الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne démocratique et populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique
جامعة سعد دحلب البليدة
Université SAAD DAHLAB de BLIDA

كلية التكنولوجيا
Faculté de Technologie
قسم الإلكترونيك

Département d'Électronique

Mémoire de Master

Option Électronique
Spécialité : Signaux en Ingénierie des Systèmes et Informatique Industrielle
présenté par :

Benselama Mohamed Redha
&
Chetoui Hamza

Developpement d'un mini system SCADA

Proposé par :Mr.Bennila Nourdin

Année Universitaire 2016-2017

Remerciement

Noustenons à exprimer ici nos sincères remerciements à tous ceux qui ont contribué du près ou du loin à la réalisation de ce mémoire.

A Mr Bennila Nourdin de nous avoir encadré dans notre mémoire de fin d'étude.
Un gros merci également à nos familles pour leurs soutiens bien moral que financier et pour leurs sacrifices.
Nous tenons à remercier également Mr Benselama Zoubir responsable de mastère SISII.

Nous tenons d'autre part à remercier les respectables membres du jury pour bien vouloir nous accorder de leur temps précieux pour commenter, discuter et juger notre travail .

ملخص: ان التطور الدي شهده مجال الاتصال سمح الانتشار السريع للمعلومة , الاتصال عبارة عن تنقل المعلومة على شكل اشارات كهربائية ثم ترجمتها الى بيانات لها معنى محدد.
ان قطعات الصناعة احسنت استخلال هدا المفهوم و ادماجه في انظمتها لتسهيل عملية نقل المعلومات بداية من مستوى الحساسات الى المستوى الدي يحتوي على نضام الحاسوب الخاص بمعالجة البيانات
عموما تكون عملية المراقبة و التحكم عن بعد نظرا لبعد المنشئات عن بعضها بدلا من التحكم اليدوي
تعتبر انضمة السكادا (انضمة مراقبة التحكم و نضام اكتساب البينات) من اهم الطرق المعمول بها حديثا و التي توفر خدمتا التحكم و المراقبة على نطاق واسع.
ان اغلب عمليات التحكم الصناعي تتم عن طريق الPLC
يكمن هدفنا في تطوير نضام ال SCADAمصغرللمراقبة و التحكم في جهازين PLC من نوعS7-300 لشركة سيمنس باستخدام الادوات التالية : المبرمج TIA Portalلبرمجة ال PLC و المحيط Visual Studio لتطوير واجهة المستخدم(GUI)
كلمات المفاتيح :السكادا, التحكم, المراقبة,واجهة المستخدم

Résumé :

Dans nos jours l'information est par tout ou en vas et ça c'est grâce au développement des moyens de communications.

En industrie communiquer consiste à transmettre une donnée sous forme d'un signal(sonores,optique,lumineux,electrique)puis la traduire en une information qui a un sens, exemples : vanne ouverte, présence d'objet...Etc.

L'industrie a su exploité ce principe et l'intégré dans ces système pour faciliter le transfert de données en commençant des niveaux Capteur/Actionneur jusqu'aux niveaux les plus élevés. En général la surveillance de la production se fait à distance plutôt que par intervention manuelle sur site, Aujourd'hui les systèmes SCADA sont devenus omniprésents sur les installations a caractère industriel , c'est un outil qui permet la surveillance et le contrôle des sites entiers ou des ensemble de systèmes repartie sur des vastes zones, la plupart des action de contrôle sont effectuées automatiquement par des contrôleurs logique programmable (Plc).

Notre but consiste a développer un mini système SCADA qui gère et supervise deux API (PLc)de type S7 300 de SIEMENSen utilisant deux outils:le logiciel TIA Portal pour la programmation de l'API et l'environnement Visual Studio pour le développement d'une GUI

Mot clés : Supervision, surveillance, contrôle, SCADA,GUI. API.

Abstract:

Today the information is everywheredue to the development of the communication means

In the industry communicate is to send a data in a signal form (sonar, optic, luminous, electric) then convert it to a information which have a sense. for example: an open vane, object presence.

The manufacturing companies (industry) used this technic and integrate it to their systems to simplifies the transfer of data from Sensor/actuator level to the top levels. in general the survey of production is don remotely rather than human intervention on site.

Today SCADA systems are omnipresent on the industrial installation, scada is powerful tool which perform the surveillance and the control of an entire site, the majority of control actions are automatically performed by an PLC

Our goal is to develop a small scada system to supervise and control two s7 300 plc with two tools, the first is the TIA portal to program the plc or the process to be performed by the plcs and the second is the Visual studio environment to develop the human machine interface

Key word:supervise , control, human machine interface , SCADA, plc.

Listes des acronymes et abréviations

UI : **U**ser **I**nterface

API : **A**utomate **p**rogrammable **i**ndustriel

IHM : **I**nterface **H**omme-**M**achine

PC :**P**artie **C**ommande (**P**ersonal **C**omputer)

RS232 :**R**ecommended **S**tandard 232

USB: **U**niversal **S**erial **B**us

COM :**C**omponent **O**bject **M**odel

LPT: **L**ine **P**rinter **T**erminal

MAU : **M**ultistation **A**ccess **U**nit

CSMA/CD : **C**arrier **S**ense **M**ultiple **A**ccess / **C**ollision **D**etection

MAC :**M**edia **A**ccess **C**ontrol

IEEE :**I**nstitute of **E**lectrical and **E**lectronic **E**ngineers

DP :**D**eported Peripheral (**p**ériphérie **d**éportée)

Km :**K**ilo**M**ètre

MPI :**M**ulti **P**oint **I**nterface

OP :**O**perator **P**anel

RTU :**R**emote **T**erminal **U**nit

WIMP: **W**indows, **I**cons, **M**enus and **P**ointing **D**evice

ASCII :**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange

CPU :**C**entral **P**rocessing **U**nit

CA :**C**ourant **A**lternative

CC :**C**ourant **C**ontinue

PO : **P**artie **O**perative

RAM :**R**andom **A**ccess **M**emory

E/S :**E**ntrees /**S**orties

UC: **U**nité **C**entral

ROM: **R**ead **O**nly **M**emory

FEM:**F**orce **E**lectro**m**otrice

GRAFCET: **GRA**phe**F**onctionnel de **C**ommande par **E**tapes et **T**ransitions

ABB : **A**SEA **B**rown **B**overi.

GUI :**G**raphical **U**ser **I**nterface

TIA : **T**otally **I**ntegrated **A**utomation.

TOR : Tout Ou Rien

LED :**L**ight **E**mitting **D**iode

IDE : **I**ntegrated **D**eveloppement **E**nvironnements

VB :**V**isual **B**asic

C# :**CS**harp

IOS :**I**nternetwork **O**perating **S**ystem

DLL: **D**ynamic **L**ink **L**ibrary

TCP: **T**ransmission **C**ontrol **P**rotocol
PPI: **P**oint to**P**oint **I**nterface

Table des matières

Liste des figures

Liste des tableaux

Introduction générale

Le souci majeur des industries est la rentabilité, cette dernière ne peut être réalisé que s'ils trouvent une réponse rapide aux attentes du marché, de leurs clientèles et de leurs partenaires, tout en restant à l'écoute de ses concurrents pour ne pas se laissé distancer. Pour cela ils doivent adopter une politique qui tient compte de l'évolution spectaculaire des technologies de l'automatisation de la production industrielle.

En effet, cette évolution rapide a permis de contourner la plupart des difficultés rencontrées dans le monde industriel, et a fournie plusieurs possibilités pour satisfaire les exigences et les critères demandés tels que la production, la productivité, la sécurité, l'optimisation des coûts de production, l'amélioration des conditions de travail et la rapidité d'exécution des taches de production.

Et pour répondre à ces derniers critères les industriesont trouvé dans des éléments programmable appeler Automates Programmables Industriels (API) une réponse optimal pour des systèmes compliqués autonome. Cependant pour les systèmes industriels de plus en plus complexes et sophistiqués. Les APIs ne suffisent plus et ne répondent pas convenablement aux exigences prédéfinies, pour cela les techniciens on introduit une technique appeler la « Supervision » cette dernière sert à superviser tout le processus de production afin de répondre instantanément aux différentes défaillance qui pourraient se produire tout le long du déroulement du processus et administrer la production.

L'objectif de notre projet et de réaliser une mini plateforme de supervision « SCADA » qui peut nous aider à superviser le travail de deux automates programmables en même temps avec une seule interface graphique (User Interface(UI)) .

Sachant que l'initiation de tout projet nécessite une phase de planification. Celle-ci permet de définir les tâches à réaliser, maîtriser les risques et rendre compte de l'état d'avancement du projet. Et pour garantir le bon déroulement du projet, tout en respectant les délais, nous avons élaboré une planification globale de conduite.

Outre cette introduction générale, le présent mémoire comporte quatre chapitres et se présente comme suit :

- Le chapitre I, consiste à définir les notions de bases des technologies et protocoles de communications utilisées dans l'industrie
- Le chapitre II, définit le System d'acquisition et de contrôle de données (SCADA)
- le chapitre III, définit l'élément Automate Programmable Industriel qui est à la base de tout system SCADA .
- Et enfin le chapitre IV qui présente les outille utilisés ainsi que la réalisation de l'interface avec une méthodologie de simulation adoptée

Et on terminera par une conclusion générale et perspective.

Chapitre 1:Introduction aux réseaux industriel

1.1 Introduction :

Pour survivre aux évolutions du marché et la concurrence, les entreprises manufacturières doivent arriver à produire vite et juste,pour cela la mise en place d'une architecture qui répond aux exigences est donc nécessaire.
Pour contrôler la production, des systèmes de supervision et de contrôle ont été mise en place. ces derniers se composent de plusieurs niveaux lié par différentsprotocoles de communication selon les besoin et les contraintes , les éléments de base composant un tel système sont classés en deux grand blocs, le premier s'appuyant sur l'équipement tel que les Capteurs/ Actionneurs, API, IHM…. et les PC et le second, s'appuyant sur les techniques de gestions tel que la supervision qui se base sur les techniques de communication, ces derniers seront développé dans ce chapitre.

1.2 Notions de base :

Tout système automatisé se trouve en face de deux aspects à traiter, le premier une acquisition de donnée qui peut être une information sous forme analogique.(Figure1)

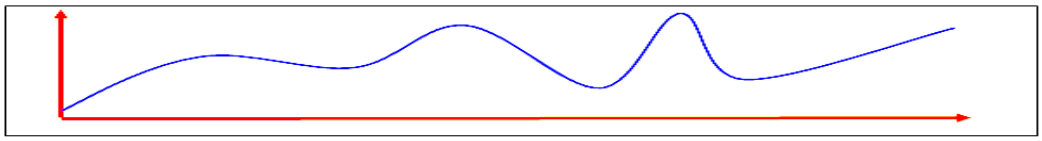


Figure 1 :signal analogique

Ou sous forme numérique .(Figure2)

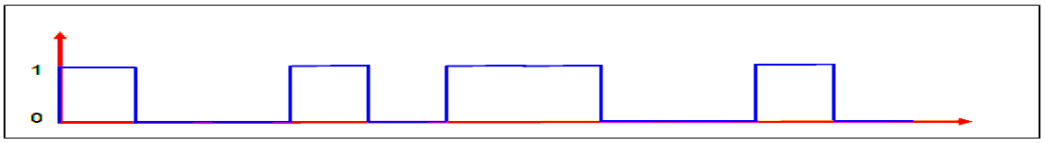


Figure 2 :signal numereique

Et le second qui s'articule autour de la transmission de ces données vers un système de traitement pour une prise de décision qui s'accommode aux consignes, dont on y trouve plusieurs type.

1.3 Les Types de transmissions :

a) Transmission simplex: cette dernière est monodirectionnel.(Figure3).

Figure 3 : transmission simplex

b) Transmission half duplex: bidirectionnel alterné .(Figure4).

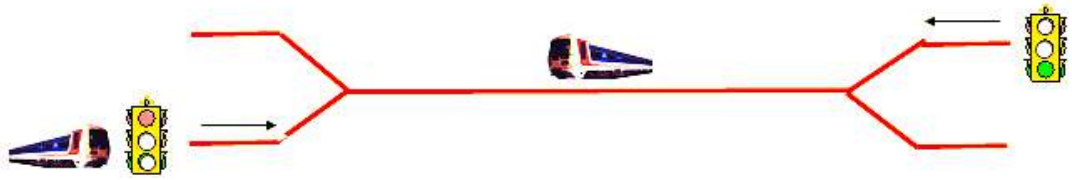


Figure 4: transmission half duplex

c) Transmission full duplex : bidirectionnel simultané. (Figure).

Figure 5 : transmission full duplex

d)Transmission série :

La liaison nécessite 3 fils : émission, réception, et masse, les bits d'un octet sont transmis les uns à la suite des autre.

e)Transmission série synchrone :

Les informations sont transmis de façon continue un signal de synchronisation est transmis en parallèle aux signaux de données.

f) Transmission série asynchrone :

Les informations peuvent être transmis d'une façon irrégulier cependant l'intervalle entre deux bits est fixe des bits de synchronisation (Start/stop) encadrent les informations de données.

i)Transmission parallèle :

Les bits d'un octet sont transmis simultanément.

La plupart des réseaux de communication industriel utilisent une transmission numérique série asynchrone half-duplex.

Tous ces types de transmissions que l'on vient de voir sont utilisées à l'aide des ports d'accès se trouvant, dans les éléments de commande tels que les API, les microcontrôleur, les ordinateurs etc.., le port série, le port parallèle représente les interface de communication les plus utilisés.

1.4 Port série :

Le port série basé sur la norme RS232 offre une voie de communication de type série disponible sur presque tous les ordinateurs. les ports RS232(Figure6) sont désignées par les noms (COM1, COM2,…etc.) encore utilisé de nos jours cependant il est de plus en plus remplacé par le port USB.

Figure 6 : port série

1.5 Port parallèle :

Dans une communication parallèle les données sont transférées de l'émetteur au récepteur par groupe de 8 bits,à travers 8 lignes. La transmission va ainsi très vite de 50 à 100 Kbytes de données par seconde (plus vite qu'en série) mais les prises sont plus encombrantes. En général Les ports parallèles ont été conçue pour communiquer avec l'imprimante, d'où le nom LPT pour (Line Printing Terminal).(Figure7).

Figure 7 : port parallèle

1.6 Les Réseaux informatique :

La principale fonction d'un réseau est de relier des objets identiques en utilisant un ensemble de règles, tous les réseaux et en particulier les réseaux informatiques assurent que les données sont partagées rapidement, de façon fiable et précise.

1.7 Topologies des réseaux :

Une topologie de réseau informatique correspond à l'architecture (physique ou logique) de celui-ci, définissant les liaisons entre les équipements du réseau et une hiérarchie éventuelle entre eux.

Elle peut définir la façon dont les équipements sont interconnectés et la représentation spatiale du réseau (topologie physique). Elle peut aussi définir la façon dont les données transitent dans les lignes de communication (topologie logique).On y trouve Topologie en bus, topologie étoile et topologie en anneau.

1.7.1 Topologie en bus :

L'avantage de cette topologie réside dans sa simplicité un câble suffit pour relier physiquement les ordinateurs(Figure8).

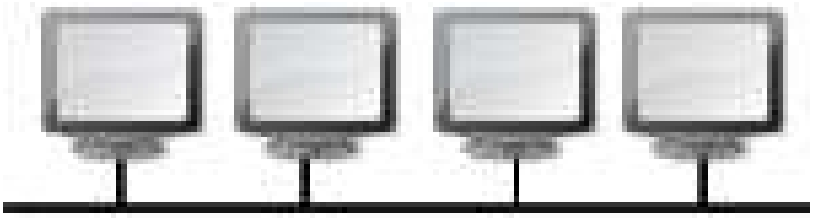


Figure 8 :schéma de topologie en bus

1.7.2 Topologies en étoile :

Les réseaux à topologie en étoile (Figure9) disposent d'une boite de jonction appelée hub ou concentrateur située au centre de réseau tous les ordinateurs se connectent au hub qui gère les communications entre ordinateurs.

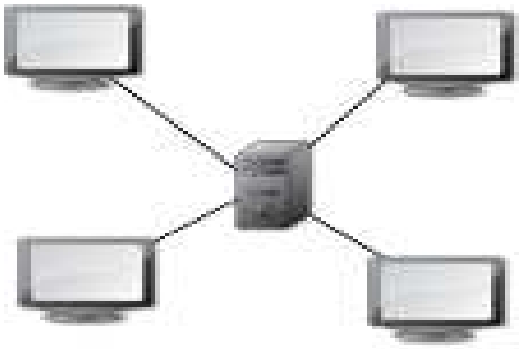


Figure 9 :schéma de topologie en étoile

1.7.3 Topologie en anneau :

Les topologies en anneau (Figure10) sont semblable a ceux d'un réseau en étoile sauf qu'au lieu de hub on trouve un équipement appelé MAU (Multistation Access Unit) qui gère différemment les communications entre ordinateurs

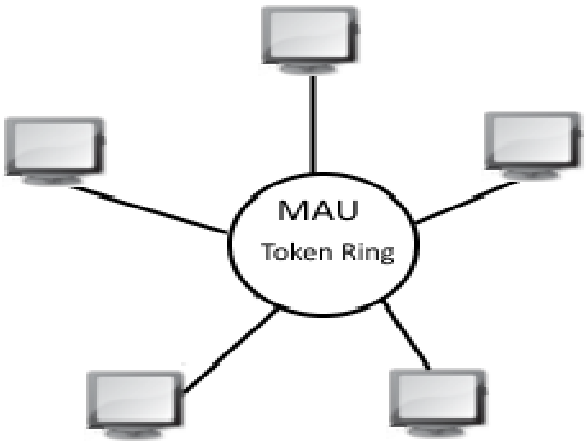


Figure 10 :schéma de topologie en anneau

1.8 Architecture d'un système automatisé :

Les besoins et les contraintes de communication imposent des architectures qui se structurent en quatre niveaux distincts et interconnectés par des réseaux (Figure11).

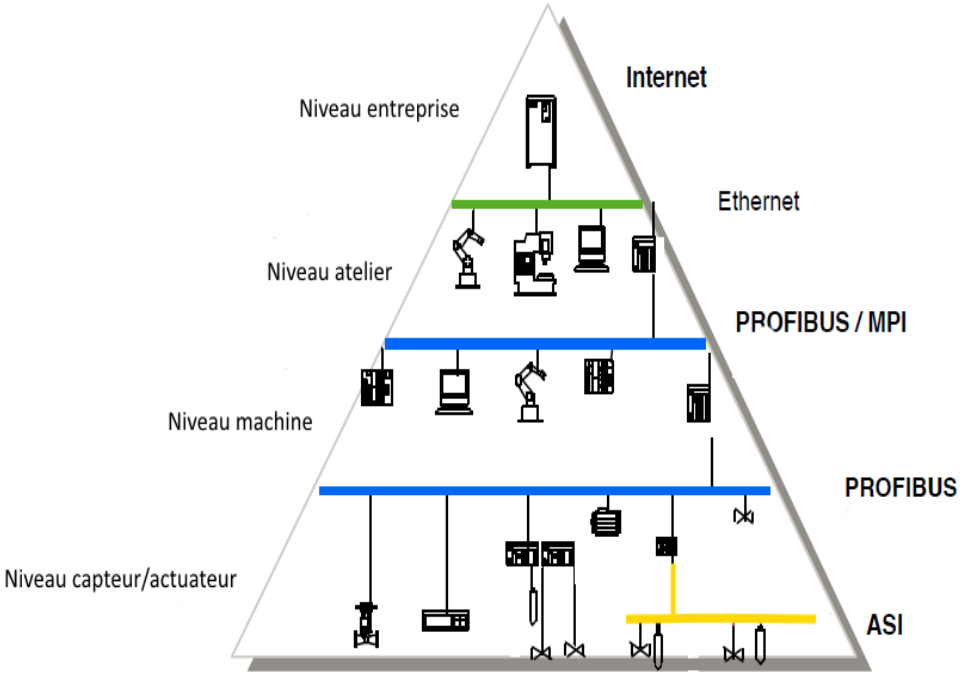


Figure 11 :architecture d'un system automatisé

1.9 Les besoins et contraintes de communication :

Tableau 1 : besoins et contraintes de communication

Niveau	Besoin	Volume information à transmettre	Temps de réponse	Distance	Topologie réseau	Nombre d'adresses	Médium
Entreprise	Echange de données sécurisé et compatibilité entre progiciels	Mbits	1 min	Monde	Bus, étoile	Non limité	Electrique, optique radio
Atelier	Performance temps réel et synchronisation entre API	Kbits	50 à 500 ms	2 à 40 Km	Bus, étoile	10 à 100	Electrique, optique, radio
Machine	Topologie et cout de connexion	Kbits	50 à 500 ms	10 m a 1K m	Bus, étoile	10 à 100	Electrique, optique, radio
Capteur	Simplification de câblage Distribution des alimentations	bits		1 à 100 m	Sans constraints	10 à 50	Electrique, radio

Le tableau si dessue montre les principaux besoin est contrainte qui peuvent être dans une hiérarchie de couches d'un domaine d'automatisation.
Comme les déférentes taches de communication ne peuvent pas être solutionnées avec un seul réseau plusieurs systèmes de communication ont été développés (Figure12).

