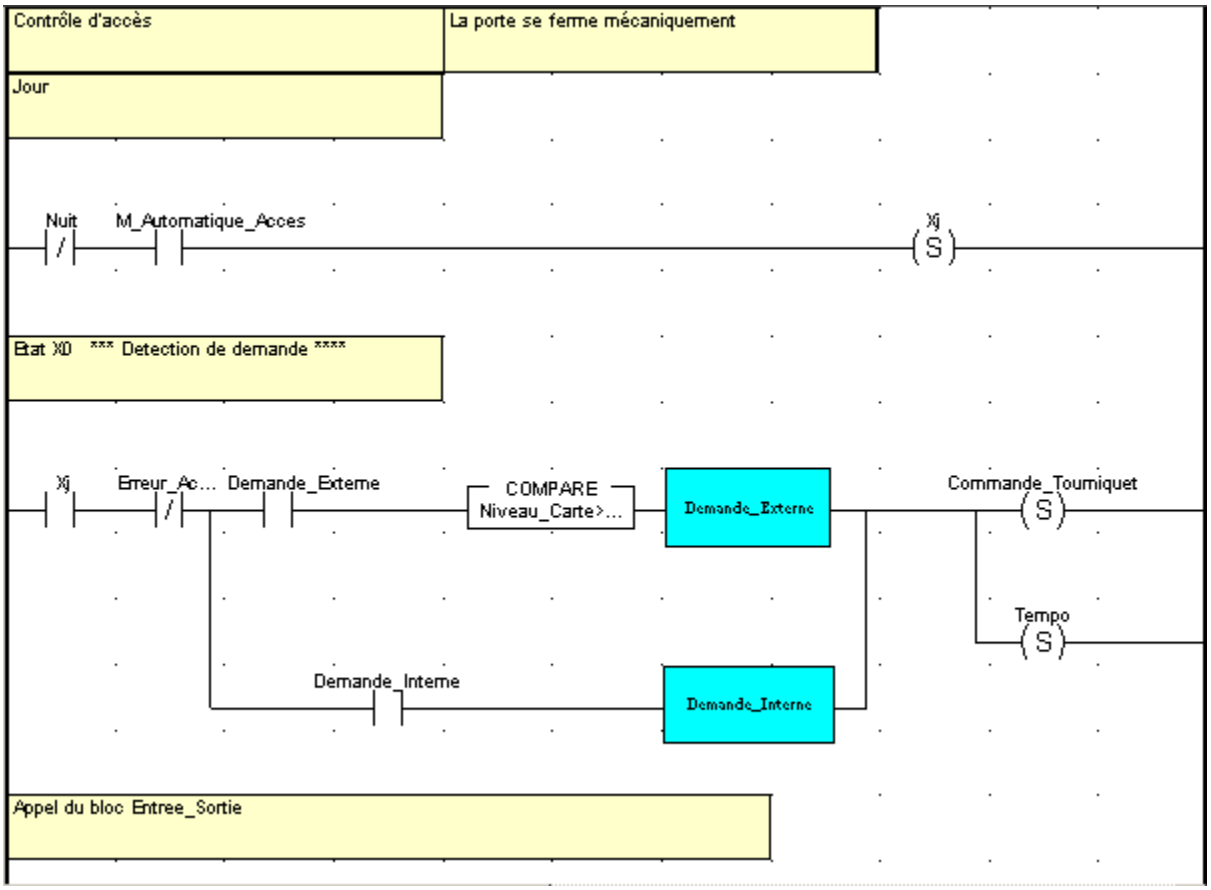

Moteur_1_SI
Passage
Alarme_Homme
S
Moteur_1_SD
Frein Moteur
Tempo_J
R
Tempo_Open_C...
TON
EN
ENO
Test d'erreur Systeme
Tempo_J
IN
Q
Q3
PT
ET
Q3
Fermee
X6
Fermee
X5
Alarme Systeme
X4