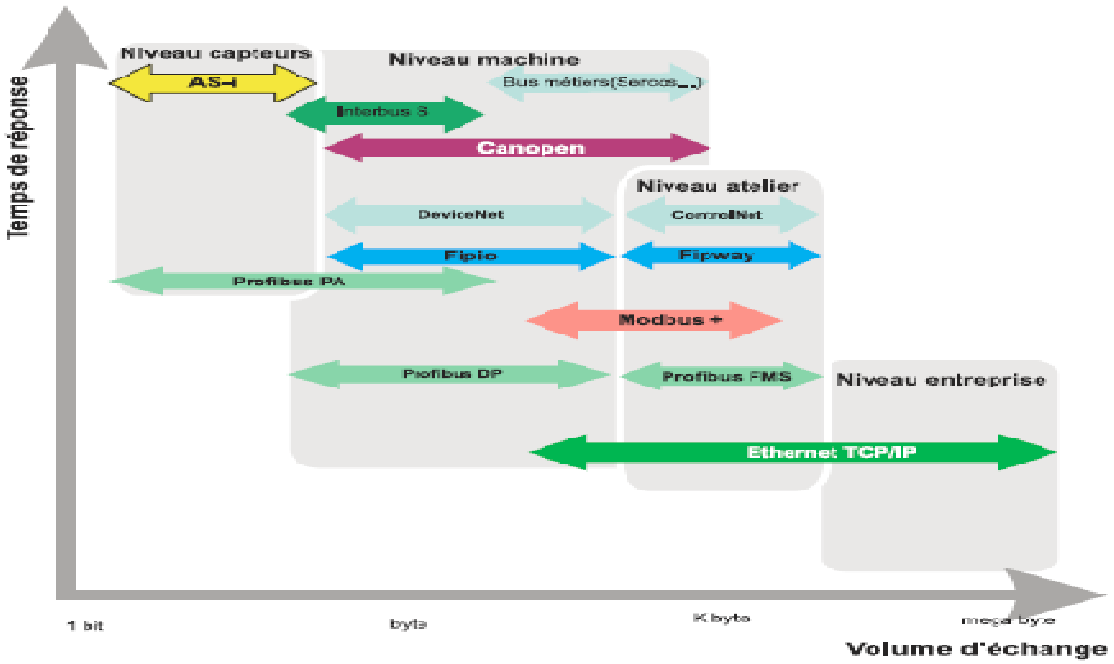


Figure 12 : les system de communications

Dans le niveau le plus haut on trouve des appareils complexes comme les ordinateurs qui traites une grande quantité d'informations à grande vitesse et qui gère et enregistre ces dernières.
Dans ce niveau le temps de réponse (traitement des processus et données) n'est pas critique **Ethernet** est donc le moyen de communication le mieux placé pour communiquer dans ce niveau, Celle-ci est Basée sur un mécanisme de détection de collision, Cette technologie connue sous le nom de Carrier Sense Multiple Access with Collision Détection Ou CSMA/CD garantie qu'une seule station à la fois transmet un message sur le medium. Chaque station dans le même réseau est caractérisé par une adresse appelée adresse MAC (unique pour chaque station)
L'évolution d'Ethernet ont données naissance au standard IEEE 802.3 qui définit les caractéristiques des couches physique et la façon dont l'information accèdent au réseau.
Les besoin de temps réel ont limité l'utilisation de ce standard aux niveaux d'un système automatisé.
La communication dans les bas niveaux est caractérisée par de petites quantités de données. Les exigences de temps réel sont ici a considéré en premier
Plusieurs types de réseaux ont été mise en place par les constructeurs d'API selon les besoin de communication
Pour communiquer avec des API, Siemens (l'objet de notre projet) a développé et adopté plusieurs systèmes de bus de transmission pour répondre aux exigences et aux besoins de communication dans chaque niveau :

1.9.1 Le PROFIBUS :

PROFIBUS (Processe Field Bus)(Figure13).est un standard de bus non propriétaire, Siemens a adopté cette norme en raison de ses vaste fonctionnalités son utilisation s'étends du niveau atelier jusqu'au niveau capteur permettant une connexion décentralisé des appareils essentiellement PROFIBUS-DP (Périphérie décentralisé).
Les constituants d'automatisme tels qu'automates programmable, PC, IHM, capteur et actionneurs peuvent communiquer sur un réseau unifié

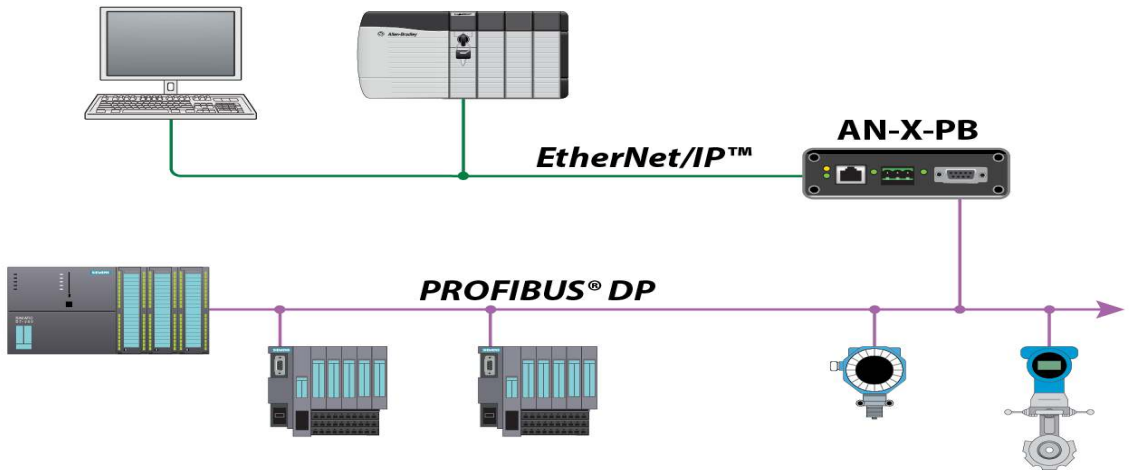


Figure 13 : communication utilisant le profibus

Les Donnée technique de PROFIBUS-DP :

- La répartition de bus s'effectue selon le procédé Passage de jeton a maitre-esclave.
- 127 participants max avec une taille de trame de 0-246 Octets
- Les vitesses standard de transmission définies sont : 9.6 Baud/19.2 Baud/93.75 KBaud/187.5Kbaud/500Kbaud/1.5 MBaud/3 MBaud /6 MBaud /12 MBaud
- la transmission des données s'effectue soit par un câble a 2 fils avec l'interface RS-485 ou par fibre optique
- extension de transmission pour une conception électrique jusqu'à 12Km, pour une conception optique jusqu'à 23.8 Km

1.9.2 L'interface AS :

L'interface Actuateur/Capteur (AS-I) sert à la transmission dans les niveaux les plus bas et est un standard ouvert. Siemens offre des produits et des interfaces AS-I.

AS-I permet une intégration simple des capteurs et d'actuateurs dans les communications industrielles .(Figure 14)

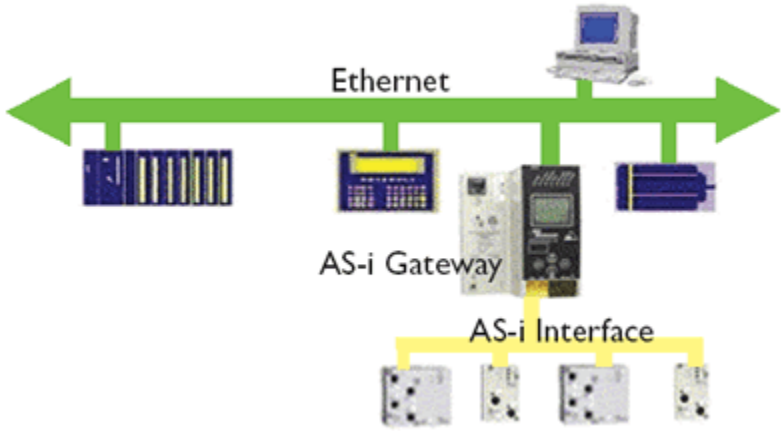


Figure 14 : interface AS

Les Données technique de l'interface AS :

- 31 participants AS-I max.
- possibilité de raccordement en étoile, en ligne ou en arbre.
- longueur max de câble 100-300m (avec répéteur)
- durée de cycle max 5ms
- le media de transmission est un câble blinde a deux fils (2 * 1.5mm²)

1.9.3 L'Interface Multipoint (MPI) :

Le MPI es un bus de terrain développé par siemens on le trouve comme interface intégré dans les appareils siemens , il sert pour la communication avec les composants mis en place pour servir et visualiser ainsi que pour la communication entre deux ou plusieurs automates

1.9.3.1 Donnes technique de la MPI

- 32 participant max
- topologie de réseau : RS485 a bus .étoile ou anneau
- media de transfert : câble blindé a deux fils ou fibre optique
- vitesse de transmission 19.2Kbit/s -12Mbit/s
- Extension de réseau jusqu'a 50m et 1100m avec des répéteurs RS485

1.9.3.2 Configuration d'un réseau MPI :

La configuration d'un reseau MPI (Figure15) a l'aspect suivant:

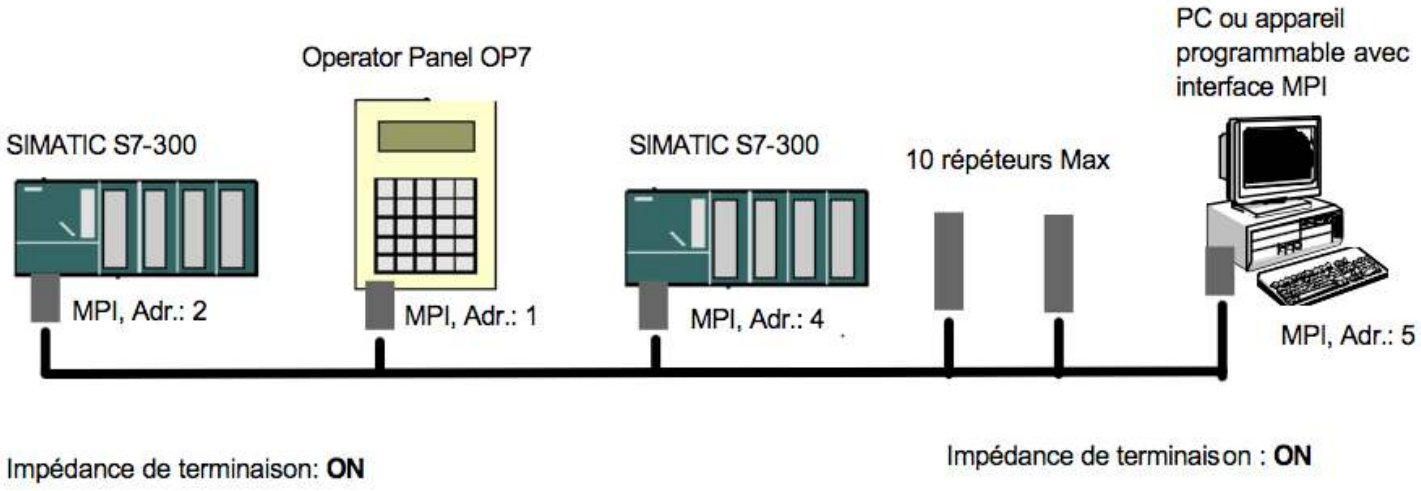


Figure 15 : réseau utilisant MPI

Un maximum de 32 participants peuvent être connecté entre eux, chaque participants doit avoir une adresse défèrent a l'intérieur d'un même sous réseau, le taux de baud doit être de la même grandeur chez tous les participants.

1.10Conclusion :

On s'est intéressé dans ce chapitre aux différentes normes de protocole de communications. Plusieurs protocoles de communicationssont apparu suite à la concurrence entre les entreprises manufacturières qui utilisent les mêmes standards de communication (Ethernet ,RS232…etc.)
Dans ce 1erchapitre, Une définition des normes des protocoles de communication utilisée dans le domaine industriel ont été nécessaire pour bien comprendre les chapitres qui suivent dans ce projet, ou on va parler de la supervision avec le system SCADA qui nécessite quelque notion de base sur les protocoles de communications.

Chapitre 2 : System d'acquisition et de contrôle de données(SCADA)

2.1 INTRODUCTION :

Comme on a vu au chapitre précédent que Pour une gestion optimal d'un système automatisé complexe, un outil de supervision global est nécessaire, ce dernier pourra augmenter la production et la productivité en contrôlant à distance toutes les étapes de production. Ce système de Supervision est appelle par les automaticiens le système SCADA ce dernier se base surtout sur les systèmes de communication

2.2 Définition du système SCADA :

Un système d'acquisition et de contrôle de données (SCADA) (Supervisory Control And Data Acquisition) est la partie qui relie le plus haut niveau (Niveau entreprise) avec le plus bas niveau (Niveau Capteurs/Actuateurs) dans la hiérarchie d'une entreprise manufacturiers(Figure 16). Elle traduit les signaux de base en une information visible sur l'écran (du capteur vers L'API et en fin à l'IHM/OP (Operator Panel))

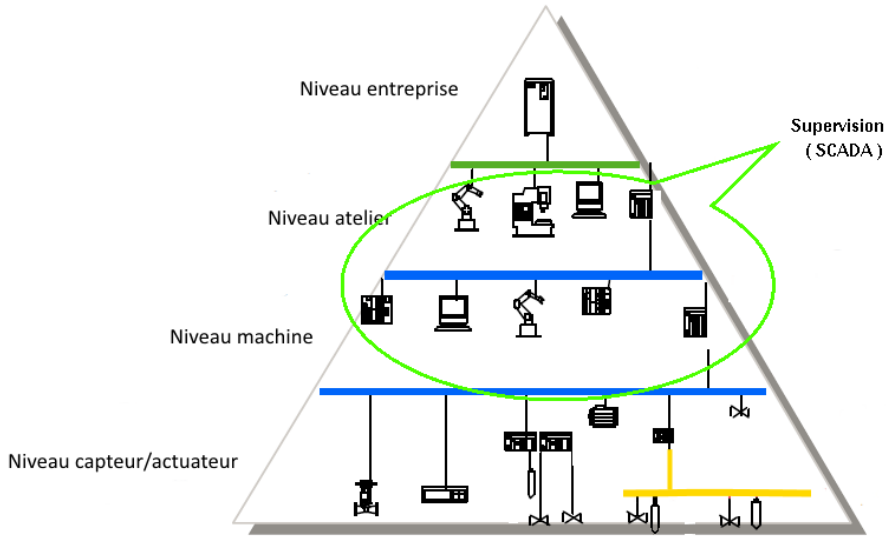


Figure 16 ;schema montrant l'emplacement du SCADA

l'API contrôle plusieurs type de processus par les module d'entrée / sortie analogique et numérique, les informations qui circulent sont collecté par les Terminaux distants (anglais : Remote Terminal Units « RTU ») , les API et d'autres appareils intelligents (anglais : Intelligent

Electronic Devices « IED ») puis transférée à l'unité central (IHM) pour le contrôle et l'analyse des données.
Les informations sont aussi affichées sur l'écran de l'opérateur (OP) pour un contrôle manuel décentralisé.

2.3 Composants du SCADA :

Les systèmes SCADA sont basés sur deux éléments essentiels, les API qui peuvent être programme pour contrôler et répéter une tache précise et les IHM qui permettent à l'homme de prendre le contrôle et connaitre l'état actuel du système (API, capteur et actuateur).

Un dispositif SCADA est généralement composé des sous-systèmes suivants :

- un système de supervision et de contrôle informatique, faisant l'acquisition des données de L'API et envoyant des commandes (consignes) à l'API,
- d'une IHM permettant à l'opérateur de superviser la machine à partir d'un tableau de bord virtuel comportant des boutons, des voyants, des

alertes.

- d'une base de données pour stocker toutes les données dont il a besoin pour la prise de décision.
- des APIs utilisés sur le terrain pour leur versatilité et flexibilité due à leur capacité d'être configurables.
- une unité terminale distante (RTU) reliant les capteurs convertissant les signaux en flux de données numériques et envoyant les données numériques à l'API.
- une infrastructure de communication reliant le système de supervision et contrôle aux éléments terminaux.
- divers instruments d'analyse.

L'ensemble des couples API/IHM forme ce qu'on appelle le SCADA. Bien entendu, le SCADA peut comporter plusieurs API qui sont extensibles en plusieurs modules d'E/S. Il présente une souplesse et une adaptabilité dans son installation puisque les fonctions logiques sont toutes rassemblées en un seul programme qui peut être aisément modifié .

2.4 Les Interfaces Homme-Machine dans le SCADA

L'un des éléments les plus importants d'un système de supervision SCADA est l'Interface Homme-Machine. Elle représente le seul point d'interaction entre l'opérateur et le système supervisé.

a) Définition d'une Interface Homme-Machine :

L'interaction Homme-Machine(Figure18) représente l'ensemble des mécanismes d'échange d'information entre un humain et une machine (système interactif) (Figure17) pour accomplir une tâche ou atteindre un but particulier pour l'humain. Elle est caractérisée par le triplet opérateur humain (ou utilisateur), machine et environnement et l'interaction (interface).

Figure 17 : schéma de l'IHM(interface homme machine

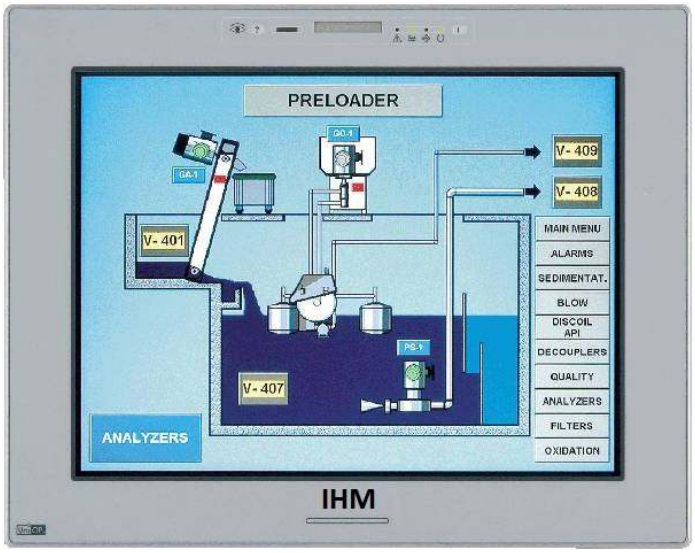


Figure 18 : Interaction Homme Machine

Une IHM sert à afficher de manière claire et concise une vue d'ensemble du système et l'état des services surveillés, de générer des rapports et de visualiser l'historique . Depuis quelques années, les IHM s'imposent comme un facteur déterminant dans le succès des applications de supervision. Cet intérêt accordé aux IHM est à l'origine de différents travaux qui se sont intéressés à la définition, la caractérisation, la conception et l'implémentation des IHM. Elles sont un outil très important pour le bon déroulement du système industriel de même pour la réception d'informations de son environnement SCADA .

b) <u>Evolution des interfaces Homme machine :</u>

Les premières interfaces apparues, dès la fin des années 60, sont les interfaces en ligne de commandes basées sur une interaction textuelle entre l'utilisateur et la machine. Elles étaient révolues lors de l'avènement des interfaces WIMP en début des années 80 (Windows, Icones, Menus and Pointing device) pour exprimer qu'elle utilise des métaphores de type : fenêtres, icônes, menus et dispositif de pointage. Les interfaces WIMP sont basées sur le principe WYSIWYG (What You See Is What You Get) pour ce
que vous voyez est ce que vous obtenez. Ces interfaces offrent une interaction plus intuitive en réduisant les efforts d'apprentissage, elles ont grandement contribué à l'utilisation des ordinateurs par le grand public.

2.5 Fonctionnalités d'un système SCADA

Un système SCADA comprend deux sous-ensembles fonctionnels:

a) Un sous-système commande : son rôle est de faire exécuter un ensemble d'opérations au procédé en fixant des consignes de fonctionnement en réponse à des ordres d'exécution. La commande regroupe toutes les fonctions qui agissent directement sur le processus du procédé:

- le fonctionnement en l'absence de défaillance.
- la reprise ou gestion des modes.
- les traitements d'urgence.
- une partie de la maintenance corrective.

b) Un sous-système surveillance : la partie surveillance d'un superviseur a pour objectifs :

- La détection d'un fonctionnement ne correspondant plus à ce qui est attendu.
- la recherche des causes et conséquences d'un fonctionnement non prévu.
- La collaboration avec les opérateurs humains pour les prises de décision critiques, pour le recueil d'informations non accessibles directement.
- reconstitue l'état réel du système commandé.
- fait toutes les inférences nécessaires pour produire les données utilisées pour dresser des historiques de fonctionnement.

2.6 Communication des systèmes SCADA

La notion temps réel est devenue très importantes et indispensable dans la procédure de surveillance et de supervision en générale, elle permet de suivre l'évolution de l'état du système d'une façon continue .
Dans un système de supervision SCADA afin d'accéder en temps réel aux informations,
en lecture et écriture, qui se trouvent au niveau des API nous avons recours a :

Des moyens de communication physiques :

- liaison série standard RS 232 ou RS485.
- liaison dédiée ASI, CAN, Profibus.
- réseau Ethernet.

Des protocoles différents :

- Modbus ASCII, RTU, TelWay.
- CanOpen, DeviceNet.
- UDP, TCP/IP, ModbusTCP.

2.7 Conclusion :

Dans ce chapitre On vient de détailler le système SCADA, ce concept étant nécessaire pour tout système automatisé complexe constitué d'un ensemble d'éléments programmable tels que les microcontrôleurs, les microprocesseurs ou les APIs. Ces derniers seront détaillés dans le chapitre qui suit en ciblant le type choisi pour notre projet qui n'est autre que l'API SIEMENS S7300 CPU 312

Chapitre 3 : Les Automates Programmables Industriel(API)

3.1 Introduction :

Tout système automatise est constitué d'une partie commande et d'une partie opérative, la première est souvent réalisé à base de circuit programmable dans lequel on y installe les consignes nécessaires pour l'exécution des taches correspondantes a l'application voulu, elles sont souvent constitué d'éléments programmables, dans l'industrie ces éléments se matérialise a travers les APIs (Automates Programmables Industriels). pour notre cas nous détaillerons API SIEMEN S7 300 CPU 312. qui sera utilisé dans notre projet

3.2 Historique des automatesprogrammables :

Les automatismes séquentiels ont été réalisés, depuis longtemps, à base de relais électromagnétiques. L'inconvénient c'est qu'il s'agit d'une système câblée ce qui mpose la refonte complète du câblage et ceci pour la moindre modification dans l'ordonnancement des séquences. En 1966, l'apparition des relais statiques a permis de réaliser des divers modules supplémentaires tel que le comptage, la temporisation, le pas à pas ... Cependant cette technologie avait le même problème : technologie câblée.
En 1968 et à la demande de l'industrie automobile nord-américaine, sont apparus les premiers dispositifs de commande logique aisément modifiable : Les PLC (Programmable Logic Controller) par *Allen Bradley, Modicom et Digital Equipement*. Le premier dispositif français était le PB6 de Merlin Gerin en 1973

3.3 API–Définition API :

L'automate programmable industriel (API) est un contrôleur à base de microprocesseur qui commande un processus (par exemple une presse pour le moulage de formes en plastique sous pression) ceci est possible grâce aux instructions d'un programme stocké dans la mémoire de l'API
Les entrée de l'automate sont des signaux provenant du processus par l'intermédiaire de capteurs (fin de course, détecteur de présence etc..), les sortie de l'automate agissent sur le processus par l'intermédiaire d'actionneurs (contacteur, commande de vanne, etc...)
Le rôle de l'automate est de réagir aux changements d'état de ses entrées en modifiant l'état de ses sorties.
Les domaines d'utilisation sont très divers : mécanique et automobile , chimique, pétrolière, alimentaires,….

3.4 Structure d'un API :

3.4.1 Architecture interne d'unAPI :

la structure de l'automate programmable ressemble à celle d'un micro-ordinateur, constitué d'un bloc d'alimentation, une unité centrale (unité de traitement), des modules d'entrées (interface d'E), des modules de sortie (interfaces de S), d'une console de programmation,...(Figure19).

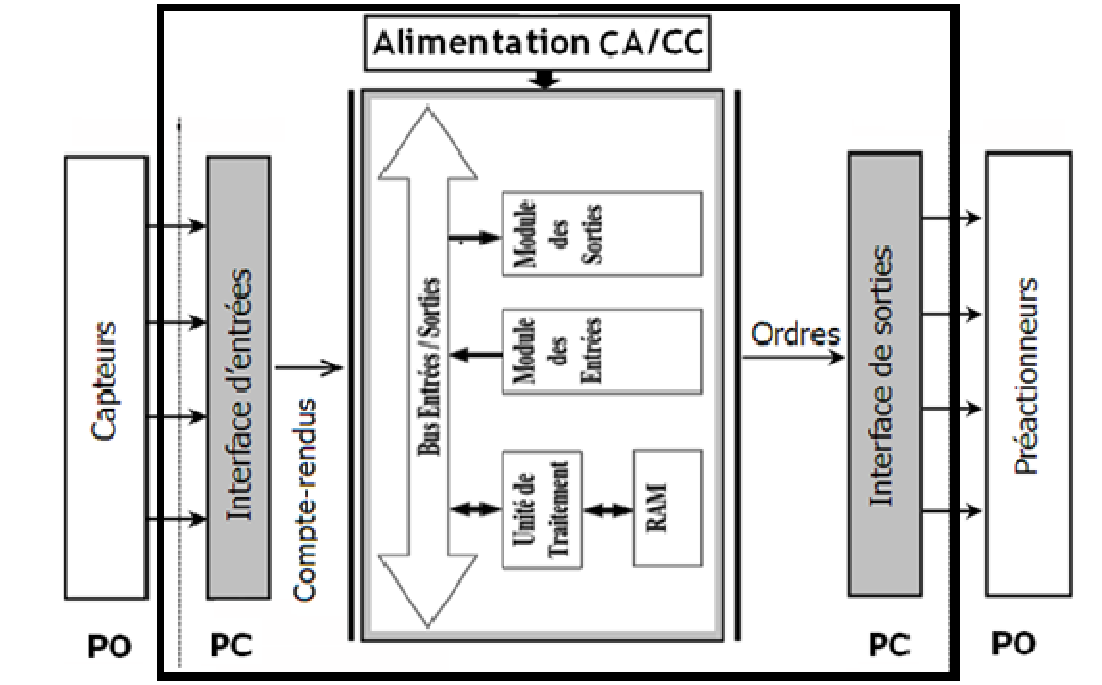


Figure 19 :L'architecture d'un API

Remarque : En général, les automates sont conçus pour être modulaires, notamment pour pouvoir augmenter le nombre d'E/S. D'ou l'utilisation d'une structure d'un rack dans lequel s'encastrent les différentes cartes (UC, alim, E/S, ...).

a) L'unité centrale(UC) :

C'est le cœur de la machine, comporte le(s) processeur(s) et la mémoire(s).
Processeur :appelé unité de traitement, il assure le contrôle de l'ensemble de la machine et effectue les traitement demandés par les instructions du programme. Il réalise les fonctions logiques, temporisation, comptage, calcul. Il comporte un certain nombre de registres (compteur ordinal, registre d'instructions, registre d'adresse, registres de données,

accumulateurs, ... Il est connecté aux autres éléments (mémoires, interfaces d'E/S, ...) par l'intermédiaire des bus.

Mémoire : La mémoire centrale est découpée en plusieurs zones :

- zone mémoire programme ;
- zone mémoire des données (états des E/S, valeurs des compteurs, temporisations)
- zone où sont stockés des résultats de calcul utilisées ultérieurement dans le programme
- zone pour les variables internes.

Il existe différents types de mémoires :

Mémoires vives : RAM ce sont des mémoires volatiles : elles perdent l'information en cas de coupure de l'alimentation. Certaines d'elles sont équipées de batteries de sauvegarde (autonomie réduite). Elles sont accessible en lecture et en écriture.

Mémoires mortes : les contenus sont figés. Ce sont des mémoires à lecture seule. Les informations sont conservées en permanence sans source externe.

ROM : mémoire programmé par le fabriquant et ineffaçables.

PROM : vendues vierges et programmables une seule fois par l'utilisateur .

REPROM ou EPROM : utilisables plusieurs fois (écriture / effacement). Effacement a UV. Elles ne peuvent être reprogrammées qu'après un effacement total .

EEPROM : effacement électrique et reprogrammation rapide sur place

b) Blocd'alimentation :

Permetdefourniràl'automatel'énergienécessaireàsonfonctionnement.Ilsdélivrent,à partir du 220V alternatif, des sources de tension nécessaires à l'automate tels que : +5V, 12Vet 24V encontinu avec un courant max de 10 A.(Figure20).

Figure 20 :Bloc d'alimentation

c) Les cartesd'E/S :

Les E/S des automates programmables revêtent une importance évidente au plan technique. Leur coût dépasse fréquemment la moitié de l'investissement total d'une configuration. Ces

facteurs justifient une étude détaillée de leur architecture générale, suivi de celle des E/S industrielles typiques.

d) console de programmation :

Pour programmer un API on utilise un microordinateur munis d'un logiciel pourcommuniquer/ implémenter le programme dans l'API qui lui permet de faire son travail

3.5 Mise en œuvre et programmation d'un API :

Les automates doivent fonctionner sans danger pour les utilisateurs et sans risque d'interrompre la production et ceci malgré les contraintes très sévères qu'ils subissent en milieu industriel. Les défaillance peuvent provenir de la conception, de l'installation, de l'exploitation d'un constituant ou d'un composant, de l'environnement, de la maintenance,... Il faut donc veiller à la sécurité en analysant les risques et les normes en vigueur.
Les automates sont soumis à différents types d'environnement qu'il faut les prendre en considération :

- Environnement physique et mécanique : vibrations, chocs, humidité, température.
- Environnement chimique : gaz corrosifs (CI2, H2S,SO2), vapeurs d'hydrocarbures, poussières métalliques (fonderies, aciéries, ...), poussières minérales (cimenteries,
- Environnement électrique : les éléments perturbateurs sont:
  - les forces électromotrices (fem).
  - les parasites d'origine électrostatiques.
  - lesinterférencesélectromagnétiques(transformateurs,postesdesoudure).
  - Certains émetteurs récepteurs à des fréquences correspondant àl'API peuvent détériorer le processeur de celui-ci.

3.5.1 Mise en œuvre:

La réalisation de tout ou partie d'une partie commande (PC) en logique programmée nécessite la traduction du modèle concerné (GRAFCET, schémas, équations, ...) en programme exécutable par la machine. L'élaboration d'un tel programme vise donc à écrire les équations d'activation de sorties de l'API et les conditions associées. Elle constitue la phase logicielle del'application.

Pour fonctionner, l'automate a besoin qu'on lui charge un programme qui sera exécuté infiniment quand l'api est sur l'état marche, le programme chargé est sauvegardée dans une mémoire morte (EPROM) la mise hors tension de l'API n'influe pas sur le programme car ce type de mémoire peu stocké des informations sans aucune alimentation électrique.

a) Adressage des E/S:

C'est l'association aux différentes affectations les repérages adéquats : par exemple pour le s7 300 CPU 312 qui est representé sur la figure22,de I124.0 à I125.1 pour les entrées et Q 124.0 à 124.5 pour les sorties.
La figure 21 représente la façon dont les adresses sont affecter depuis le logiciel de programmation

PLC tags

	Name	Tag table	Data type	Address	Retain	Visibl...	Acces...
1	Start	Default tag table	Bool	%I124.0		☑	☑
2	Stop	Default tag table	Bool	%I124.1		☑	☑
3	Engin	Default tag table	Bool	%Q124.0		☑	☑
4	<Add new>					☑	☑

Figure 21 :L'adressage des entres sorties

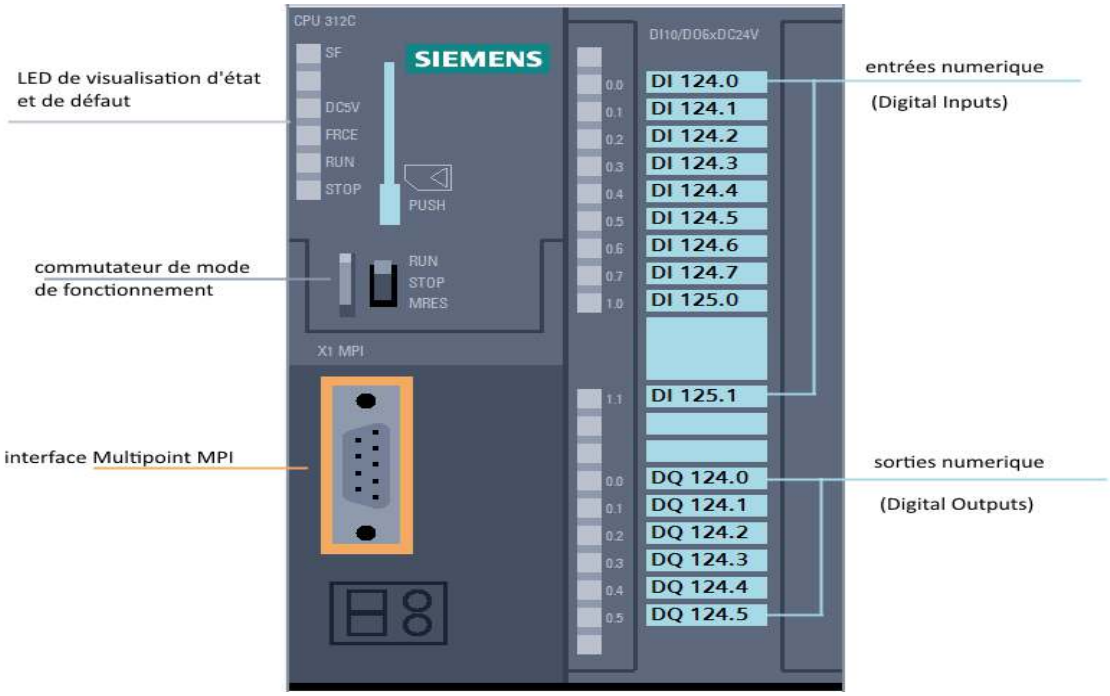


Figure 22 : les entres sortie physique

On utilise les variables physiques pour jouer sur l'état d'E /S de l'automate (figure23)

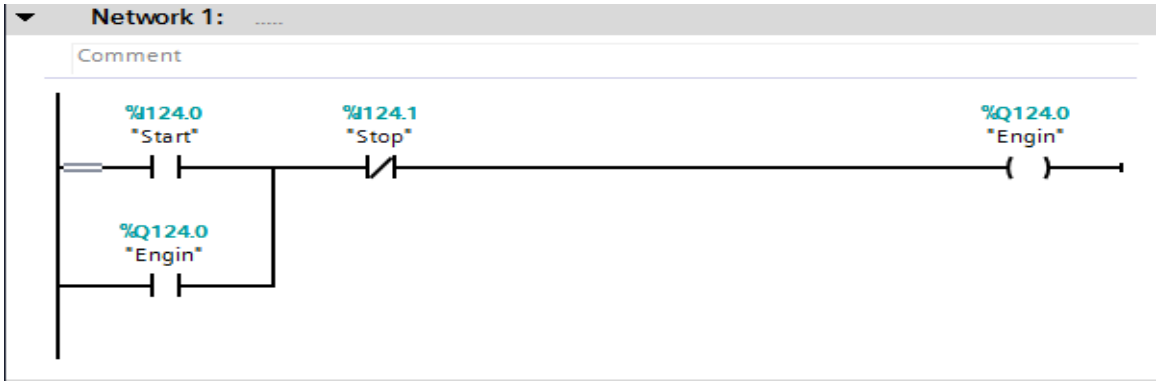


Figure 23 : les variables physiques

b) Variables internes:

L'affectation consiste également à identifier ces variables destinées à mémoriser les états et valeurs intermédiaires durant l'exécution du programme (qui sont des variables intermédiaires non phtisique) (figure 24)

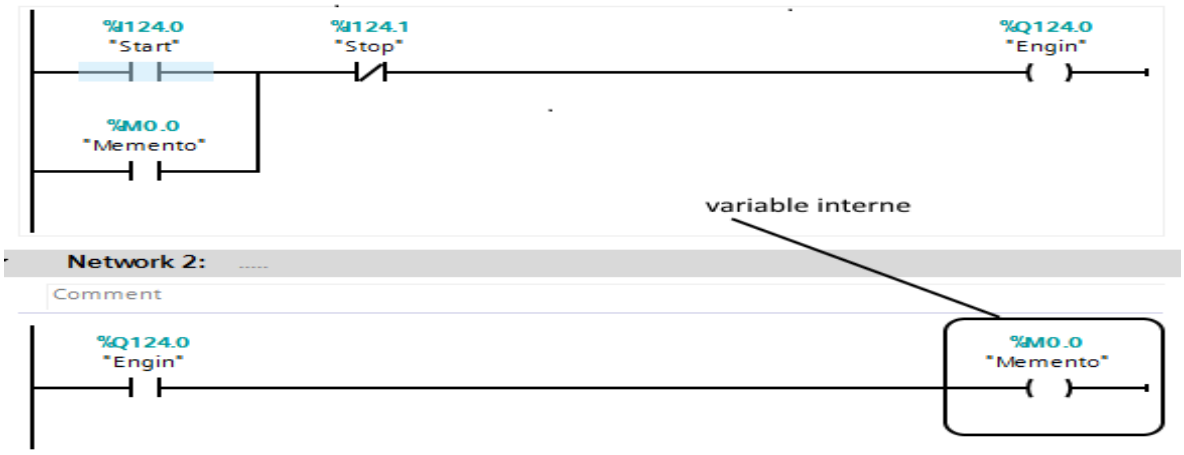


Figure 24 : variables interne

3.5.2 Programmation :

Il existe plusieurs langages pour programmer un API selon les besoin et le choix du programmeur

a) **Langages littéraux :** Langage liste d'instruction : (Instruction List) est très proche du langage assembleur qui est un langage textuel de bas niveau. (figure 25)

```
cal_sum:
      push   ebp
      mov    ebp, esp
      sub    esp, 4           ; fait de la place pour le sum local

      mov    dword [ebp - 4], 0   ; sum = 0
      mov    ebx, 1               ; ebx (i) = 1
for_loop:
      cmp    ebx, [ebp+12]        ; i <= n ?
      jnle   end_for

      add    [ebp-4], ebx         ; sum += i
      inc    ebx
      jmp    short for_loop

end_for:
      mov    ebx, [ebp+8]         ; ebx = sump
      mov    eax, [ebp-4]         ; eax = sum
      mov    [ebx], eax           ; *sump = sum;

      mov    esp, ebp
      pop    ebp
      ret
```

Figure 25 : Langage liste d'instruction

b) **Langage littérale structure :** (Structured Text) est très proche du langage « C » qui est un langage de haut niveau. (figure26)

```
            (* TEST CYCLE SETUP  *)
            cycle_In1 := tb_In1[testCycleNum];
            cycle_In2 := tb_In2[testCycleNum];
            cycle_In3 := tb_In3[testCycleNum];
            cycle_Out := tb_Out[testCycleNum];
            IF testCycleNum = 0 THEN
                (* INIT  *)
                PID_Subsystem(i0_PID_Subsystem, 0, cycle_In1, cycle_In2, cycle_In3, out_Out);
            END_IF;
            (* STEP  *)
            PID_Subsystem(i0_PID_Subsystem, 1, cycle_In1, cycle_In2, cycle_In3, out_Out);
            (* VERIFY  *)
            IF testVerify THEN
                IF cycle_Out = 0.0 THEN
                    IF ABS(out_Out) > 9.9999997473787516E-5 THEN
                        testVerify := 0;
                    END_IF;
                ELSIF ABS(out_Out - cycle_Out) > (9.9999997473787516E-5 * ABS(cycle_Out)) THEN
                    testVerify := 0;
                END_IF;
            END_IF;
            testCycleNum := testCycleNum + 1;
        END_IF;
    END_IF;
```

Figure 26 : Langage littérale structure

c) Langages graphiques :

i.Langage a contacts ou diagramme en échelle : (Ladder Diagram) c'est le langage le plus utiliser il ressemble aux schémas électriques (figure27).

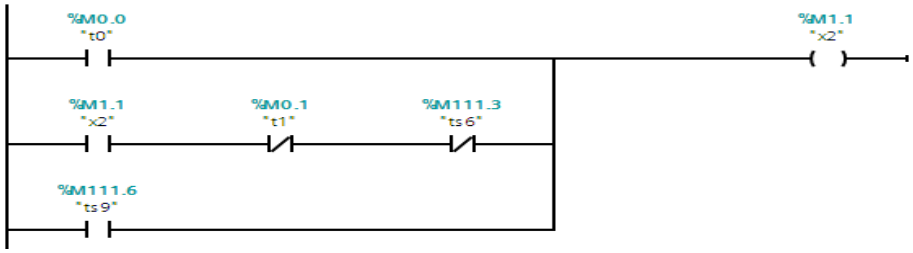


Figure 27 : langage graphique

ii.Le GRAFCET : (GRAphe Fonctionnel de Commande par Etapes et Transitions) c'est un outil graphique qui décrit l'évolution d'un automatisme par des étapes et des transitions.(figure28).

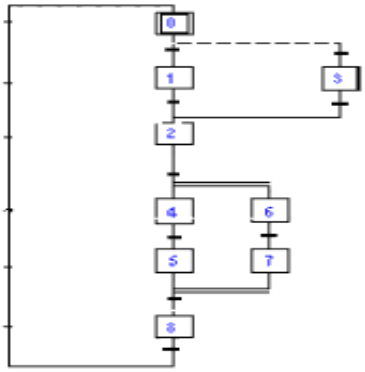


Figure 28 : GRAFCET

iii.Blocs Fonctionnels :(FBD) ce langage est aussi graphique qui utilise des blocs logiques chaque bloc représente une fonctionne qui peut être reprogrammé (figure29)

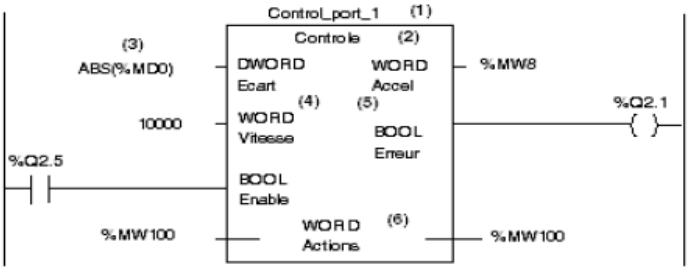


Figure 29 : bloc fonctionnel

3.6 Les constructeurs des APIs :

Parmi les plus grosconstructeurs on trouve : SIEMENS. OMRON et SCHNEIDER.

Siemens :

Siemens est un groupe international d'origine allemande spécialisé dans les hautes technologies et présent dans les secteurs de l'industrie, de l'énergie et de la santé. Il a été fondé en 1847 par Werner von Siemens. Le groupe, dont le siège est à Munich, est le premier employeur privé d'Allemagne, et la plus grande société d'ingénierie (en termes d'effectifs) en Europe.

Siemens occupe plusieurs domaines dont :

- Industrie
  - Automatisation
  - Entraînements
  - Services
  - Traitement de l'eau
  - Mécanique de grande dimension

    - Logiciels (Software)
- Énergie
    - Énergies fossiles
    - Énergies renouvelables dont Siemens Wind Power
    - Oil & Gas
    - Transmission d'énergie
    - Services
- Santé
    - Imagerie médicale, radiothérapie (scanner , échographie, radiologie, radiologie digitale, médecine nucléaire, distribution d'images, coronarographies, radiologie interventionnelle)
    - Diagnostics de laboratoire (Automates d'analyses médicales et réactifs pour analyses de Biologie médicale)
    - Prothèses auditives (Audioprothèse)
- Infrastructure & Cities
    - Technologies du bâtiment (Efficacité énergétique, Chauffage, Ventilation, Climatisation, Détection incendie, Contrôle d'accès, Gestion horaire)
    - Systèmes de transport
    - Distribution d'énergie

Omron:

Omron (*Omuron Kabushiki-gaisha*) est une entreprise japonaise d'électronique basée à Kyoto.

Omron a été créé par Kazuma Tateish en 1933.

Son secteur d'activité était initialement la vente et fabrication de systèmes d'automatisme.

Elle est connue en particulier pour les équipements médicaux comme les thermomètres et les tensiomètres. Omron a développé et mis au point le premier portillon d'accès électronique, ainsi que les premiers distributeurs automatiques de billets de banque avec carte magnétique.

Schneider :

Schneider Electric SE est un groupe industriel français à dimension internationale, qui fabrique et propose des produits de gestion d'électricité, des automatismes et des solutions adaptées à ces métiers.

Métiers :

- La distribution électrique

- La distribution électrique

Elle consiste à rendre l'énergie électrique disponible et fiable. Ainsi, Schneider Electric ne produit pas de l'électricité, mais utilise son savoir-faire pour l'acheminer, la transformer et la sécuriser.

- Positionnement

Schneider Electric est le numéro 1 mondial de la distribution électrique sur l'ensemble de son offre. Premier en gestion d'énergie, bâtiment distribution, optimisation de l'énergie, sécurité électrique.

- Exemples d'offres

Schneider Electric propose des produits de basse ou haute tensions. À titre d'exemples en basse tension :

les disjoncteurs, interrupteurs, éclairages de sécurité, canalisations électriques préfabriquées et prises électriques. En moyenne tension, les appareillages et équipements sont utilisés pour transformer l'énergie haute tension, et l'acheminer vers les utilisateurs finaux. À titre d'exemple :

un transformateur haute / moyenne tension.

  - Les automatismes

- Positionnement

Schneider Electric fait partie des leaders mondiaux en automatismes et contrôle. Plus en détail, Schneider Electric est le numéro 1 mondial dans le contrôle industriel (ex. : détecteur de mouvement) ; numéro 3 mondial en automatismes (ex. :robot d'usine) ; numéro 4 mondial en automatismes du bâtiment.

- Exemples d'offres

L'entreprise propose des produits destinés à contrôler et à alimenter les équipements : contacteurs, relais thermiques, etc.

Le groupe propose aussi des solutions d'automatisation répondant à des problématiques spécifiques telles que des automates programmables, des logiciels de paramétrage et des réseaux de communication. Avec le contrôle de mouvement, le groupe couvre également toutes les phases de procédure de commande de mouvement pour les machines automatiques (objets-robots, véhicules, etc.).

3.7 Choix de l'API:

Après l'établissement du cahier des charges, il revient à l'utilisateur de regarder sur le marché l'automate le mieux adapté aux besoins, en considérant un certain nombre de critères importants :

- Le nombre et la nature des E/S.
- La nature du traitement (temporisation, comptage, ...).
- Les moyens de dialogue et le langage de programmation.
- La communication avec les autres systèmes.
- Les moyens de sauvegarde du programme.
- La fiabilité, robustesse, immunité aux parasites.
- La documentation, le service après-vente, durée de la garantie, laformation

3.8 Conclusion :

Apres avoir détaillé l'élément majeur pour la réalisation de notre Superviseur nousnous sommes dirigés vers l'utilisation de l'API SIEMENSE CPU 312, ce choix a été fait pour deux raisons, la première était surtout du fait de sa disponibilité au sein de notre LABO d'automatisme et la seconde suite aux choix de nos application qui s'adapte à ces APIs.

Suite à cette étude et au choix prit, nous entamerons dans le chapitre qui suit notre réalisation et simulation.