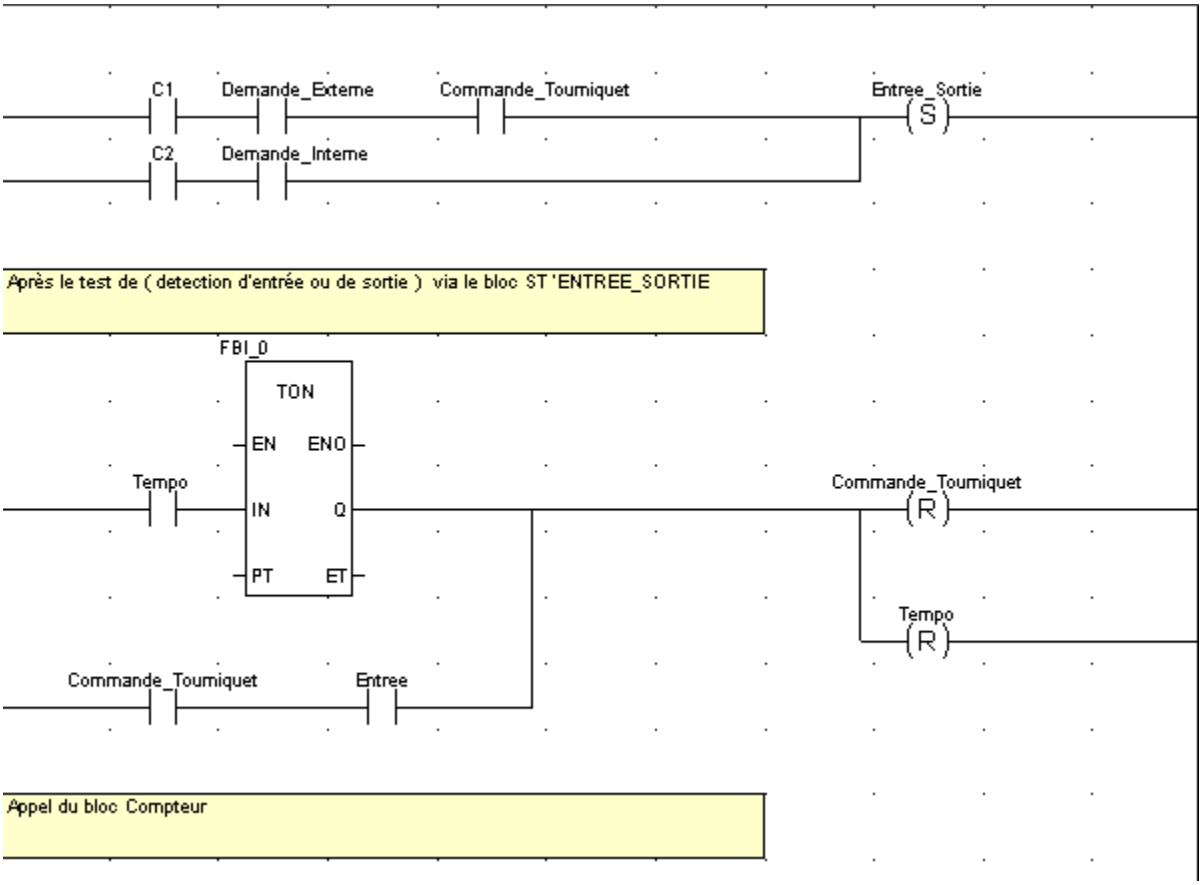

Passage sans erreur parking 1
X6
COMPARE
Niveau_Carte>Ni...
Acces_Parking_2
Sortie
X7
Alarme_Homme
X5
Alarme_Machine