Chapitre 4 :la réalisationdu projet

4.1 Introduction :

Pour suivre l'évolution d'un system automatisé en temps réel et simplifié la tache a l'operateur (qui n'est pas forcémentun automaticien ou/et programmeur) , plusieurs solutions peuvent être misent en œuvre, dont la meilleure est la visualisation ,qui est basée sur l'interface graphique (en anglais :GUI pour Graphical User Interface) . dans lequel les objets sont schématisés et facilement manipulés sur un écran.

le développement d'une interface graphiqueest possible grâce a des langages de programmation tel que JAVA et CSharp

Dans ce chapitre on va parler du déroulement et du développement d'une plateforme de supervision

4.2 Partie matériel (hardware)

4.2.1 Automate utilisé :

L'interface qu'on a développé sert a supervisé le travail de deux automates programmable de type S7-300 CPU 312 du fabriquant SIEMENS(figure31).

Figure 30 :API du SIEMENSE_S7-300_CPU-312

4.2.2 Mise en réseau :

Afin de pouvoir communiquer en même temps avec nous deux API avec un seul pc on a relié les deux API avec un câble Profibus et un pc adapter qui est connecter a l'un des deux avec un connecteur Profibus (figure 32).

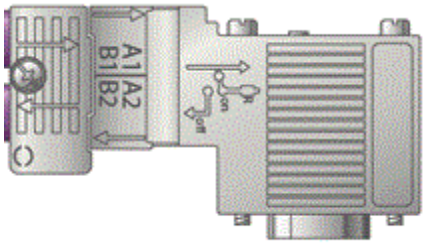


Figure 31 :connecteur Profibus avec sortie mal

Pour relier les deux APIs avec un seul PC il est nécessaire d'utilisé un PC-Adapter
La figure suivante montre un schéma de l'installationréalisée (figure33).

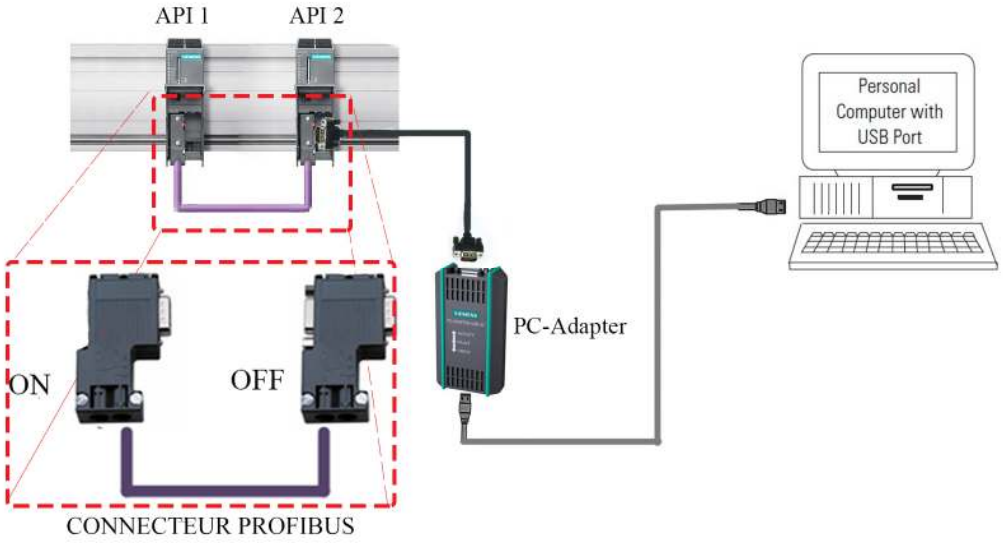


Figure 32 :L'installation mise pour le projet

Remarque :
Pour le bon fonctionnement du montage ci-dessus il faut respecter les étapes suivants :

- Maitre une adresse unique pour Chaque participant du réseau MPI
- Les connecteurs Profibus connecté au réseau doivent avoir leur résistance mise a la position OFF sauf celle du dernier de la chaine qui doit être mise a la position ON pour crée une boucle fermer.
- La console de programmation (PC) doit êtremunie d'un logiciel (plateforme de programmation) compatible avec l'API. SIEMENSE offre le logiciel TIA PORTAL.

4.3 partie programmation (software) :

4.3.1 introduction aux environnements de développement :

4.3.1.1 introduction au TIA portal:

La plateforme TIA Portal (Totally integrated Automation) est un environnement de travail Siemens qui permet de maitre en œuvre des solution d'automatisation avec un système d'ingénierieintégrécomprenant les logiciel SIMATIC STEP 7 et SIMATIC Win CC.

Lorsque l'on lance le logiciel TIA Portal, l'environnement de travail se décompose e deux types de vue :

a) **La vue de portail** qui est l'axée sur les taches àexécuter avec une prise de main rapide (figure 34)

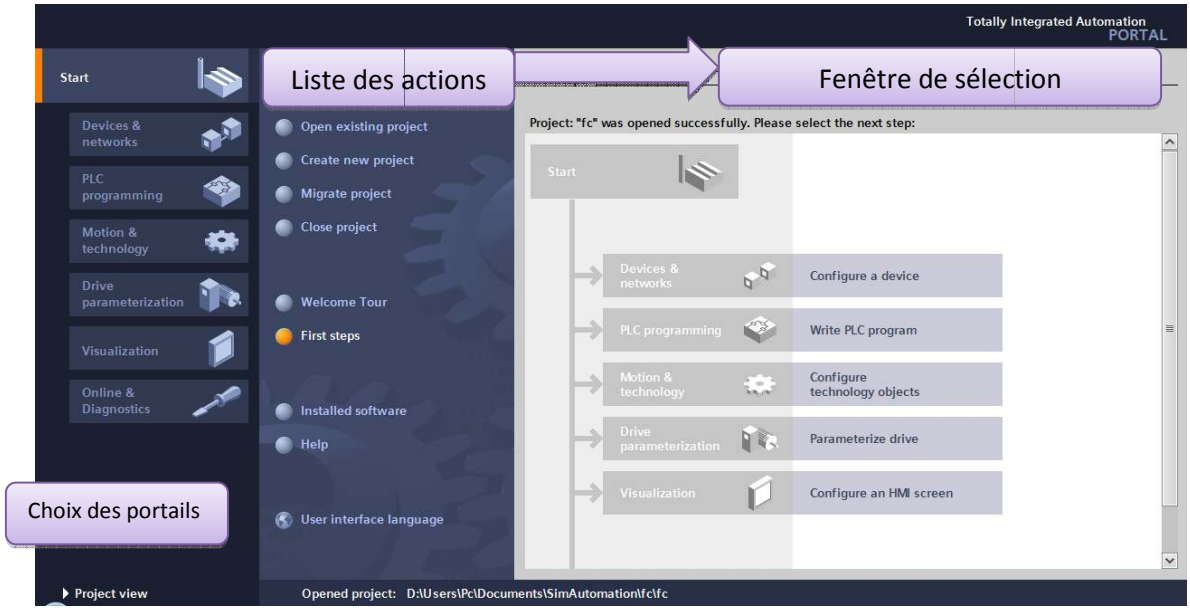


Figure 33 : vue de portail

b) la vue du projet qui comporte une arborescence avec les différentséléments du projet (figure35)

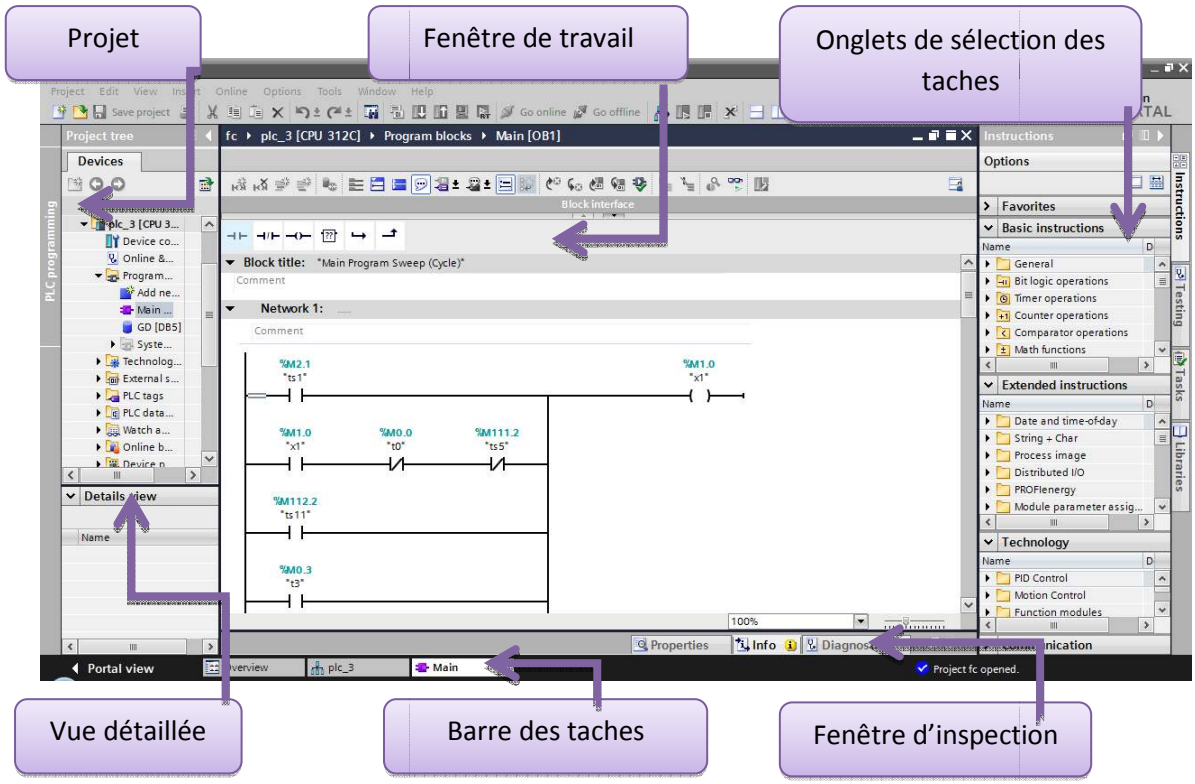


Figure 34 :vue de projet

La fenêtre de travail permet de visualiser les objets sélectionnés dans le projet pour être traités. Il peut s'agir des composants matériels, des blocs de programme, des tables des variables, des IHM,…

La fenêtre d'inspection permet de visualiser des informations complémentaires sur un objet sélectionné ou sur les actions en cours d'exécution (propriété du matériel sélectionné, messages d'erreurs lors de la compilation des blocs de programme,…).

Les onglets de sélection de tâches ont un contenu qui varie en fonction de l'objet sélectionné (configuration matérielle => bibliothèques des composants, bloc de programme => instructions de programmation).

Cet environnement de travail contient énormément de données. Il est possible de masquer ou réduire certaines de ces fenêtres lorsque l'on ne les utilise pas.

Il est également possible de redimensionner, réorganiser, désancrer les différentes fenêtres.

4.3.1.2 Création d'un projet et configuration d'une station de travail

a) Création d'un projet

pour créer un projet il faut Just cliquer sur le bouton « Create » (figure 36)

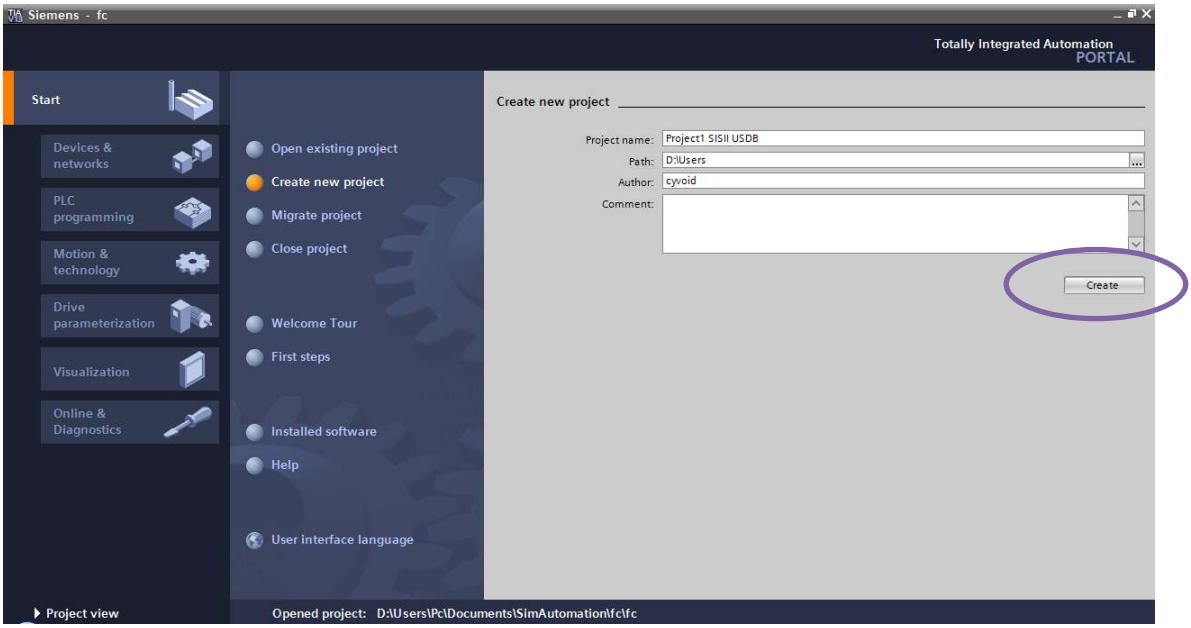


Figure 35 :création d'un projet

b) Configuration et paramétrage du matériel

Une fois votre projet crée, on peut configurer la station de travail. (figure37)

La première étape consiste à définir le matériel existant. Pour cela, on peut passer par la vue du projet et cliquer sur « ajouter un appareil » dans le navigateur du projet.

La liste des éléments que l'on peut ajouter apparait (API, HMI, système PC). On commencera par faire le choix de notre CPU pour ensuite venir ajouter les modules complémentaires (alimentation, E/S TOR ou analogiques, module de communication AS-i,…).

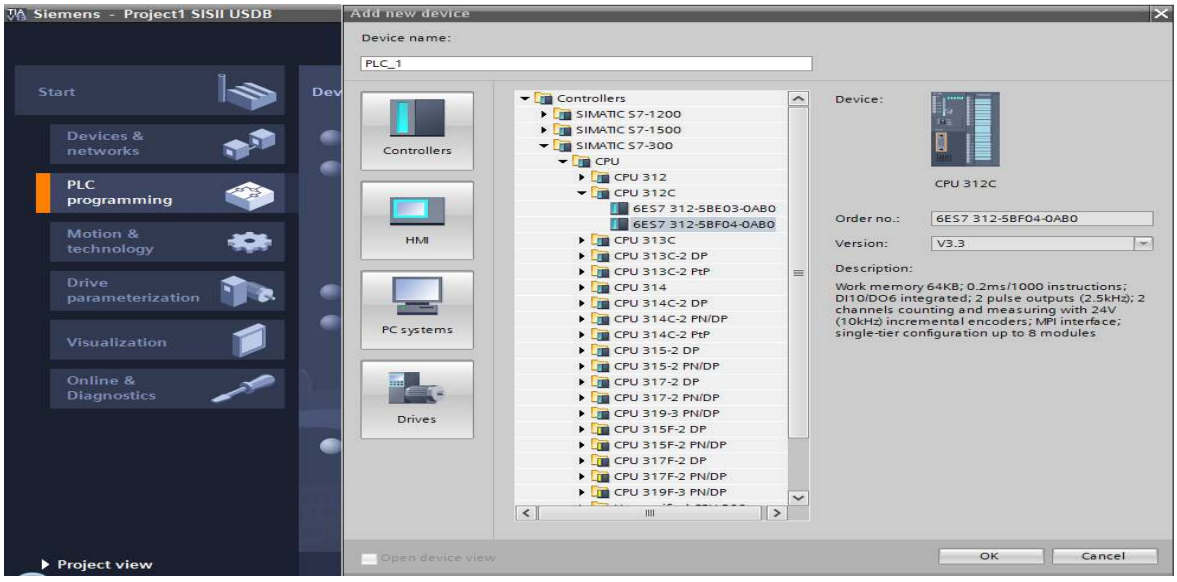


Figure 36 :choix du matériel

Les modules complémentaires de l'API peuvent être ajoutés en utilisant le catalogue. Si l'on veut ajouter un écran ou un autre API, il faut repasser par la commande « ajouter un appareil » dans le navigateur du projet. Lorsque l'on sélectionne un élément à insérer dans le projet, une description est proposée dans l'onglet information. (figure38).

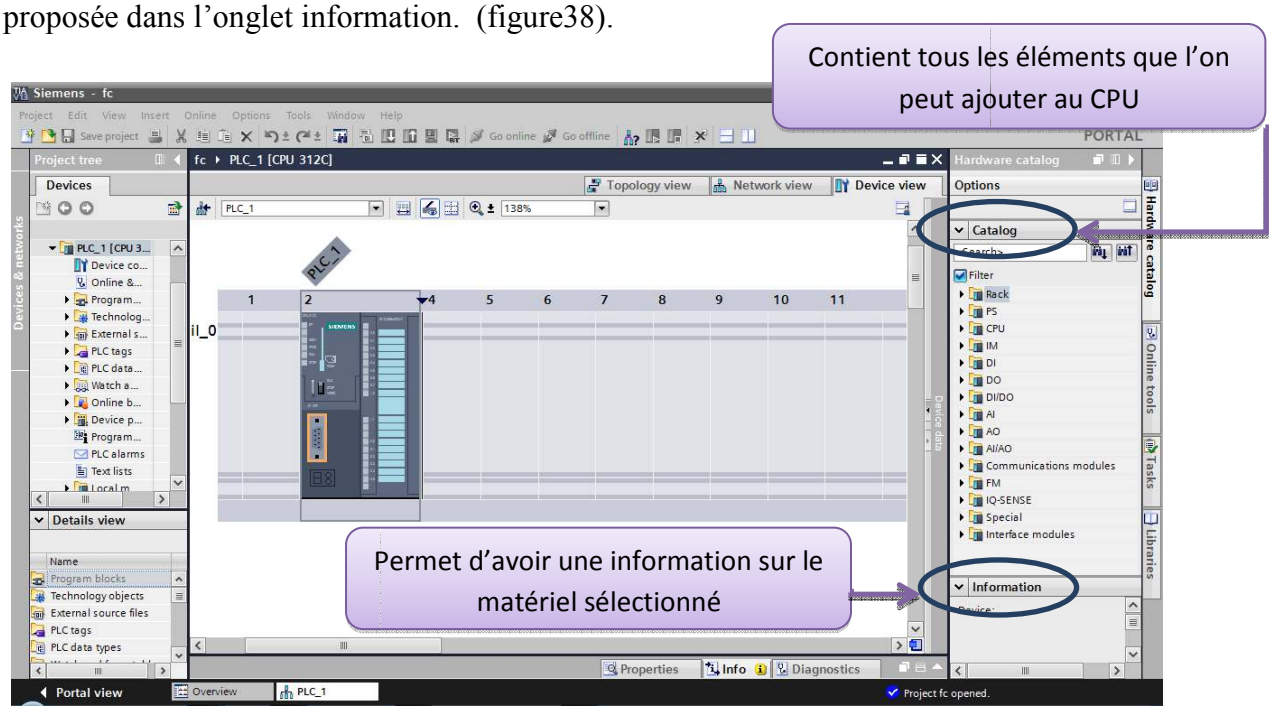


Figure 37 : configuration du l'API

c) Adressage des E/S

Pour connaitre l'adressage des entrées et sorties présentes dans la configuration matériel, il faut aller dans « appareil et réseau » dans le navigateur du projet.

Dans la fenêtre de travail, on doit s'assurer d'être dans l'onglet « Vue des appareils » et de sélectionner l'appareil voulu.(figure39).

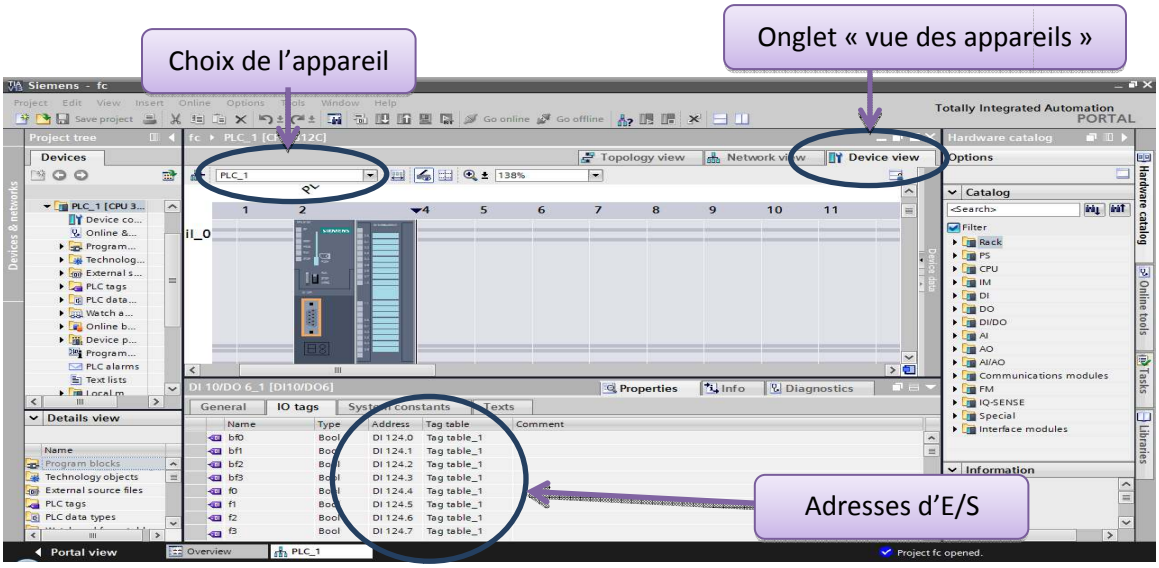


Figure 38 : affectation des adresses

On sélectionne la CPU Les adresses des entrées et sorties apparaisse. Vous pouvez les modifier en entrant une nouvelle valeur dans la case correspondante.

d) Adresse de la CPU

Toujours dans les propriétés de la CPU, il est possible de définir son adresse. Un double clic sur le connecteur MPI de la station fait apparaitre la fenêtre d'inspection permettant de définir ses propriétés.

Pour établir une liaison entre la CPU et la console de programmation, il faut affecter aux deux appareils des adresses appartenant au même réseau. L'adresse par défaut de l'API est de 2, on peut la modifier selon les besoins (figure40).

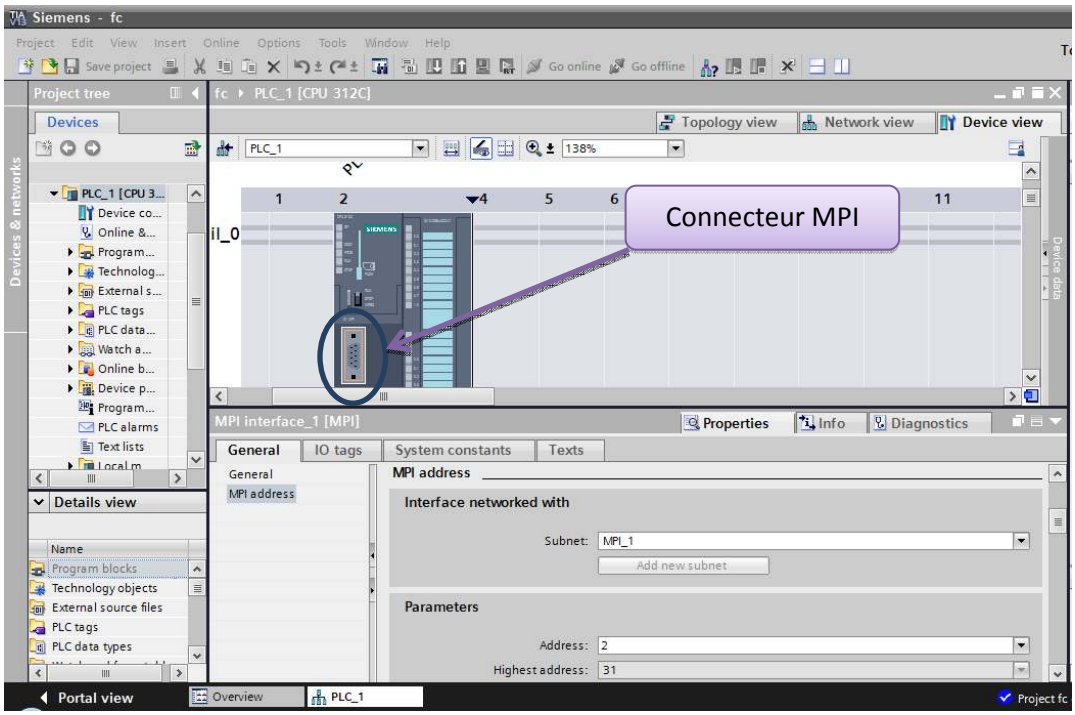


Figure 39 :configuration l'interface MPI de l'API

e) Compilation et chargement de la configuration matérielle

Une fois la configuration matérielle réalisée, il faut la compiler et la charger dans l'automate.
La compilation se fait à l'aide de l'icône « compiler » de la barre de tâche. On sélectionne l'API dans le projet puis cliquer sur l'icône « compiler ».
En utilisant cette manière, on effectue une compilation matérielle et logicielle.
Une autre solution pour compiler est de faire un clic droit sur l'API dans la fenêtre du projet et de choisir l'option « Compiler => Configuration matérielle »(figure41).

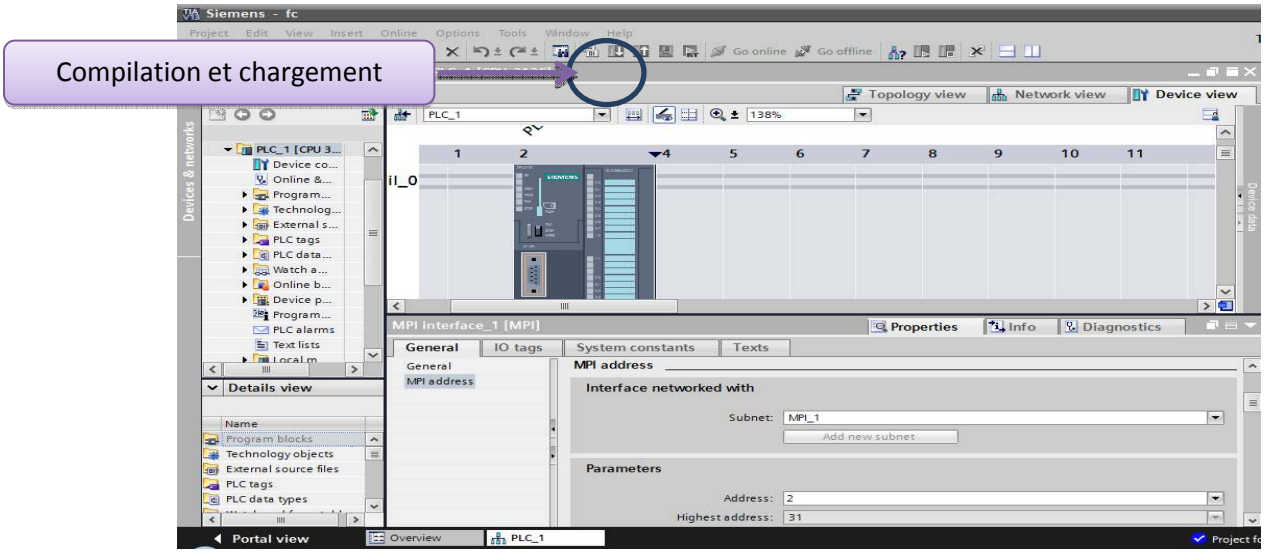


Figure 40 : compilation et chargement

Pour charger la configuration dans l'automate, on effectue un clic sur l'icône « charger dans l'appareil ». La fenêtre ci-dessous s'ouvre et vous devez faire le choix du mode de connexion (PN/IE, Profibus, MPI). Si vous choisissez le mode MPI, l'API doit posséder une adresse MPI.

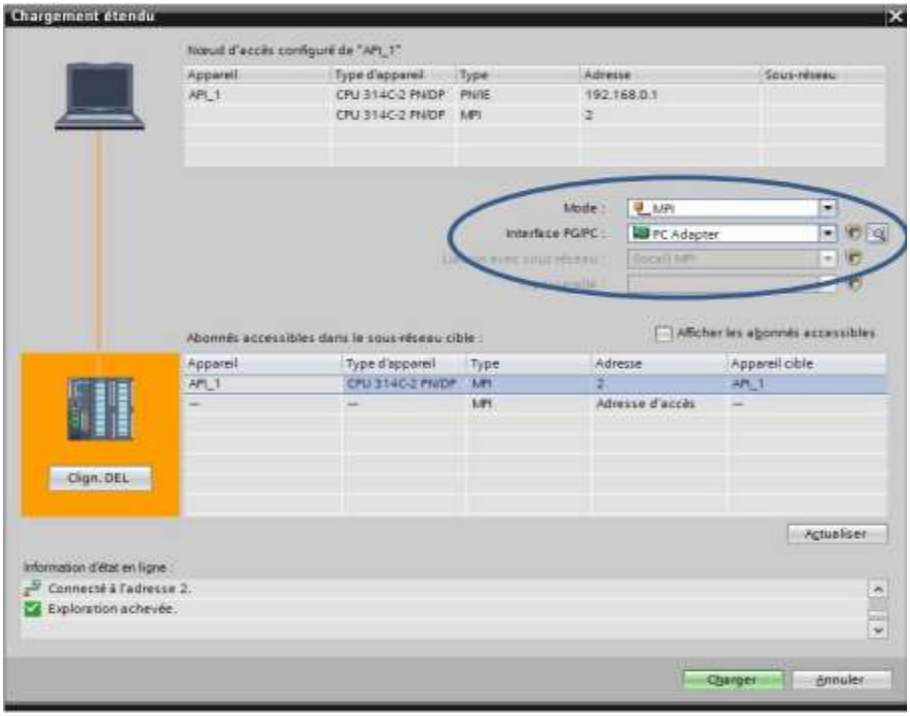


Figure 41 : propriété de connexion

Pour une première connexion ou pour charger l'adresse désirée dans la CPU, il est plus facile de choisir le mode de connexion MPI et de relier le PC à la CPU via le « PC Adapter ». (figure42)
Si le programme trouve un appareil, il figure dans la liste en bas de la fenêtre. La touche « Clign. DEL » permet de faire clignoter une LED sur la face avant de l'appareil afin de s'assurer que l'on est connecté à l'appareil désiré.

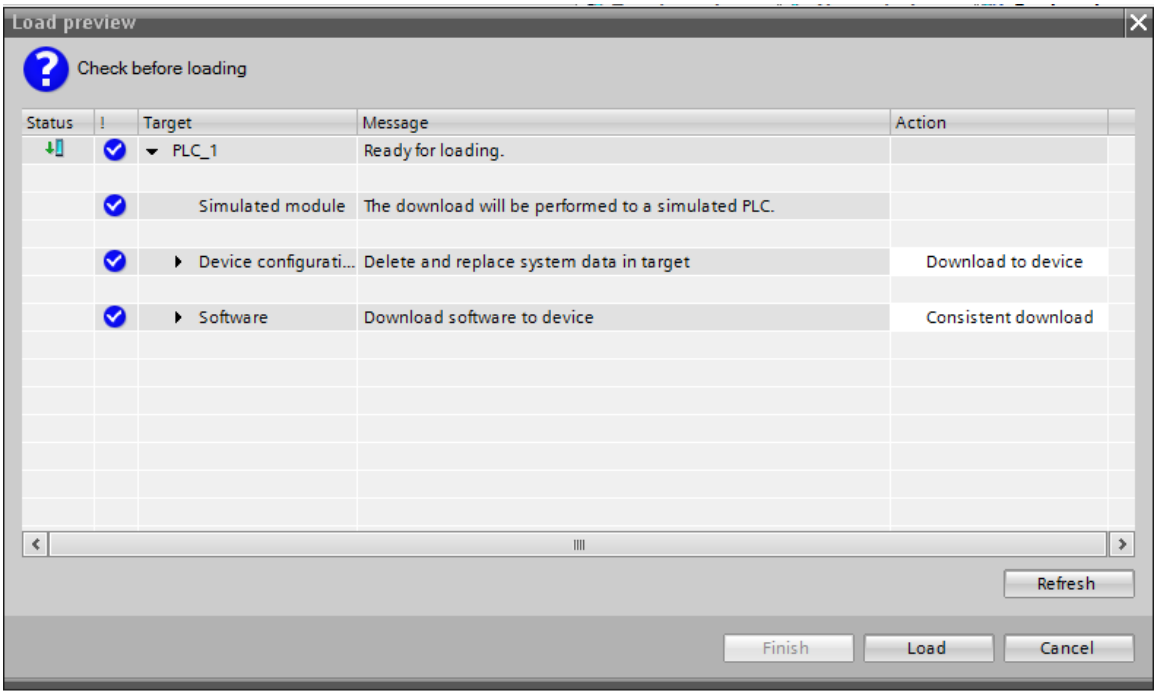


Figure 42 : chargement du programme

Une fois la configuration terminée, on peut charger le tout dans l'appareil (figure42). Des avertissements / confirmations peuvent être demandés lors de cette opération. Si des erreurs sont détectées, elles seront visibles via cette fenêtre(figure43). Le programme ne pourra pas être chargé tant que les erreurs persistent.

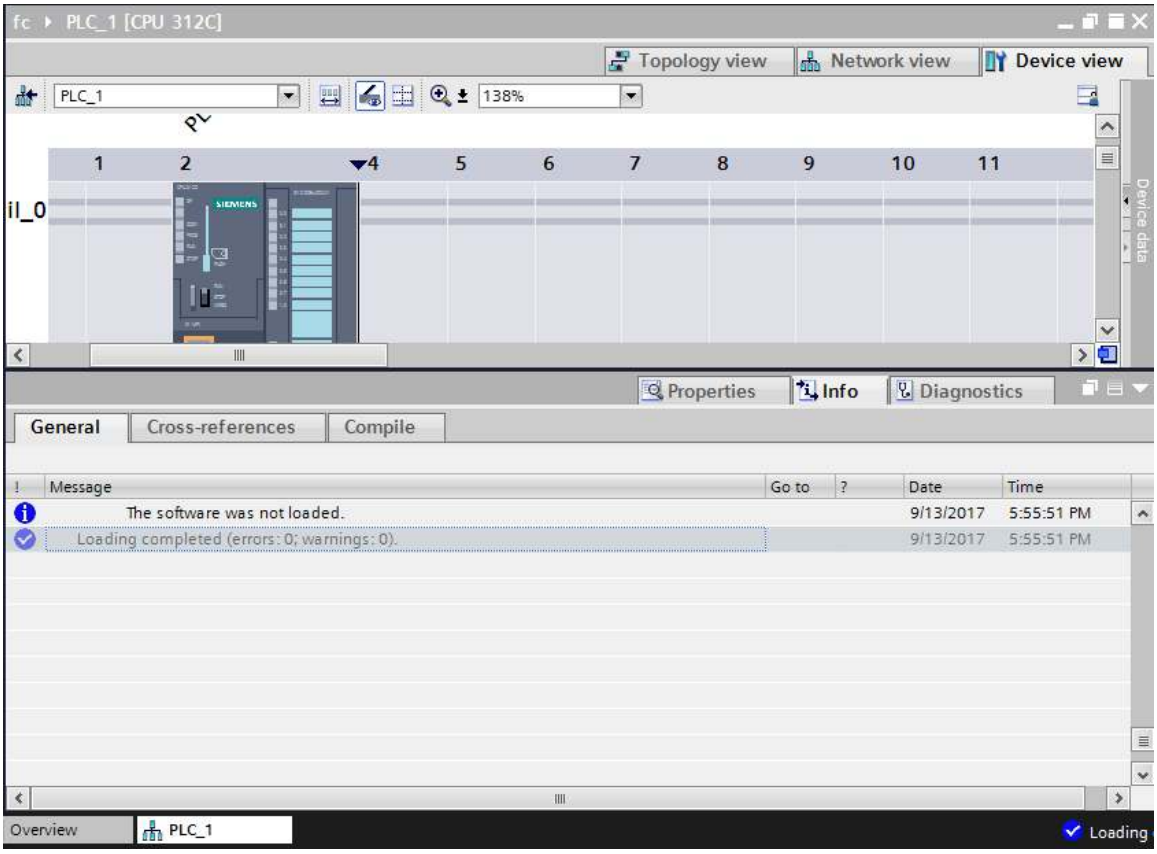


Figure 43 : affichage des erreurs de compilation

f) Les variables API

Dans TIA Portal, toutes les variables globales (entrées, sorties, mémentos,..) possède une adresse symbolique et une adresse absolue.

- L'adresse absolue représente l'identificateur d'opérande (I, Q, M,…) et son adresse et numéro de bit.
- L'adresse symbolique correspond au nom que l'utilisateur a donné à la variable (ex : Bouton Marche).

Le lien entre les adresses symbolique et absolue se fait dans la table des variables API.

Lors de la programmation, on peut choisir d'afficher les adresses absolues, symboliques ou encore les deux simultanément. (figure44).

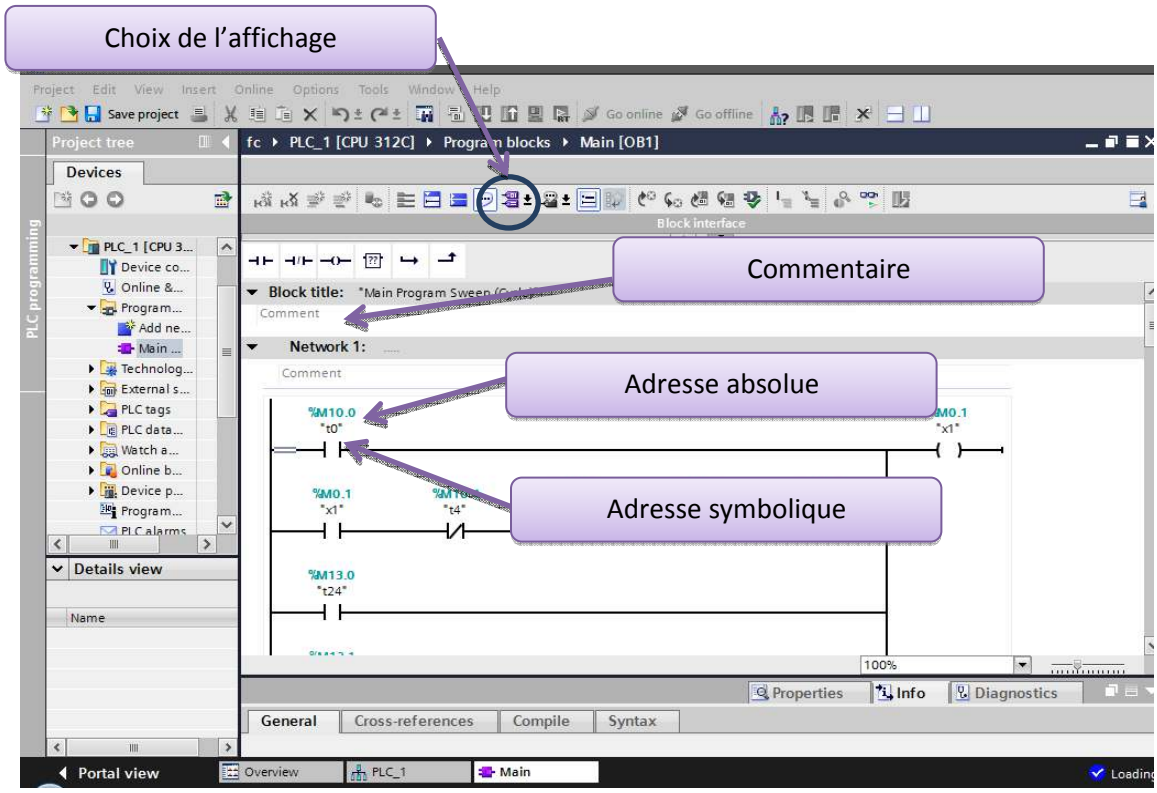


Figure 44 : type de variables

g) Table des variables API

C'est dans la table des variables API que l'on va pouvoir déclarer toutes les variables et les constantes utilisées dans le programme.

Lorsque l'on définit une variable API, il faut définir :

- Un nom : c'est l'adressage symbolique de la variable.
- Le type de donnée : BOOL, INT,…
- L'adresse absolue : par exemple Q124.3

On peut également insérer un commentaire qui nous renseigne sur cette variable. Le commentaire peut être visible dans chaque réseau utilisant cette variable. (figure46).

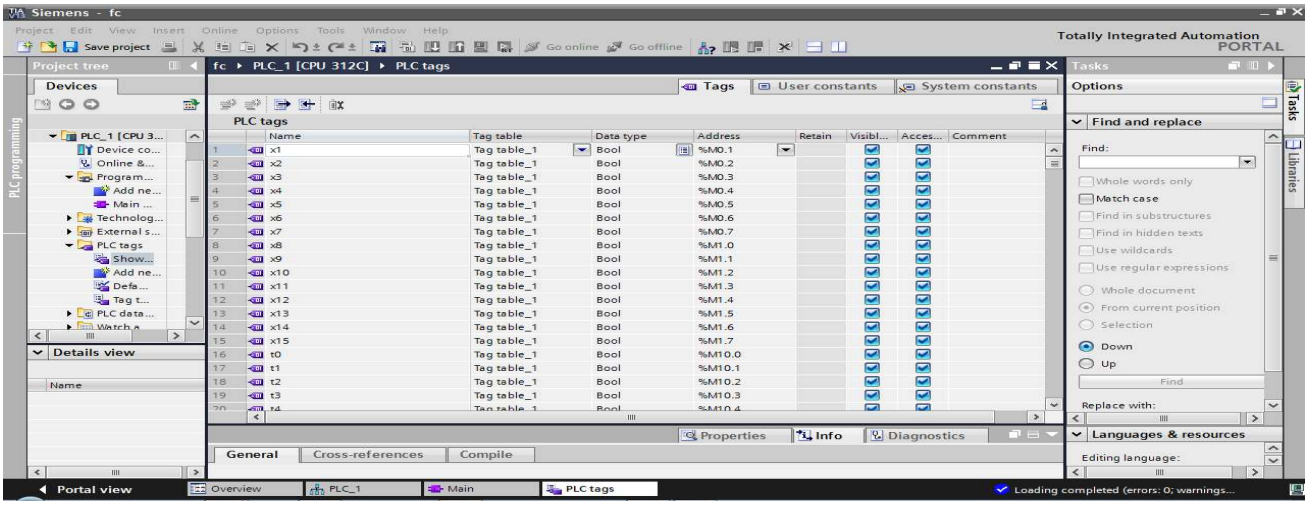


Figure 45 : table des variables

4.3.1.3 Mise en œuvre du travail dans TIA PORTAL

Apres l'introduction au logiciel, les pages qui vont suivre démontrent les étapes qu'on a suivies pour programmer les deux API.

a) Programmation de l'API 1 (feu tricolore) :

i. La mise en œuvre du GRAFCET

Le cahier de charge qu'on a fait nécessite une traduction en GRDAFCT pour mieux comprendre et schématiser le déroulement(figure47).

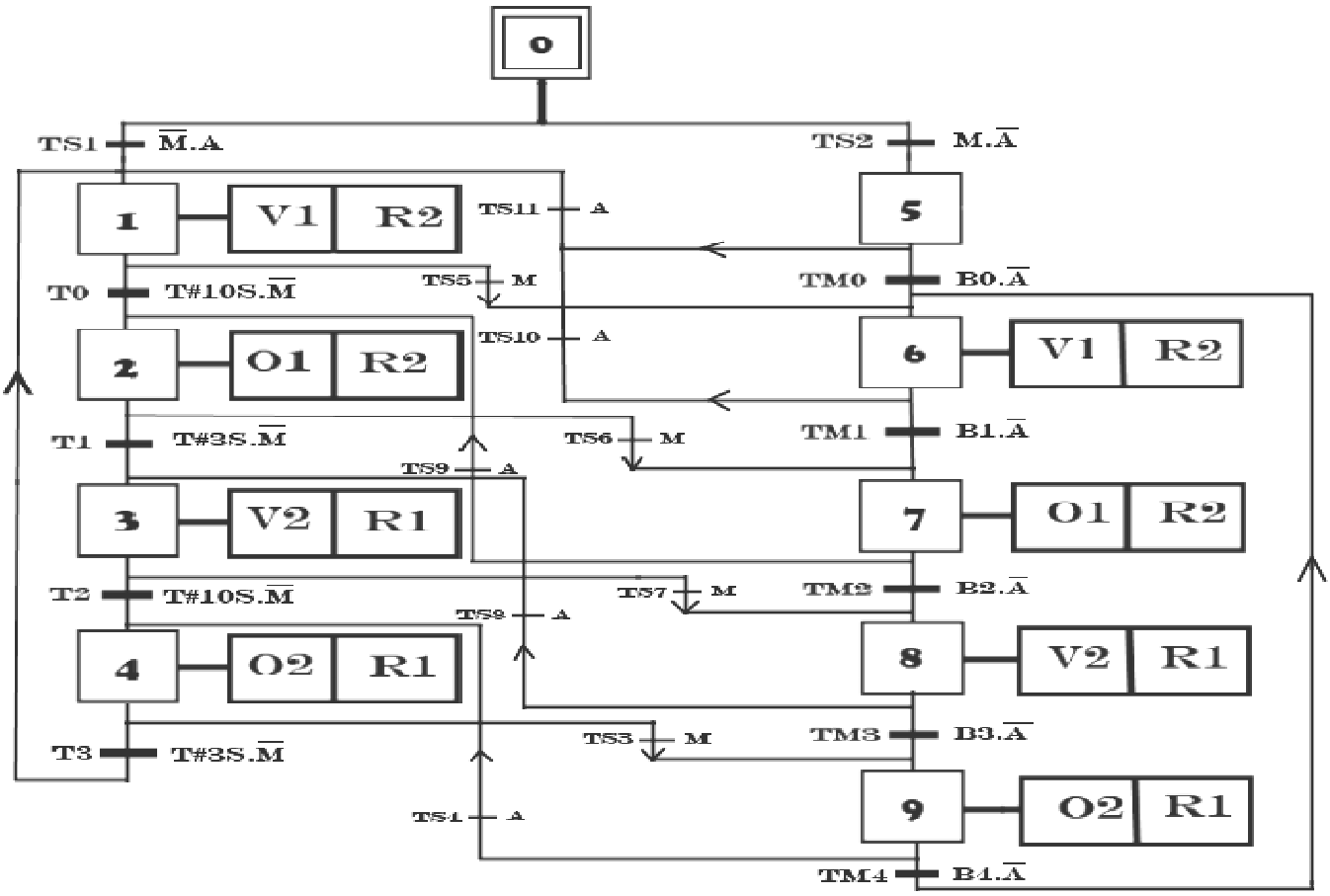


Figure 46 : grafcet de feu tricolore

ii. Mise en équation du grafcet :

Pour faciliter la traduction du grafcet au Lader il est conseillé d'écrire les équations du grafcet. (tableau 2)

Tableau 2 :équation grafcet

Equation d'étapes	Equations de transition	Equation de sorties
X1=ts1+t3+x1.t0 \+ts4 X2= t0 +x2.t1\ X3= t1 + x3.t2\ X4= t2+x4\.t3\.ts3\ X5=ts3+ts2+x5.tm0\ X6=tm0+tm4+x6.tm1\ X7=tm1+x7.tm2\ X8=tm2+x8tm3\ X9=tm3+x9.ts4\.tm4\	Tm0=x5.b0.A\ Tm1=x6.b1.A\ Tm2=x7.b2.A\ Tm3=x8.b3.A\ T0=x1.t#10s.M\ T1=x2.t#3s.M\ T2=x3.t#10s.M\ T3=x4.t#3s.M\ Ts5=x1.M Ts6=x2.M Ts7=x3.M Ts8=x8.A Ts9=x7.A Ts10=x6.A	V1=x6+x1 O1=x2+x7 R1=x3+x4+x8+x9 V2=x3+x8 O2=x4+x9 R2=x1+x2+x6+x7

b) Programmation de L'API 2 (Ascenseur a troisétages) :

i. mise en œuvre du GRAFCET :

le schema suivant montre le GRAFCET de l'ascenseur (figure48).