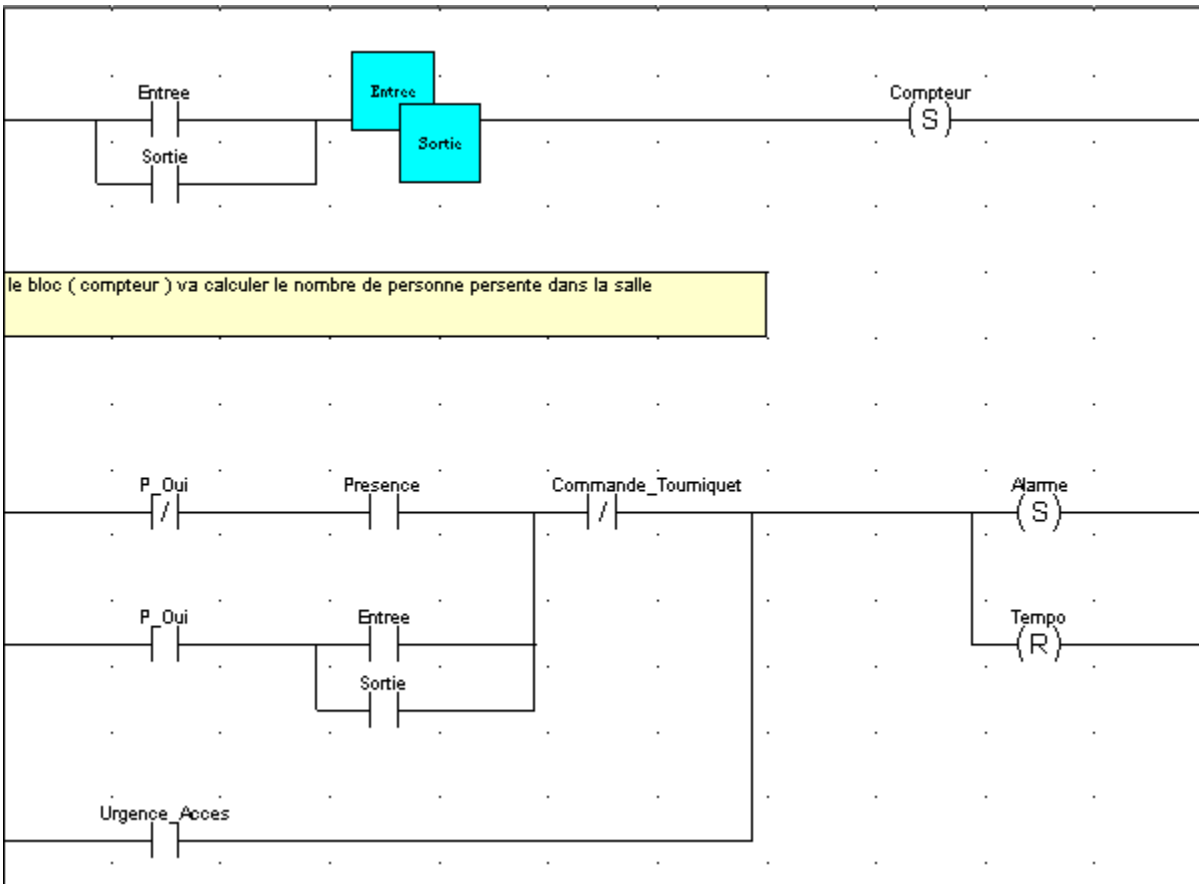


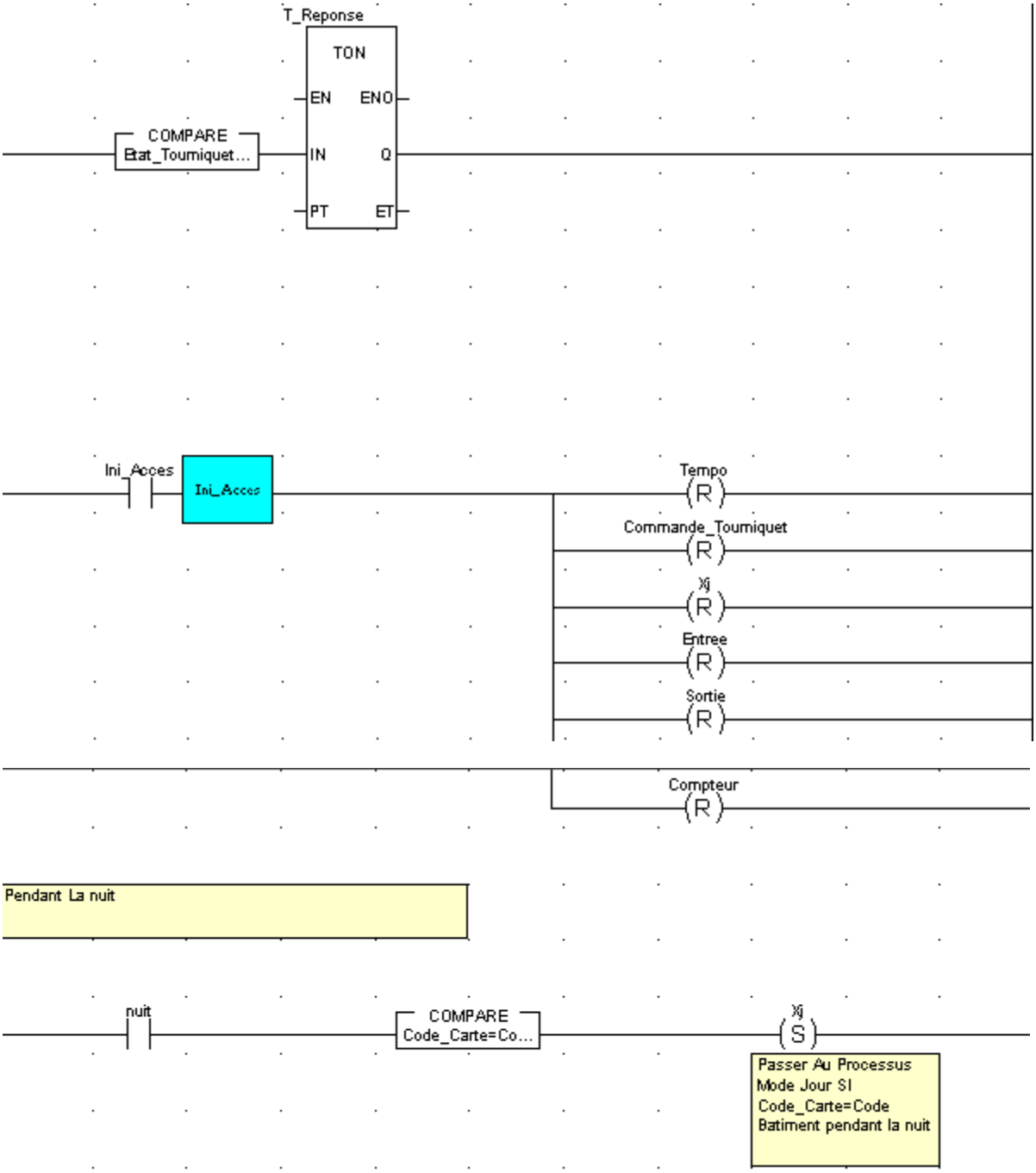
5.4. Contrôle d'accès :

Gestion d'accès étage :




C1
Demande_Externe
Commande_Tourniquet
Entree_Sortie
S
C2
Demande_Interne
Après le test de (detection d'entrée ou de sortie) via le bloc ST 'ENTREE_SORTIE
FBI_0
TON
EN
ENO
Tempo
IN
Q
PT
ET
Commande_Tourniquet
R
Tempo
R
Commande_Tourniquet
Entree
Appel du bloc Compteur


Entree
Sortie
Entree
Sortie
Compteur
S
le bloc (compteur) va calculer le nombre de personne persente dans la salle
P_Oui
Presence
Commande_Tourniquet
Alarme
S
P_Oui
Entree
Sortie
Tempo
R
Urgence_Acces


T_Reponse
TON
EN
ENO
COMPARE
Etat_Tourniquet...
IN
Q
PT
ET
Ini_Acces
Ini_Acces
Tempo
R
Commande_Tourniquet
R
Xj
R
Entree
R
Sortie
R
Compteur
R
Pendant La nuit
nuit
COMPARE
Code_Carte=Co...
Xj
S
Passer Au Processus
Mode Jour SI
Code_Carte=Code
Batiment pendant la nuit

Compteur :