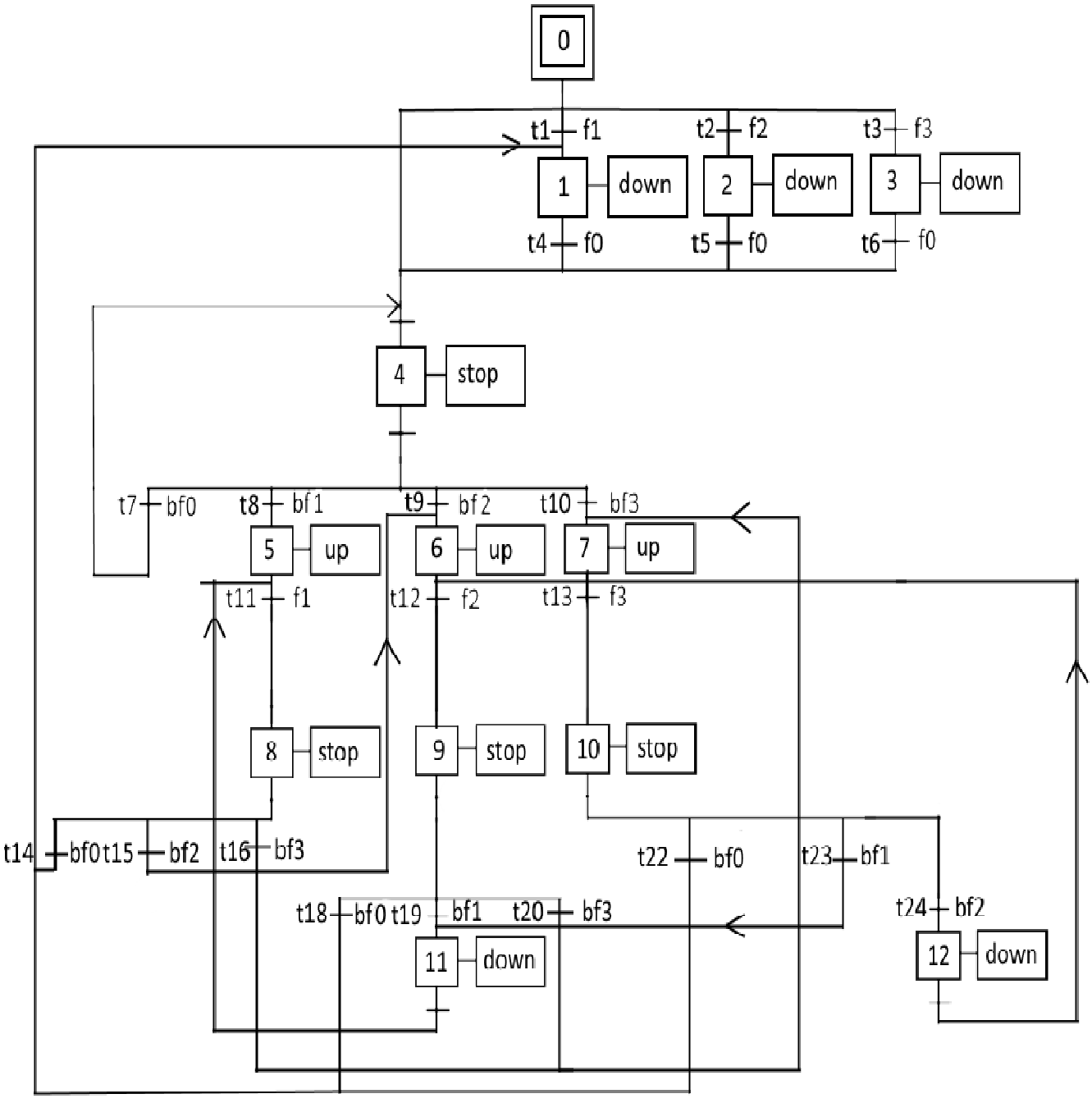


Figure 47 : grafcet de l'Ascenseur a trois étages

ii. Mise en équation du grafcet :

Tableau 3 : équations grafcet

équations d'étapes	équation de transitions		équation de sorties
x0=init+t24+t25+t26+x0.$\overline{t0}$.$\overline{t1}$.$\overline{t2}$.$\overline{t3}$	t0=x0.f0	t16=x10.bf2	ds=x2+x3+x4x+x13+x14+x15
x1=t0+x1.$\overline{t4}$+$\overline{t5}$+$\overline{t6}$+$\overline{t7}$	t1=x0.f1	t17=x10.bf3	mt=x6+x7+x8
x2=t1+x2.$\overline{t24}$	t2=x0.f2	t18=x11.bf0	st=x1+x5+x19+x10+x11+x12
x3=t2+x3.$\overline{t25}$	t3=x0.f3	t19=x11.bf1	
x4=t3+x4.$\overline{t26}$	t4=x1.bf0	t20=x11.bf3	
x5=t4+x5.$\overline{t8}$	t5=x1.bf1	t21=x12.bf0	
x6=t5+t12+x6.$\overline{t9}$	t6=x1.bf2	t22=x12.bf1	
x7=t13+t16+t6+x7.$\overline{t10}$	t7=x1.bf3	t23=x12.bf2	
x8=t7+t14+t20+x8.$\overline{t11}$	t8=x5	t24=x2.f0	
x9=t8+x9.t12.t13.$\overline{t14}$	t9=x6.f1	t25=x3.f0	
x10=t9+x10.t15.$\overline{t16}$.$\overline{t17}$	t10=x7.f2	t26=x4.f0	
x11=t10+x11.t18.$\overline{t19}$.$\overline{t20}$	t11=x8.f3		
x12=t11+x12.t21.$\overline{t22}$.$\overline{t23}$	t12=x9.bf1		
x13=t19+x13.$\overline{t9}$	t13=x9.bf2		
x14=t22+x14.$\overline{t9}$	t14=x9.bf3		
x15=t23+x15.$\overline{t10}$	t15=x10+bf0		

Remarque

On a choisit d'utiliser le langage graphique LADER afin d'implémenter le programme dans l'API.

La traduction des deux GRAFCET feu tricolore et de l'ascenseur en LADER est mentionner dans l'Annexe1 et Annexe 2 respectivement

Afin d'atteindre notre but qui consiste a maitre en point une interface graphique qui traduit le programme (lader) en une animation sur un écran (PC) on a utilisé l'environnement de développement Visual Studio(IDE) qui offre plein de possibilités et moyen de communication avec les automates programmables de siemens.

4.3.2 Introduction à l'environnementVisual studio :

Les environnements de développement intégré (Integrated developpement environnements, IDE) offrent des outils permettant de développer différents types d'application (Windows, Web,

Android,IOS, etc.) en utilisant de nombreux langages de programmation comme C#,VB.NET,C++,Java et bien plus.

4.3.2.1 Présentation de L'IDE de Visual Studio :

Apres le lancement du Visual Studioon a le choix entre ouvrir un projet existant et cree un nouveau projet (figure 49)

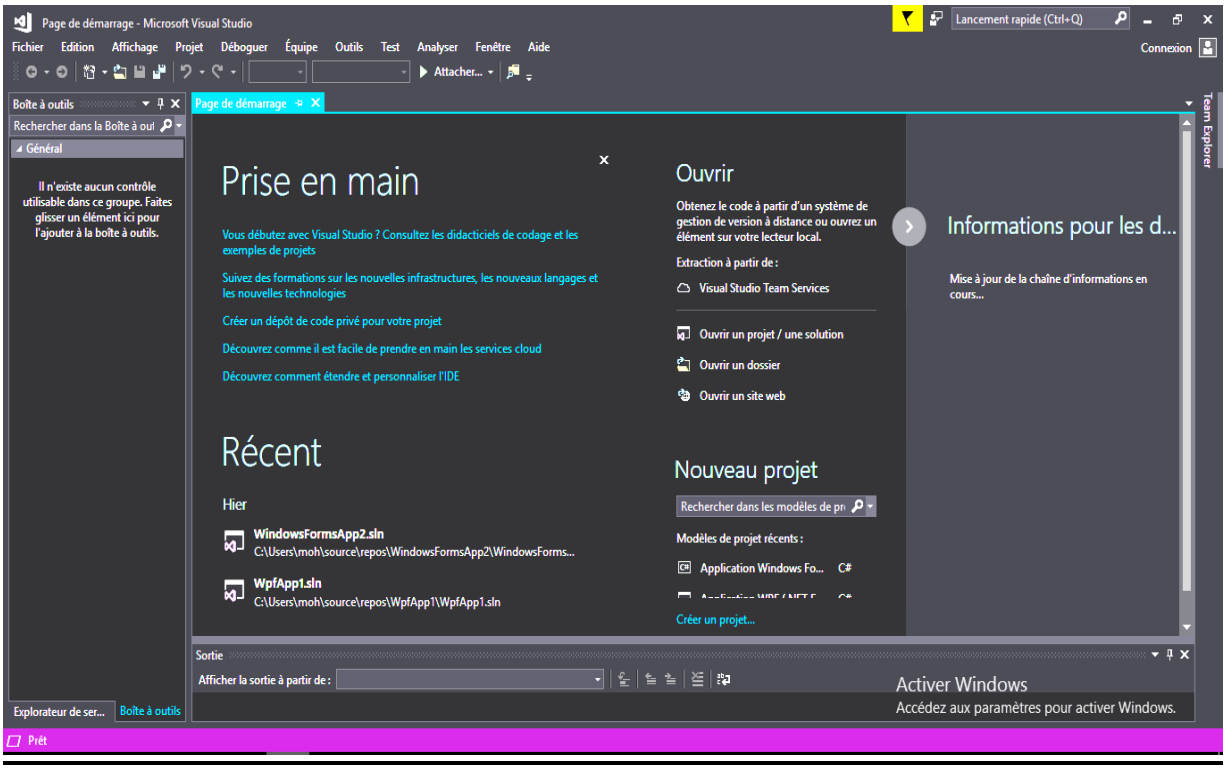


Figure 48 : fenêtre principale du Visuel Studio

Création d'un nouveau projet (figure50).

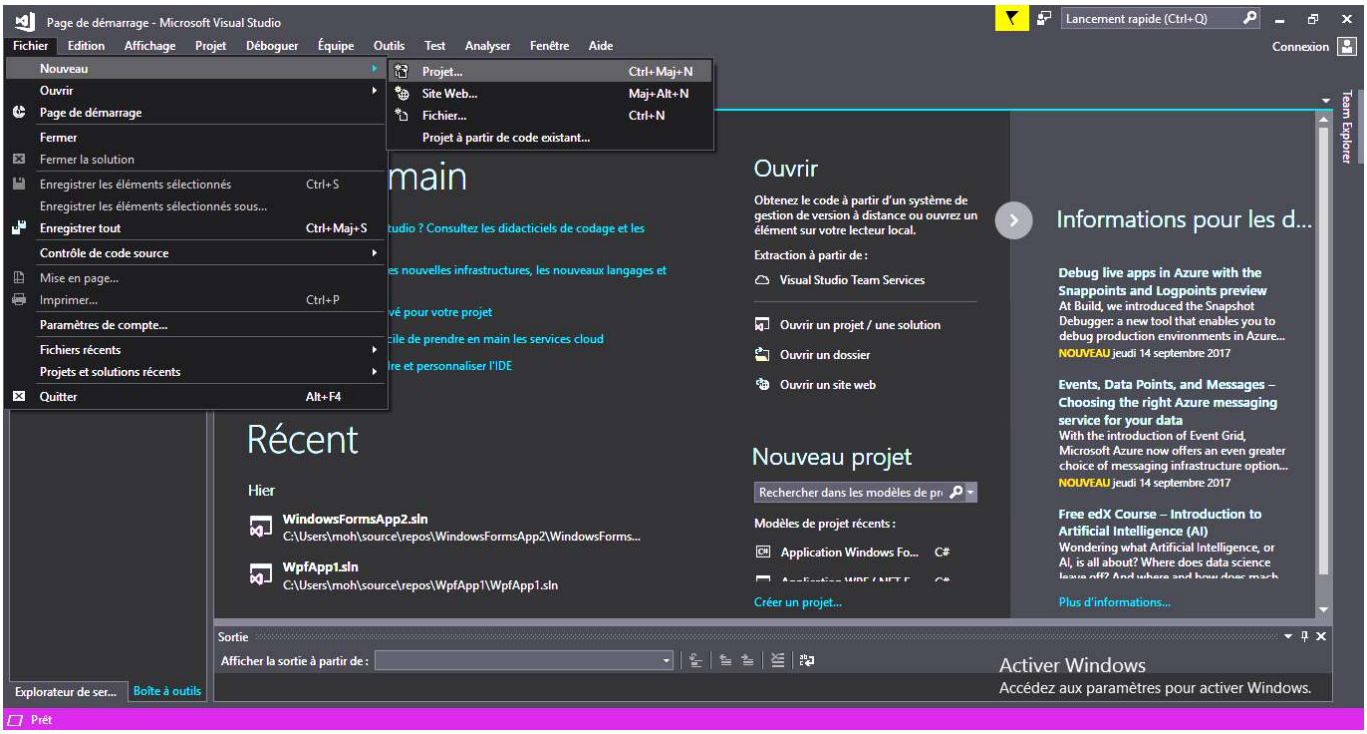


Figure 49 :création de projet

Apres la création du projet une fenêtre de propriétés s'ouvre (figure51).

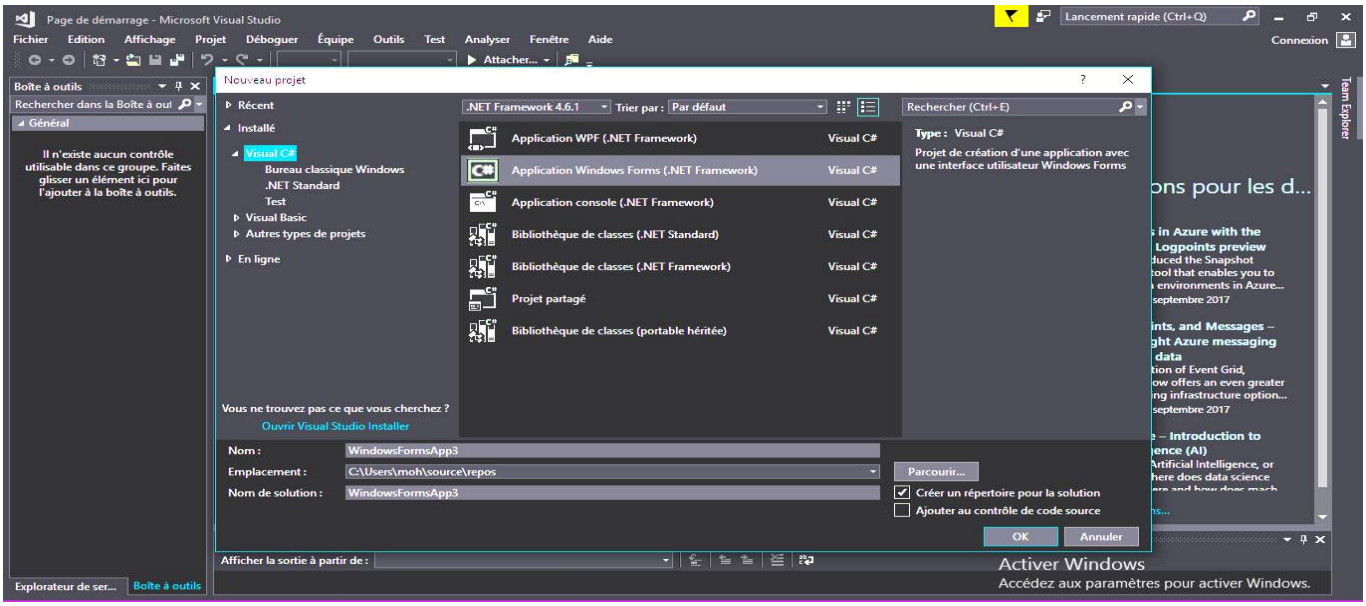


Figure 50 : les propriétés et le type du projet

Dans La fenêtre si dessus on sélection la propriété du projet basé sur quelle type d'application on veut crée

Ona le choix entre :

- ✓ Application WPF
- ✓ Application Windows Form...etc.
- ✓ Application console

Dans notre cas ont choisi la deuxième option. Ladernière fenêtre qui s'ouvre contient l'espace ou on va développer des applications (figure 52)

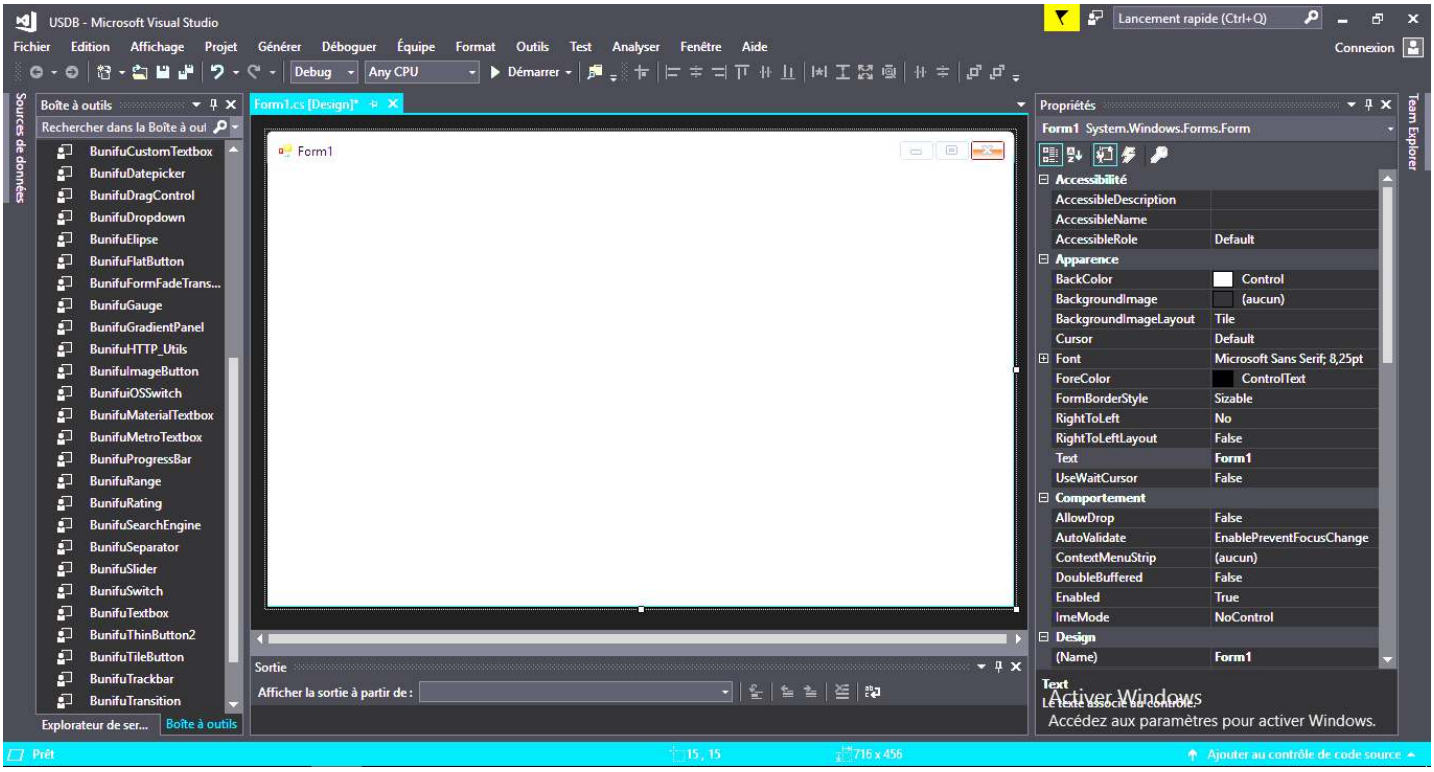


Figure 51 :L'espace de developpement

4.3.3 Conception de la solution:

4.3.3.1 Avec le langage Java :

- **Bibliothèque utilisé :**
  Si on veut visualiser des processus contrôler par un API du fabriquent siemens alors Libnodave représente un bon choix
- **Libnodave**
  Libnodave est une bibliothèque libre pour l'échange de données entre un PC et l'automate programmable de siemens, libnodave a été écrite avec le langage C++ et peu communiquer via MPI,PPI et ISO sur TCP.
- **Echange de données avec l'API Siemens :**
  Les ressources téléchargées par la bibliothèque LibNoDave contiennent des classes java NoDave et des classes de test java. Avec les classes java NoDave, on peut créer l'interface requise et le type de connexion (MPI, PPI et ISO sur TCP).
  Pour utiliser les classes de libnodave on doit les ajouter dans notre projet tel qu'indiquer sur l'image ci-dessous (figure 53)

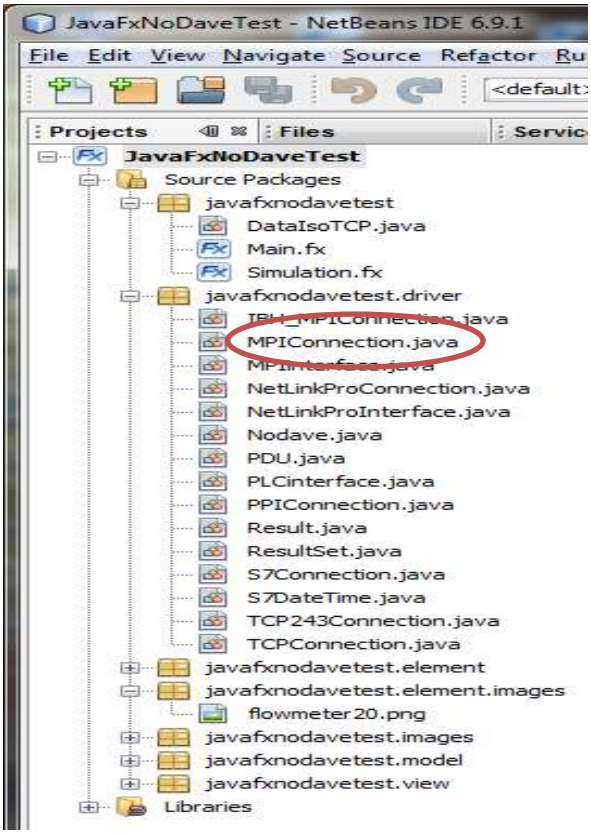


Figure 52 : utilisation de libnodave.