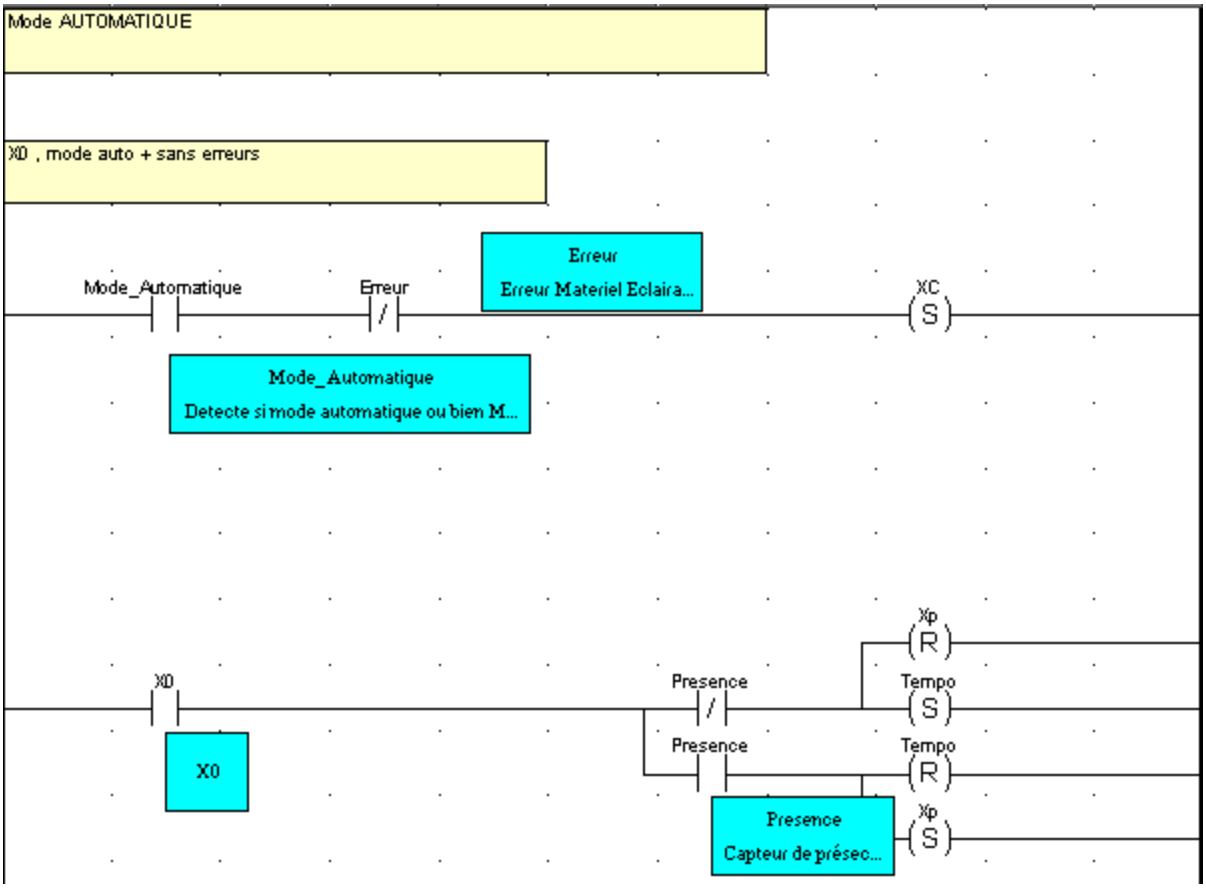
```
IF Compteur=True   then
                        (********Initialisation du nombre de personne presente dans la sall

        if (P_oui =false ) then Nbr_personne_salle:=0 ;  end_if ;

                        (********Incrémentation du compteur en cas d'entree dans la salle**
        If (Entree=true  and Sortie=false)
                then nbr_personne_salle:=nbr_personne_salle+1 ;
        End_if ;

                        (********Décrentation du compteur en cas de sortie de la salle*****
        If (Sortie=True and Entree=false)
                then    if nbr_personne_salle =0
                                        then Alarme:=true ;  (*Alarme si compteur=0 et sor
                        Else nbr_personne_salle:=nbr_personne_salle-1;
                        end_if ;
        End_if ;
                        (*******Mémoirisation de la presence d'un ou plusieurs personnes***
        if nbr_personne_salle>0 then P_Oui:=True;

        Else P_oui:=False ;

        End_if;
End_if ;
                                        (******Fin Programme*******)
```

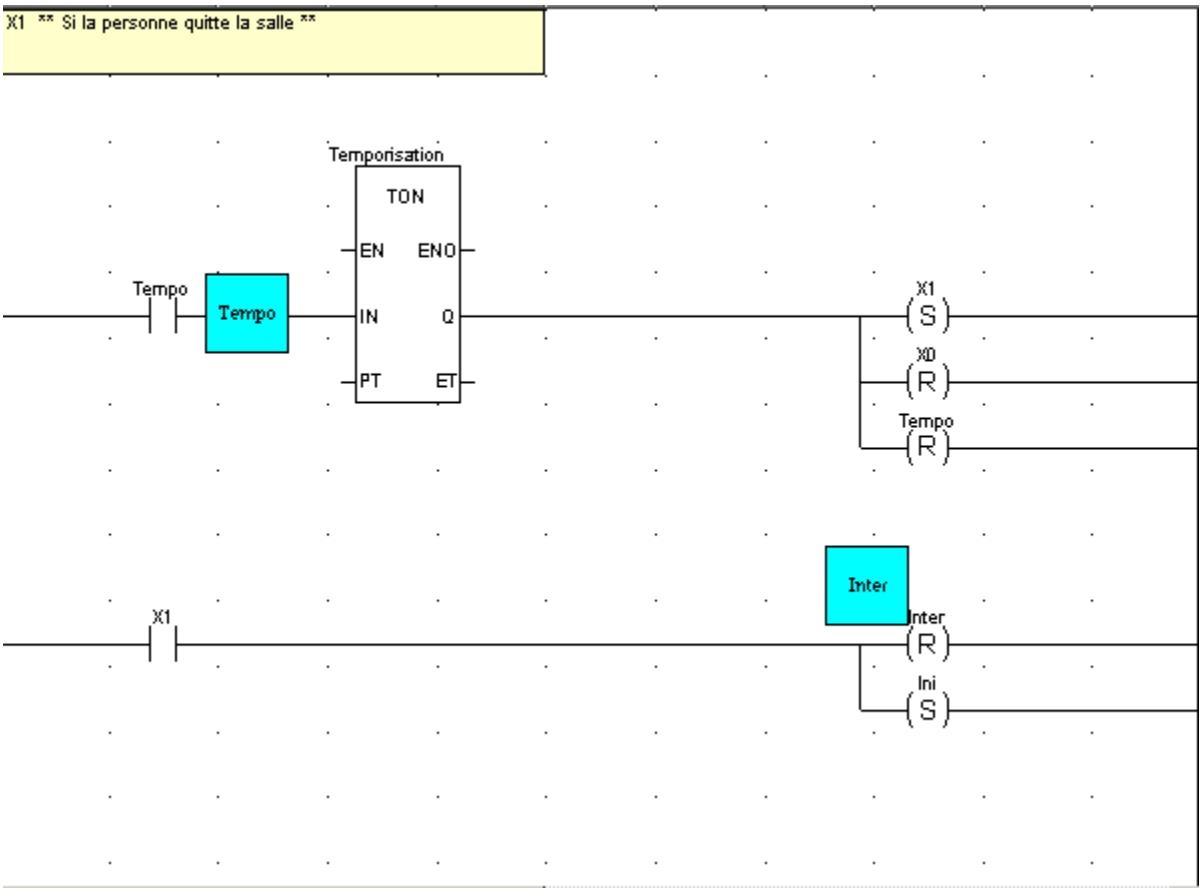
Entrée ou Sortie :

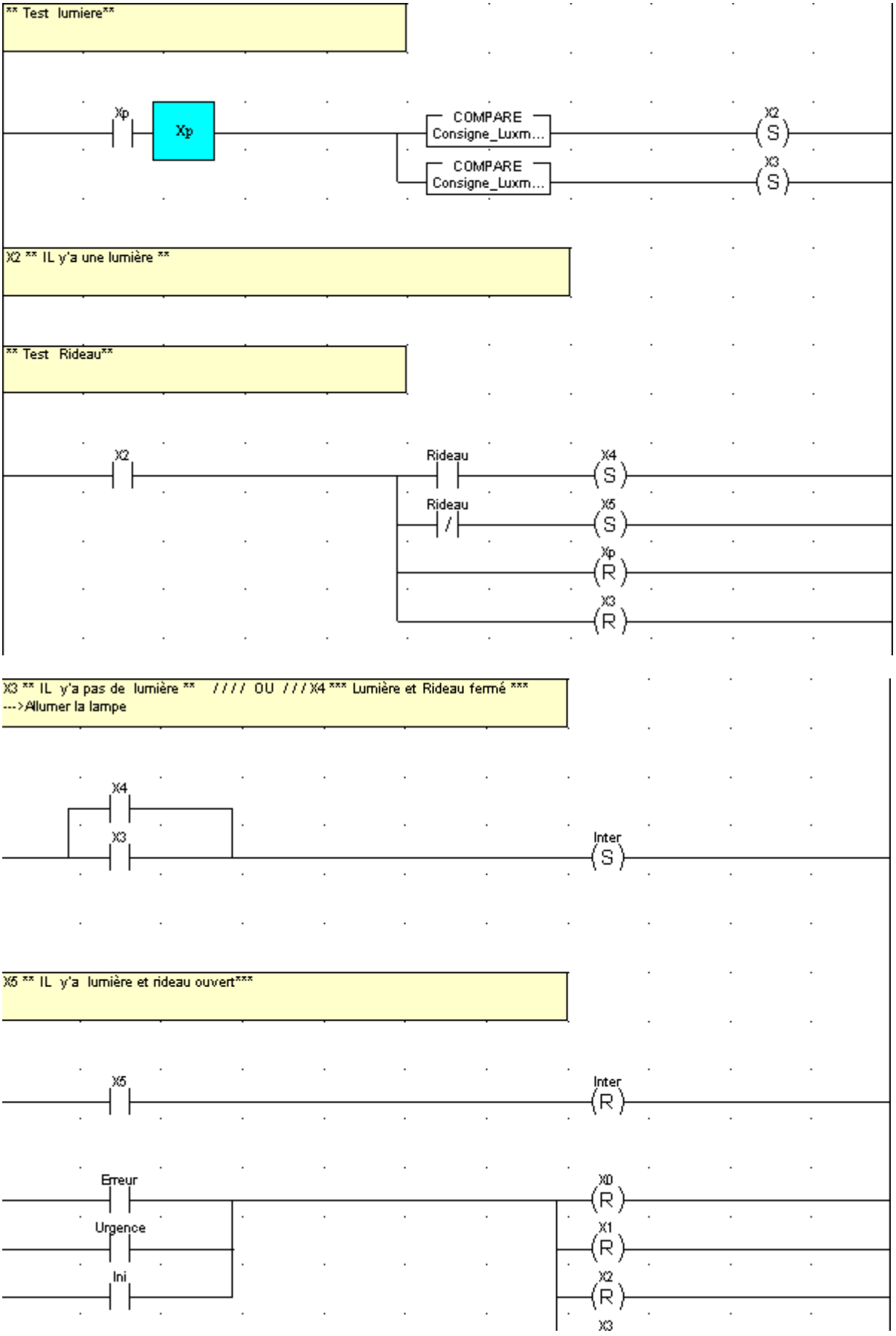
```
                                (*******Debut Programme*******)


                            (*********** Detection si entrée ou sortie*********)

if Entree_Sortie=true  then


                    if (C1=true and C2=false )

                            then P_Exterieur :=true ;
                    End_if;

                    if (P_Exterieur=true and C1=False and C2=True )

                            then Entree:=True ;

                    End_if;



                    if (C2=true and C1=false )

                            then P_Interieur :=true ;
                    End_if;

                    if (P_Interieur=true and C1=False and C2=True )

                            then Sortie:=True ;

                    End_if;
```