La classe MPIConnection comporte des fonctions pour connecter, déconnecter et échangerdes données.

- **Les limites de libnodave**

  Pour communiquer avec les automates programmables de siemens via l'interface MPI libnodave se limite à trois adaptateur MPI (figure 54)

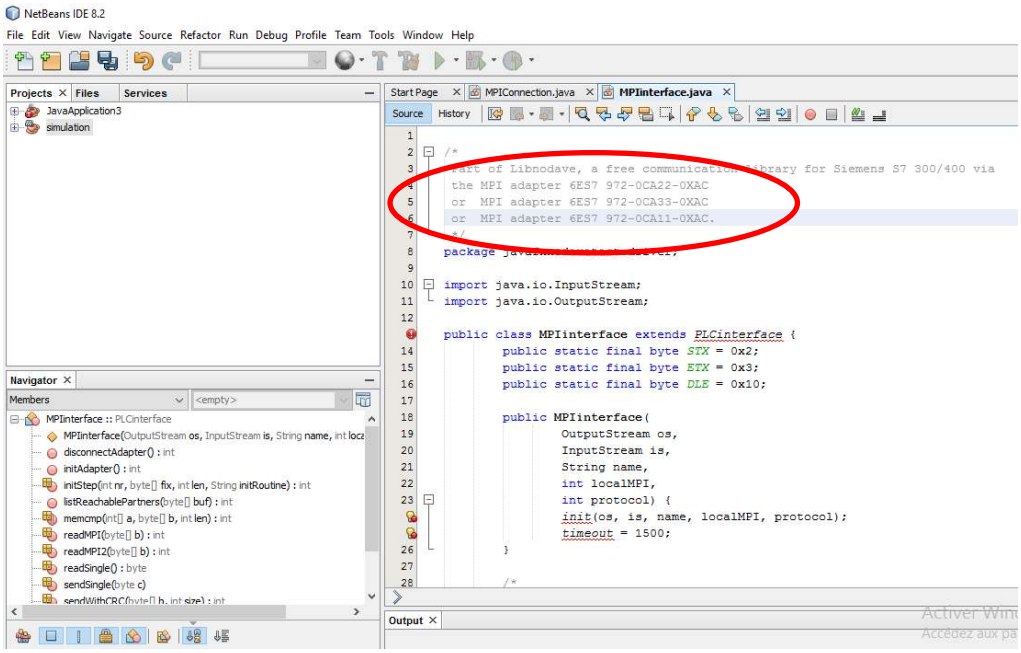


Figure 53 : adaptateur MPI

Sans ces référence de l'adaptateur (figure54) la communication entre notre application et les API ne s'établit pas ce qui nous a obligés de passer à la simulation.

4.3.3.2 Simulation

La simulation consiste à concevoir un modèle du système (réel) étudié et de mener des expérimentations sur ce dernier afin d'interpréter les résultats fournis par le déroulement du modèle puis formuler des décisions relatives au système, selon les besoins.

Notre système est composée de deux API simulé à l'aide de l'outil PLCSIM de TIA Portal (figure 55) et notre interface graphique développé avec l'environnement Visual Studio .

Figure 54 :PLCSIM

A. Introduction a S7PROSIM :

S7PROSIM est une interface de programmation ou API (**A**pplication **P**rogramming **I**nterface) qui fournit un Access a l'interface API simulé du S7-PLCSIM qui simule une connexion MPI par défaut c'est l'outil qu'on vas intégrer avec Visual Studio pour établir une connexion entre PLCSIM et notre application.(figure 56).

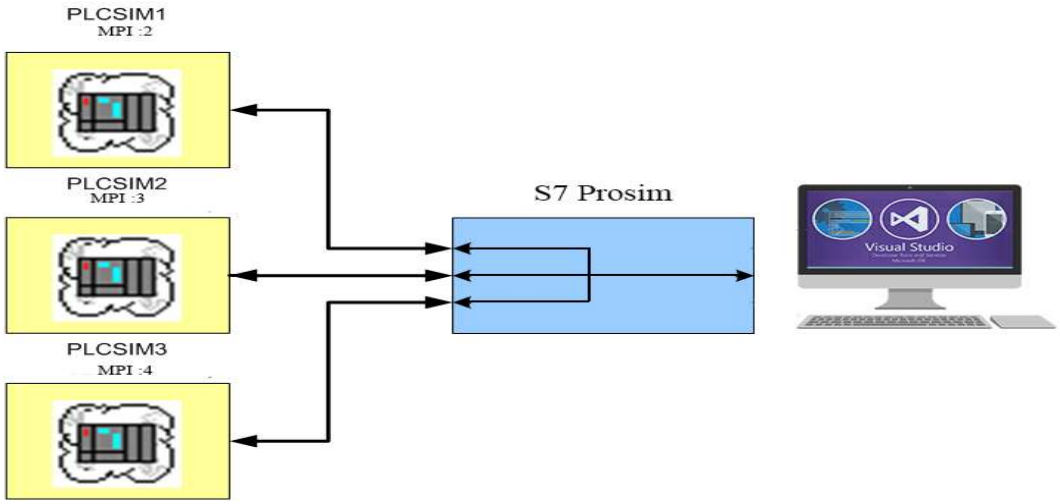


Figure 55 :Interface S7 Prosim

4.3.3.3 Présentations de l'application

- **Paged'accueil :**

Figure 56 : Page d'accueil

Cette fenêtre nous donne l'accès a la vue IHM de feu tricolore (figure 55) et à la vue IHM de L'ascenseur (figure 56) en cliquent sur les boutons respectif Feu tricolore, Elevator.

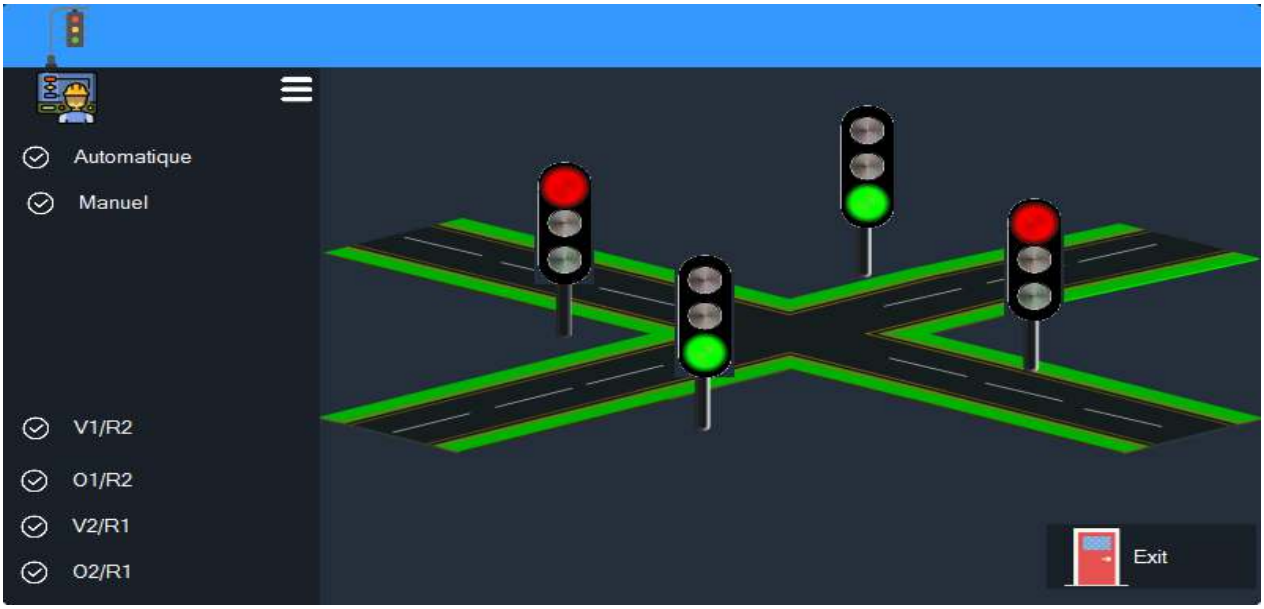


Figure 57 feu tricolore

Cette fenêtre nous donne une vue sur le feu tricolore, on a le choix entre deux mode, Automatique et Manuel qui nous permet de contrôler les feu manuellement

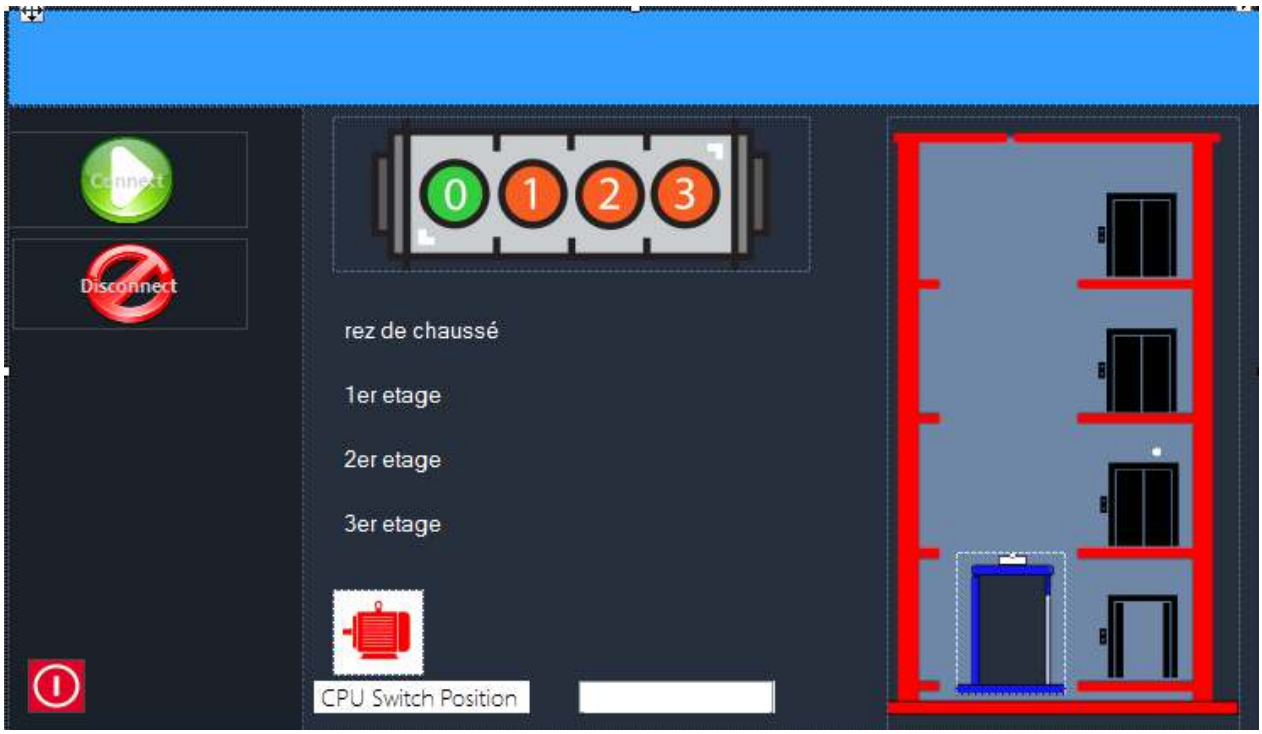


Figure 58 Ascenseur

Cette fenêtre nous donne une vue sur l'état de l'ascenseur et la possibilité de la contrôlera partir de cettefenêtre.

4.4 Conclusion :

Dans ce chapitre on a présenté notre travaille et les démarches suivie, en commençant par la définition du cahier de charge puis la traduction de ce dernier en grafcet et Lader et en fin la réalisation de l'interface

Ces étapes sont importantes à suivre pour n'importe quel projet à réaliser en automatisme, elle nous permettra d'avoir une vision globale sur l'ensemble du projet.

Les résultatsobtenus par les simulations sont proches de celle en réalité.

Conclusion générale

Durant le présent projet il nous a été demandé de développer une application permettant de communiquer avec deux API mise en réseau, une fois la communication établie il est possible de lire les entrées sorties de l'API qui sert à la supervision et le contrôle du processus ,

Notre toute première étape était d'écrire deux programmes réalisant deux applications différentes afin de développer une supervision autour de deux automates, cette étape nous a permit d'apprendre et se familiariser avec le logiciel Step7 TIA Portal v13

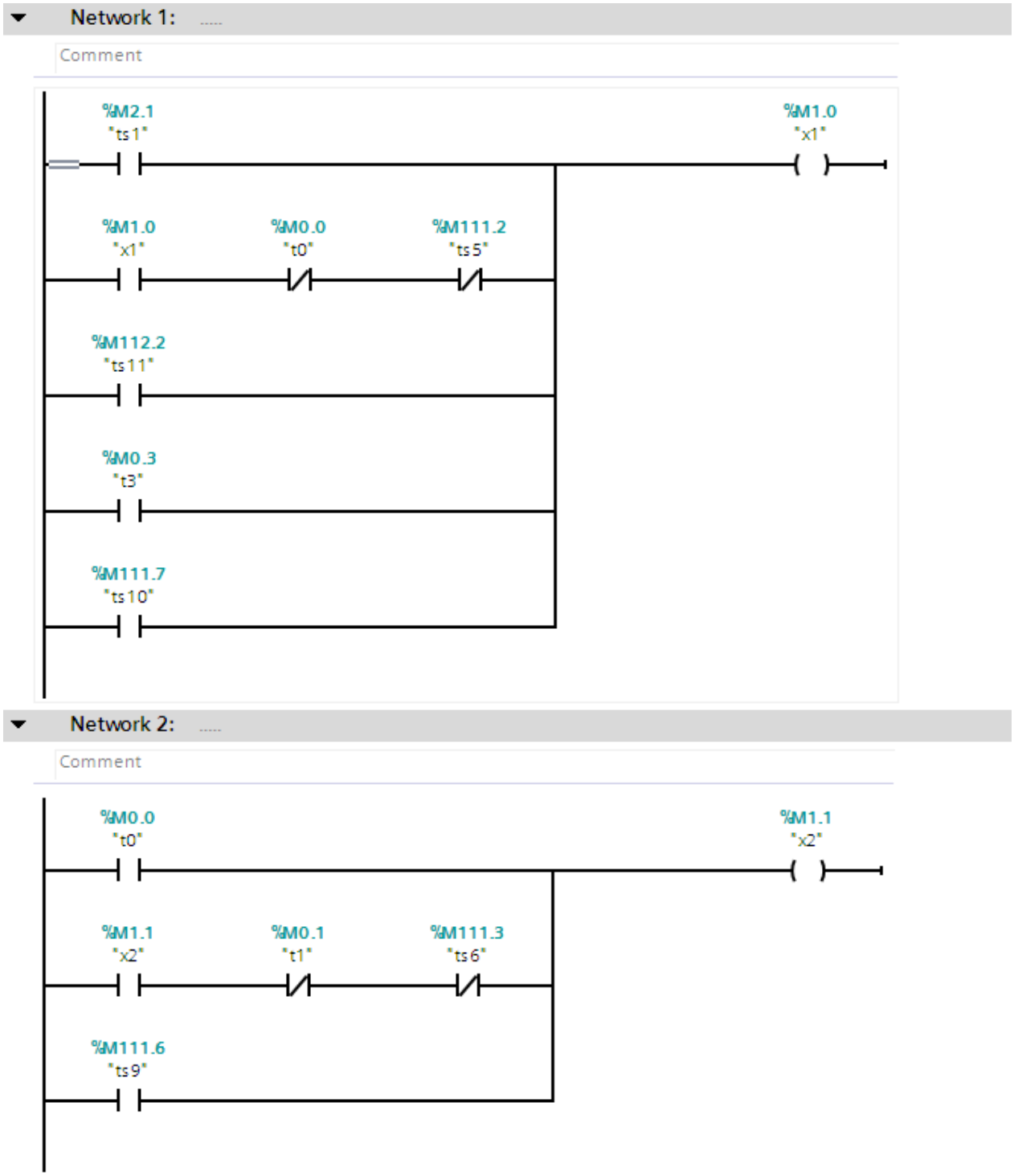
La seconde étape était de développer notre application, il nous a été proposé de travailler avec langage Java auquelle on avait de bonnes notions sur ce langage, il nous a été facile de développer une premiere interface graphique et exploiter ce langage a son maximum mais sans avoir un résultat finale à cause d'une incompatibilité entre la bibliothèque Java qui fournit l'accès au APIs et notre adaptateur qui relie la console de programmation avec les deux Automate , on aurais pu régler le problème simplement en remplaçant l'adaptateur mais les références compatible n'existe plus en marché .

On a pensé à faire le même travaille mais en simulant une communication entre la console de programmation et les APIs, pour la mise en place de la solution on est passé du Java vers C Sharp qui offre la possibilité de communiquer avec PLCSIM TIA Portal v13 en utilisant une

DLL (Dynamics Link Library), maintenant on peut avoir le même résultat qu'avec des Automates réels et connecter autant d'Automate qu'on désire les superviser tous à la fois.

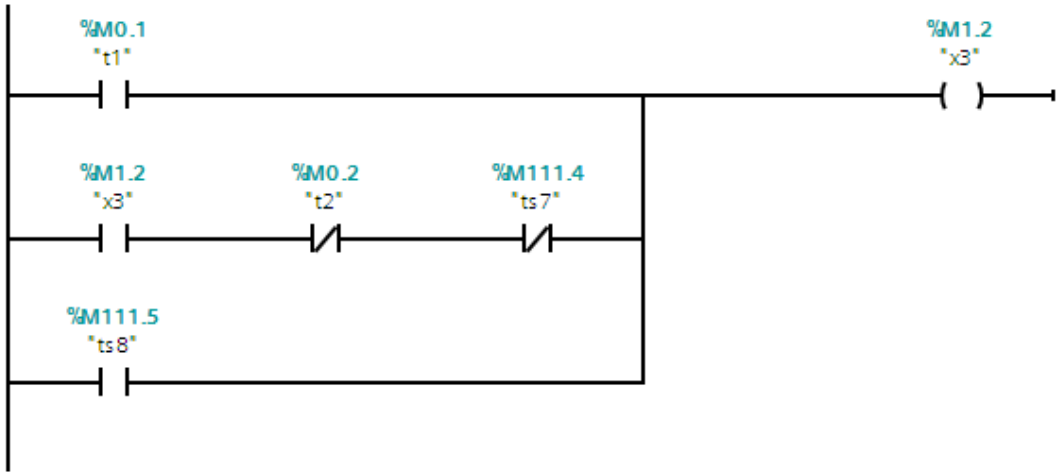
Au long ce travaille nous avons acquis plusieurs connaissance que ce soit sur le plans logiciels ou matériels et sur tout sur la façon dont ces deux derniers sont liés.