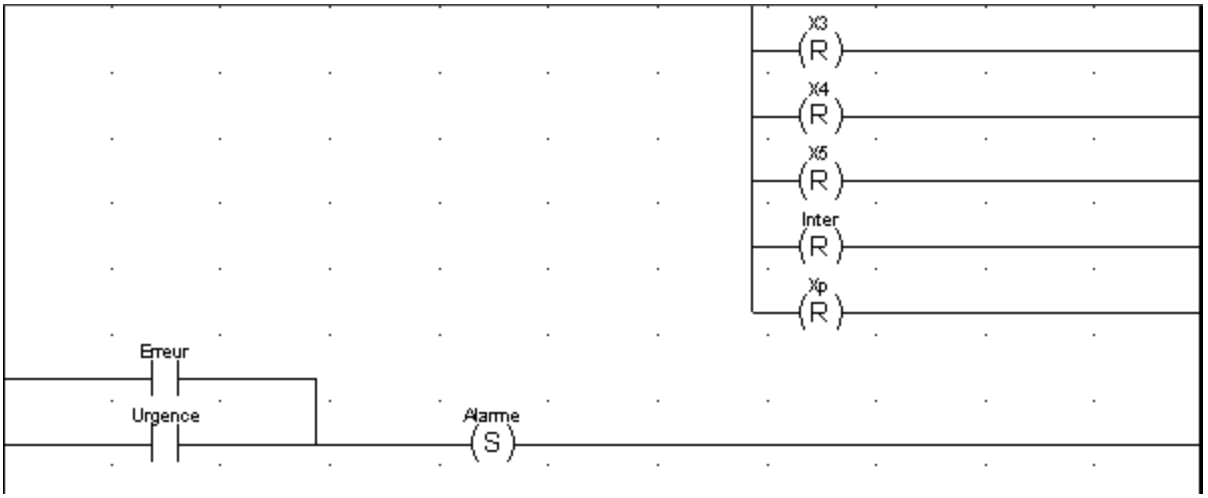
```
End_if ;


                                (***********Fin Programme*********)
```

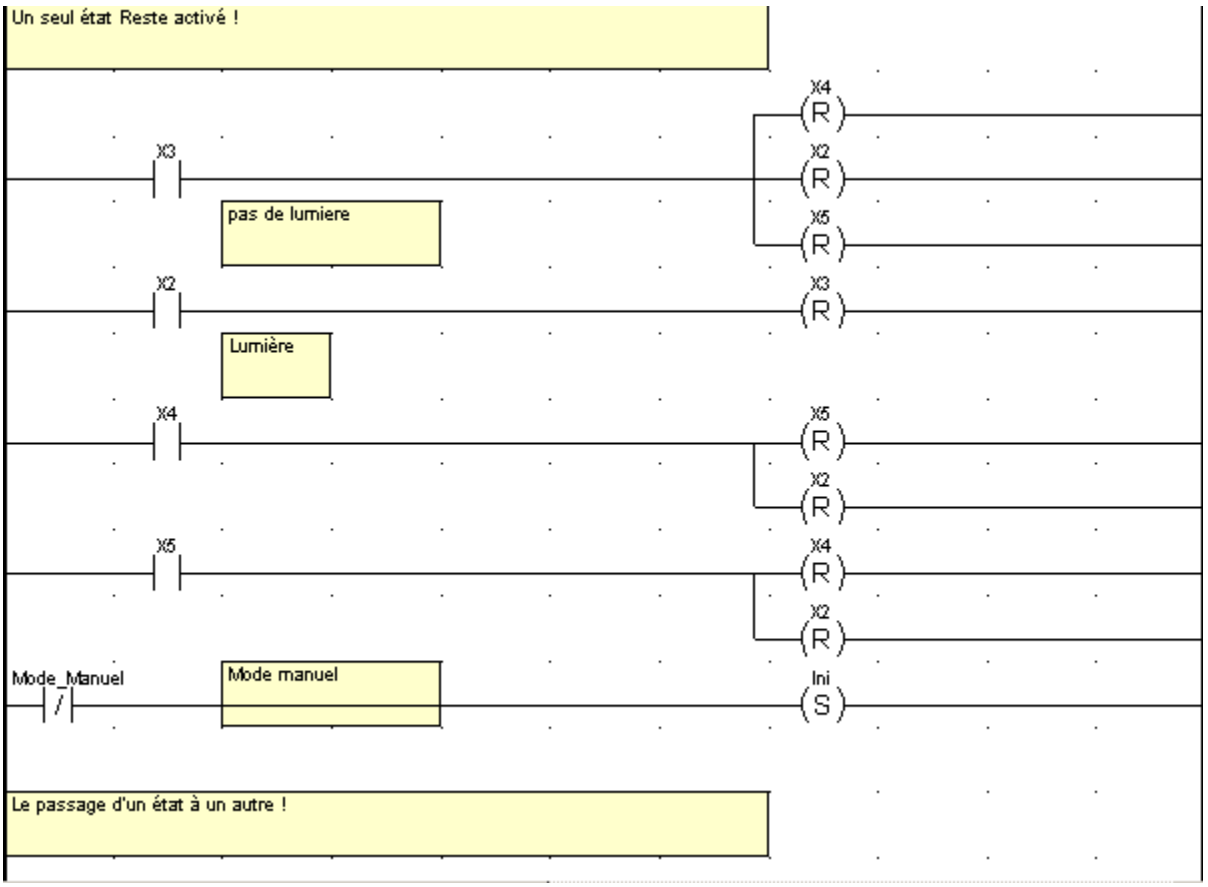
5.5. Gestion d'eclairage :

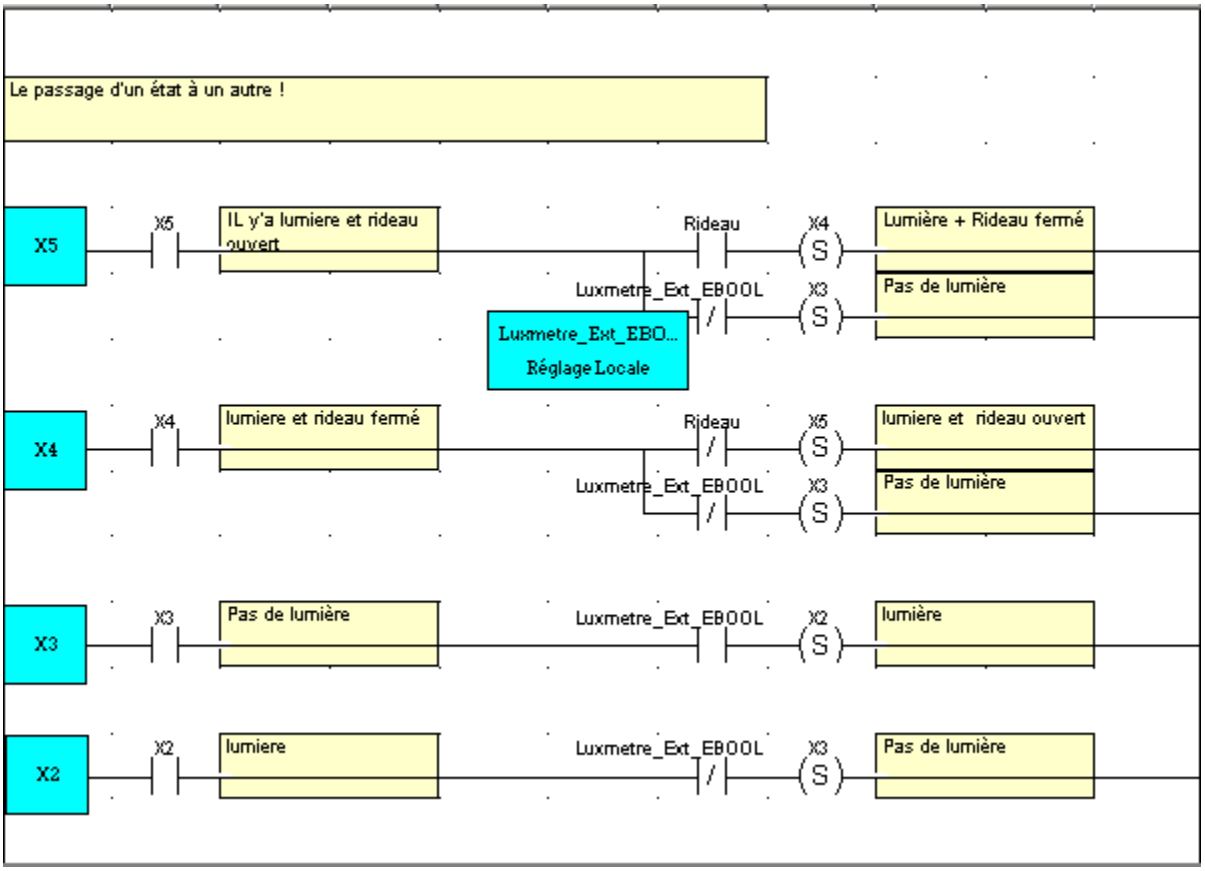

Mode AUTOMATIQUE
X0 , mode auto + sans erreurs
Mode_Automatique
Erreur
Erreur
Erreur Materiel Eclaira...
XC
S
Mode_Automatique
Detecte si mode automatique ou bien M...
Xp
R
X0
Presence
Tempo
S
Presence
Tempo
R
Presence
Capteur de présec...
Xp
S
X0


X1 ** Si la personne quitte la salle **
Temporisation
TON
EN
ENO
Tempo
Tempo
IN
Q
PT
ET
X1
S
X0
R
Tempo
R
Inter
X1
Inter
R
Ini
S


** Test lumiere**
Xp
Xp
COMPARE
Consigne_Luxm...
COMPARE
Consigne_Luxm...
X2
S
X3
S
X2 ** IL y'a une lumière **
** Test Rideau**
X2
Rideau
X4
S
Rideau
X5
S
Xp
R
X3
R
X3 ** IL y'a pas de lumière ** //// OU ///X4 *** Lumière et Rideau fermé ***
--->Allumer la lampe
X4
X3
Inter
S
X5 ** IL y'a lumière et rideau ouvert***
X5
Inter
R
Erreur
Urgence
Ini
X0
R
X1
R
X2
R
X3


X3
R
X4
R
X5
R
Inter
R
Xp
R
Erreur
Urgence
Alarme
S


Un seul état Reste activé !
X4
R
X3
X2
R
pas de lumiere
X5
R
X2
X3
R
Lumière
X4
X5
R
X2
R
X5
X4
R
X2
R
Mode_Manuel
Mode manuel
Ini
S
Le passage d'un état à un autre !



5.6. Les blocs d'erreurs :

Erreur générale :

```
                        (******Programme pour Détecter l'Erreur*****)

                                    (**** Begin *****)

                        (************Type de L'Erreur***********)

                            (******Erreur d'éclairage*******)

if (Erreur_Rideau=true or  Erreur_Lumiere=True or Erreur_Disjoncteur=True or Erreur_Presenc
        then Erreur_Eclairage:=True ;
Else Erreur_Eclairage:=False;
END_if;

                            (********** Erreur d'acces*******)

if(Erreur_Tourniquet=True or Erreur_Carte=true or Erreur_Alarme=True )
        then Erreur_Acces:=True ;
Else Erreur_Acces:=False ;
END_if;

                        (**********Erreur de climatisation*****)

                        (**********Erreur de climatisation*****)

if(Erreur_Variateur=true or Erreur_Volet=True or Erreur_Temperature )
        then Erreur_climatisation:=true ;
Else Erreur_Climatisation:=False;
END_if;

                        (******* Traduction de l'Erreur*********)

if      Erreur_Rideau=true       then R:=1;  else R:=0;   end_if ;
if      Erreur_Lumiere=true      then L:=1;  else L:=0;   end_if ;
if      Erreur_Disjoncteur=true  then D:=1;  else D:=0;   end_if ;
if      Erreur_Presence=true     then P:=1;  else P:=0;   end_if ;
if      Erreur_Tourniquet=true   then Tr:=1; else Tr:=0;  end_if ;
if      Erreur_Alarme=true       then A:=1;  else A:=0;   end_if ;
if      Erreur_Carte=true        then C:=1;  else C:=0;   end_if ;
if      Erreur_Variateur=true    then Vr:=1; else Vr:=0;  end_if ;
if      Erreur_Volet=true        then Vl:=1; else Vl:=0;  end_if ;
if      Erreur_Temperature=true  then Tr:=1 ;else Tr:=0;  end_if ;


                        (*********Niveau de l'Erreur**********)

N_E_S := R+L+D+P+Tr+A+C+Vr+C+Vl ;
```

```
                        (*********Niveau de l'Erreur**********)

N_E_S := R+L+D+P+Tr+A+C+Vr+C+Vl ;

                         (**********En cas d'urgence********)

if (Urgence= true or Erreur_Eclairage=true or Erreur_Acces=true or Erreur_Climatisation=tru
          then Erreur_Salle:= true ;

else Erreur_Salle:=false ;

End_if ;

                        (*** Les Erreur du Bloc DFB Erreur_Generale***)

If Erreur_EClairage_Salle= True then set (Erreur_Eclairage) ;

end_if ;

                                (******* The END *******)
```

Erreur capteur :

```
                        (************Erreur d'eclairage**********)


                                (* SAlle 1 Une Salle de Reunion *)

if Inter_1=true then

if (Lux_1_R1< CSL/6 ) and ( Lux_1_R1 >0)
                then  set(Erreur_Lampe_Reunion[1,1]); set (Erreur_Lampe_Reunion[1,2]);
                else  reset(Erreur_Lampe_Reunion[1,1]); reset (Erreur_Lampe_Reunion[1,2]);
                end_if;

if (Lux_1_R1 <= CSL/3 ) and (Lux_1_R1>CSL/6)
                then set(Erreur_Lampe_Reunion[1,2]); set (Erreur_Lampe_Reunion[1,3]);
                else reset(Erreur_Lampe_Reunion[1,2]); reset (Erreur_Lampe_Reunion[1,3]);
                end_if ;

if (Lux_1_R1> CSL/3) and (Lux_1_R1 <= 2*CSL/3)
                then set(Erreur_Lampe_Reunion[1,2]);
                else reset(Erreur_Lampe_Reunion[1,2]);
                end_if;

if (Lux_1_R1 < CSL ) and (Lux_1_R1>2*CSL/3)
                then set(Erreur_Lampe_Reunion[1,3]);
                else set(Erreur_Lampe_Reunion[1,3]);
                end_if;
if ( Lux_1_R1=(CSL-CSL)) then set(Erreur_Lampe_Reunion[1,1]); set(Erreur_Lampe_Reunion[1,2]); set(Erreur_Lampe_Reunion[1,3]
else reset(Erreur_Lampe_Reunion[1,1]); reset(Erreur_Lampe_Reunion[1,2]); reset(Erreur_Lampe_Reunion[1,3]);

end_if;
```

```
end_if;
(*********************************************************************************************)

                                    (* SAlle 3  Un bureau*)

if Inter_3=true then


if (Lux_B_1=0 )

                then  set(Erreur_Lampe_Bureau[1]);

                else  reset(Erreur_Lampe_Bureau[1]);

                end_if;

end_if;


                                    (* SAlle 4 Un bureau *)

if Inter_4=true then


if (Lux_B_2=0 )

                then  set(Erreur_Lampe_Bureau[1]);

                else  reset(Erreur_Lampe_Bureau[1]);

                end_if;
end_if;
                                    (* SAlle 5 Un bureau *)

if Inter_5=true then


if (Lux_B_3=0 )

                then  set(Erreur_Lampe_Bureau[1]);

                else  reset(Erreur_Lampe_Bureau[1]);

                end_if;

end_if;
```

Niveau d'erreur :

```
                        (**************Debut Programme*******************)

                                (************Test des Erreur**********)

If Erreur_S1=true then S1:=1; End_if ;
If Erreur_S2=true then S2:=1; End_if ;
If Erreur_S3=true then S3:=1; End_if ;
If Erreur_S4=true then S4:=1; End_if ;
If Erreur_S5=true then S5:=1; End_if ;

                                 (************Test Niveau************)

Niveau_Erreur:=S1+S2+S3+S4+S5;

if  Niveau_Erreur>=2 then
                If Niveau_Erreur <4  then Alarme_B:=True ; end_if ;
                IF Niveau_Erreur >=4 then Alarme_A:=True ; end_if ;


Else Alarme_C:=True;
end_if ;

                                      (******Fin Programme****)
```