Annexes 1


Network 1:
Comment
%M2.1
"ts1"
%M1.0
"x1"
%M1.0
"x1"
%M0.0
"t0"
%M111.2
"ts5"
%M112.2
"ts11"
%M0.3
"t3"
%M111.7
"ts10"
Network 2:
Comment
%M0.0
"t0"
%M1.1
"x2"
%M1.1
"x2"
%M0.1
"t1"
%M111.3
"ts6"
%M111.6
"ts9"

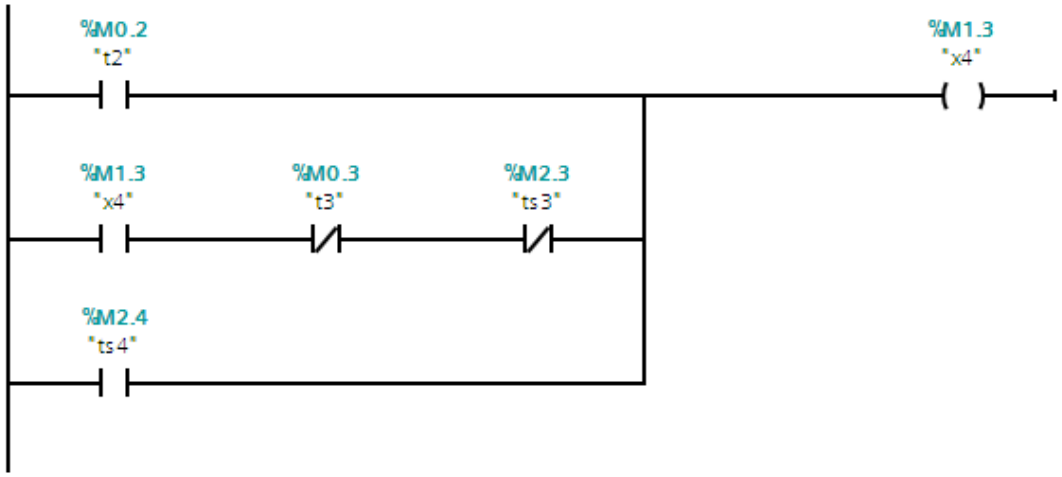
Network 3:

Comment



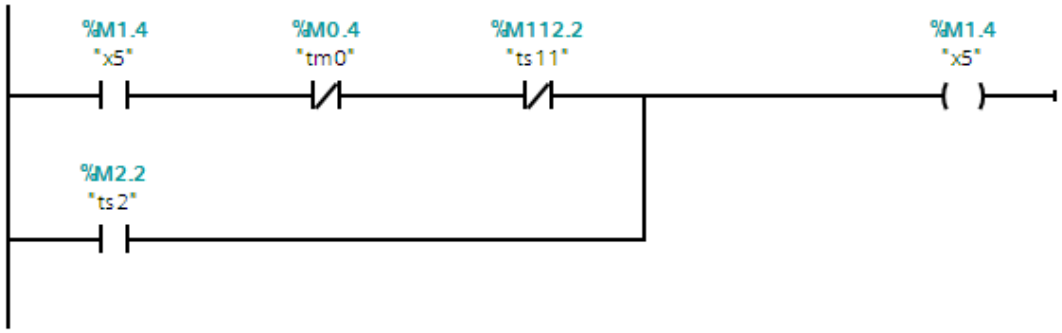
Network 4:

Comment



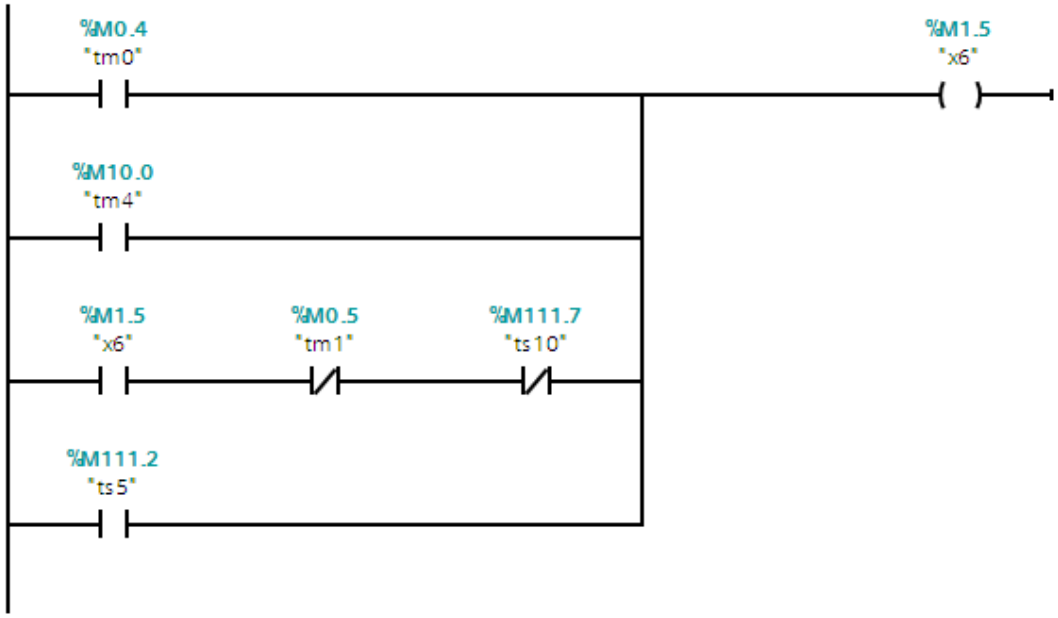
Network 5:

Comment



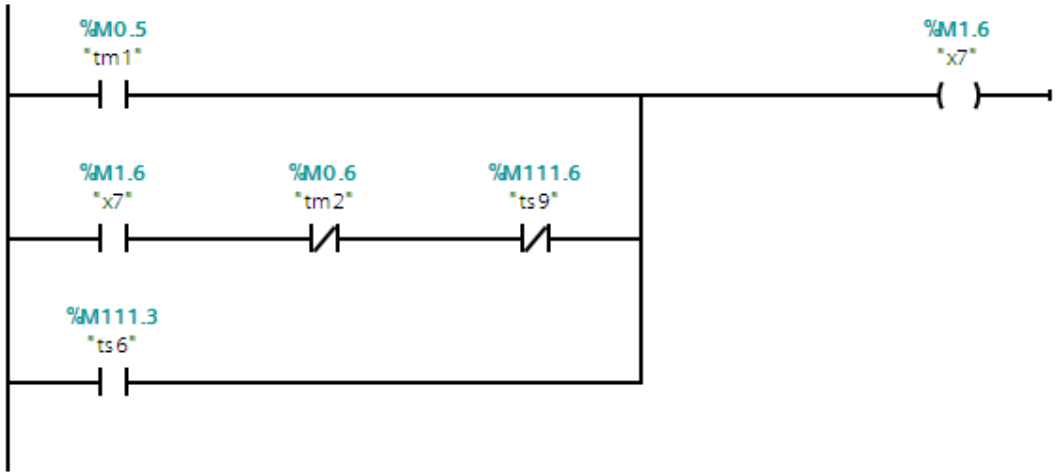
Network 6: .....

Comment



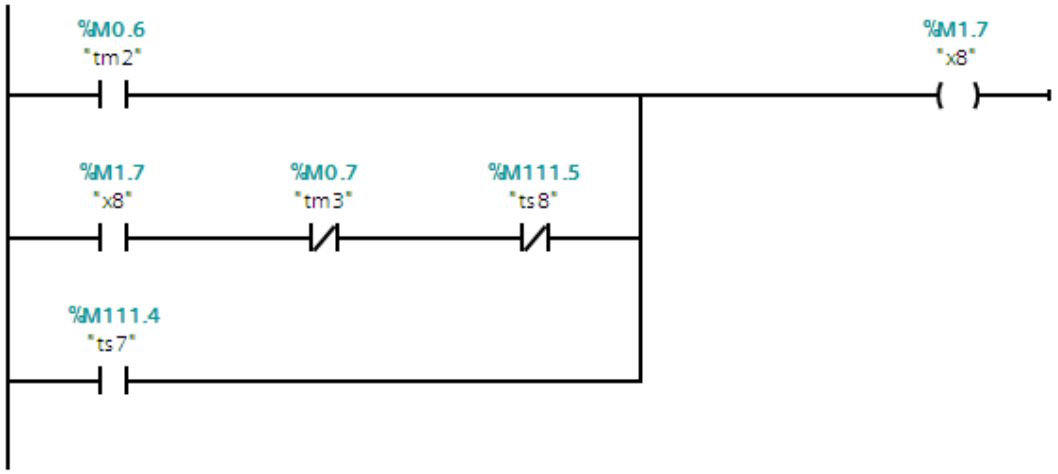
Network 7: .....

Comment



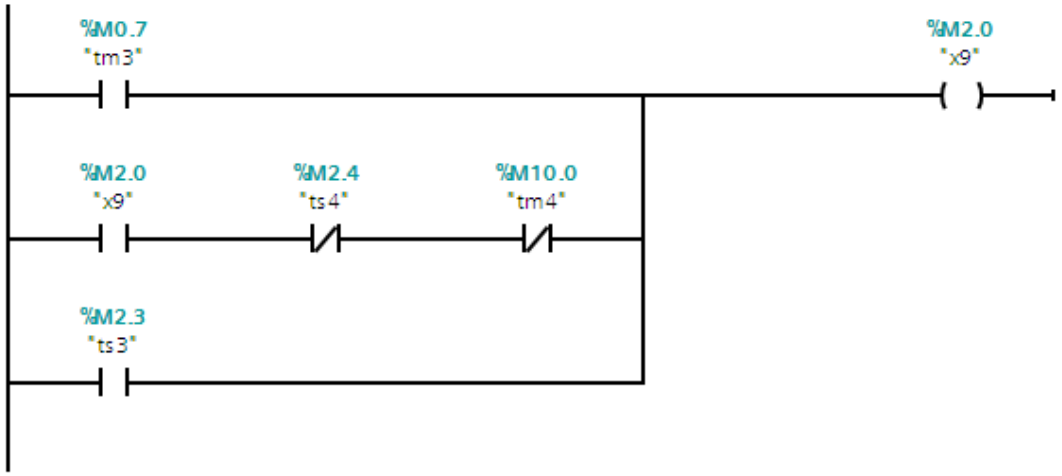
Network 8: .....

Comment



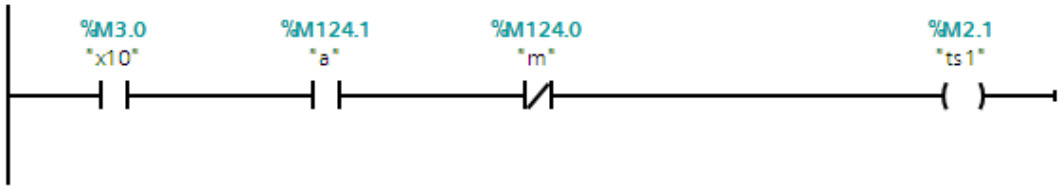
Network 9: .....

Comment



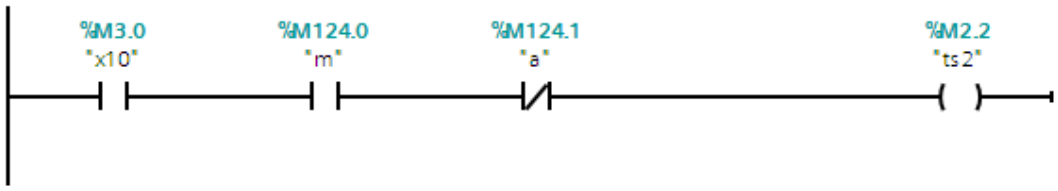
Network 10: .....

Comment



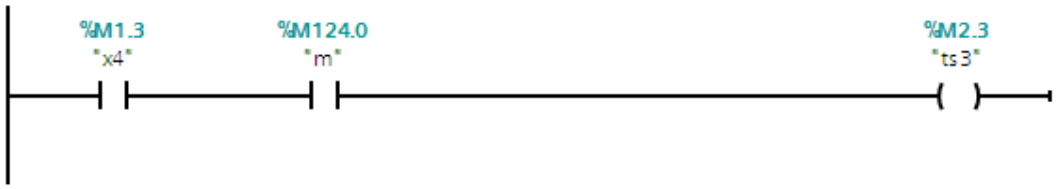
Network 11:

Comment



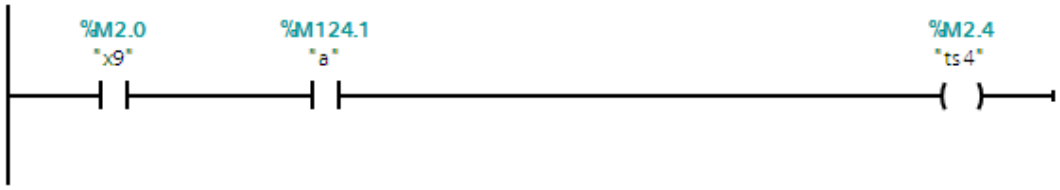
Network 12:

Comment



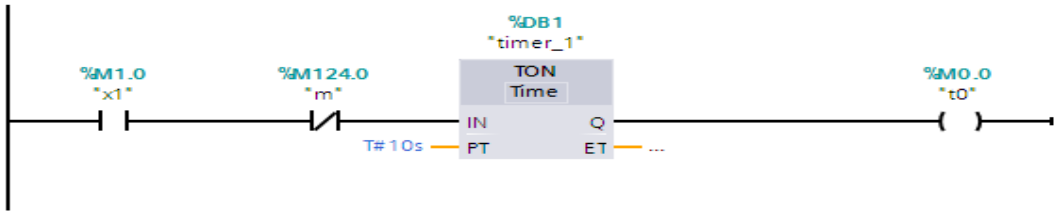
Network 13:

Comment



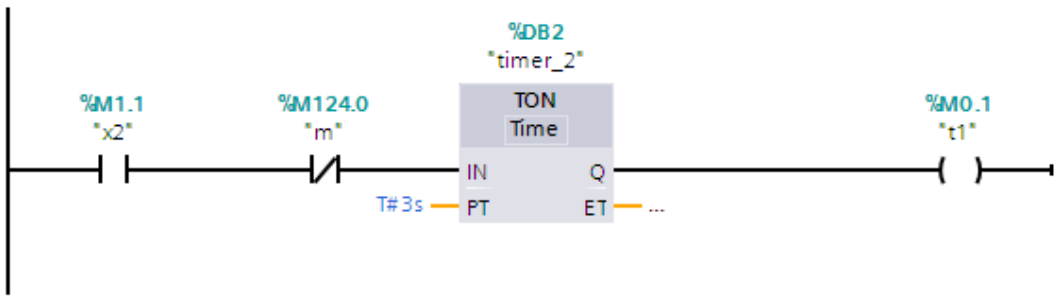
Network 14:

Comment



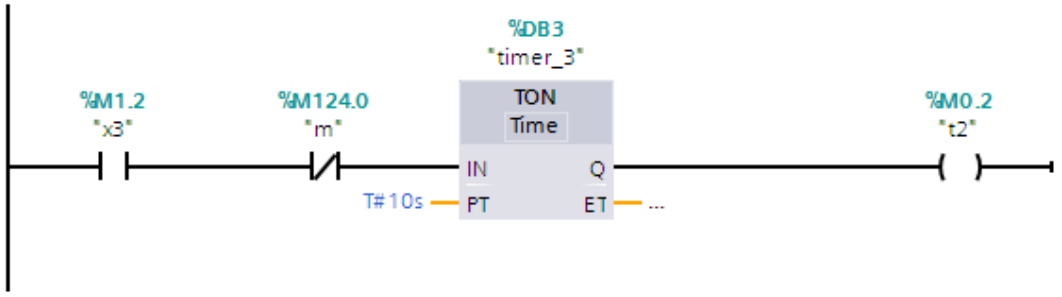
Network 15:

Comment



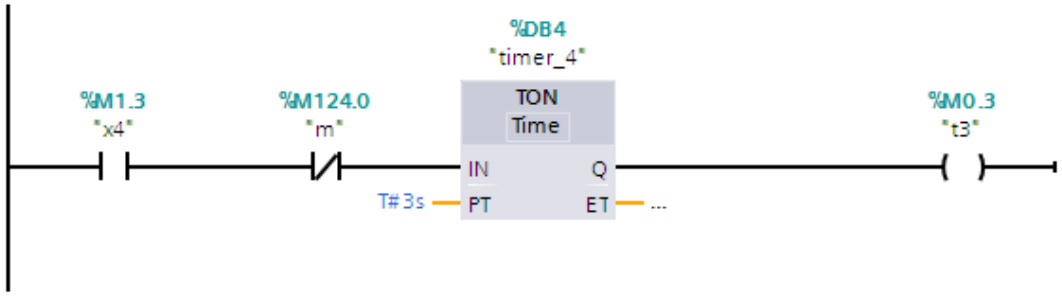
Network 16:

Comment



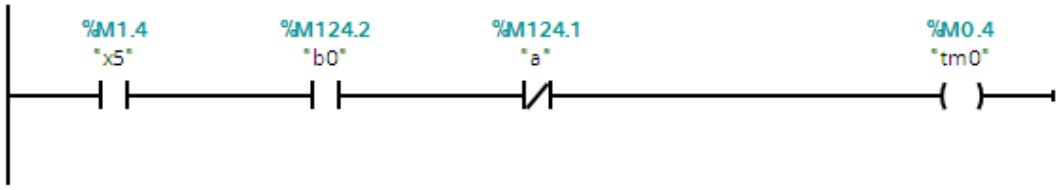
Network 17:

Comment



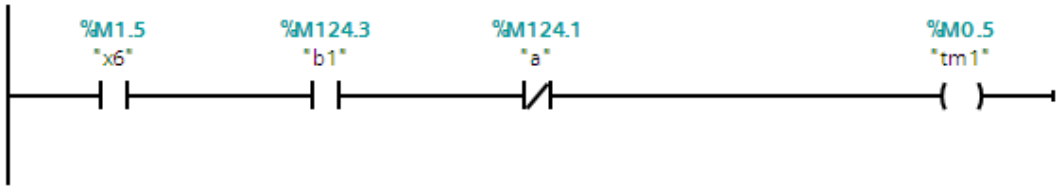
Network 18:

Comment



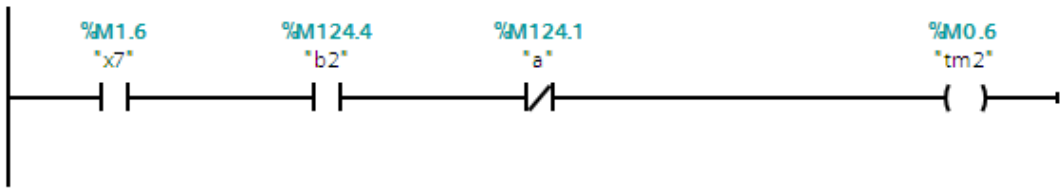
Network 19:

Comment



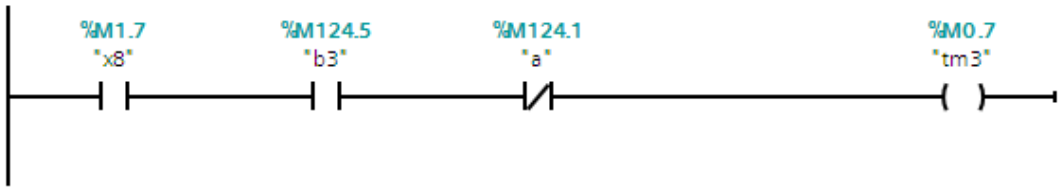
Network 20:

Comment



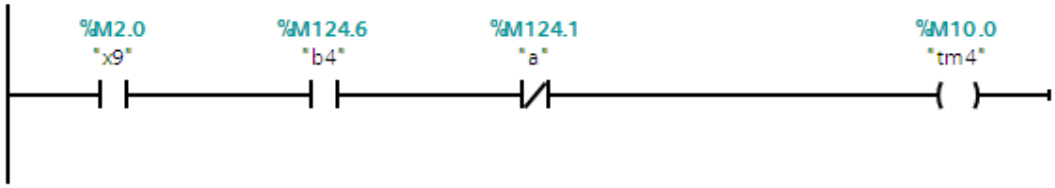
Network 21:

Comment



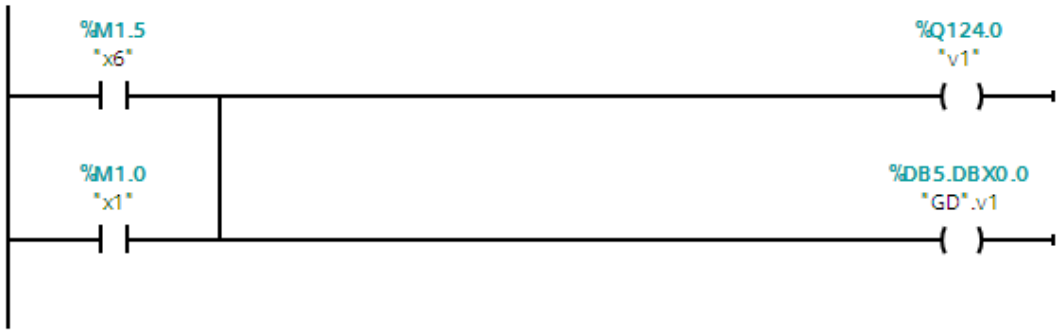
Network 22:

Comment



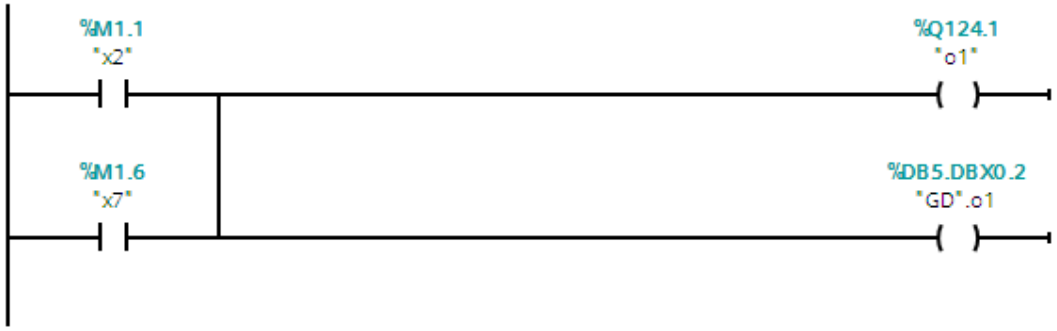
Network 23:

Comment



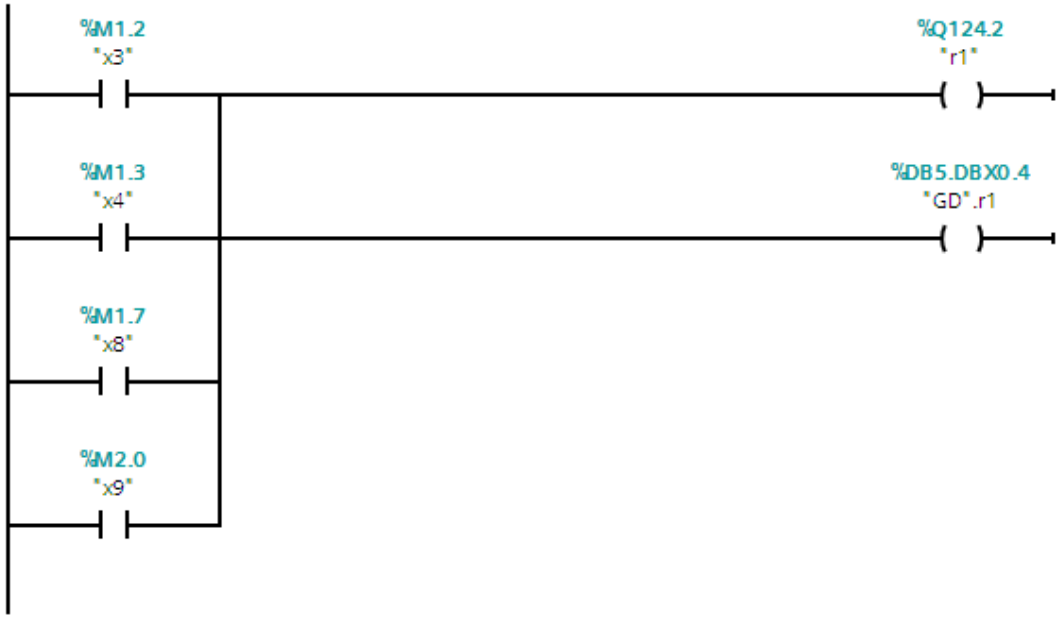
Network 24:

Comment



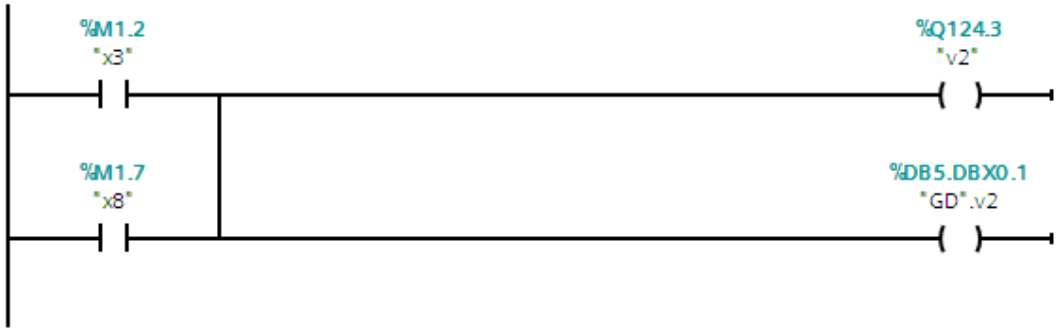
Network 25:

Comment



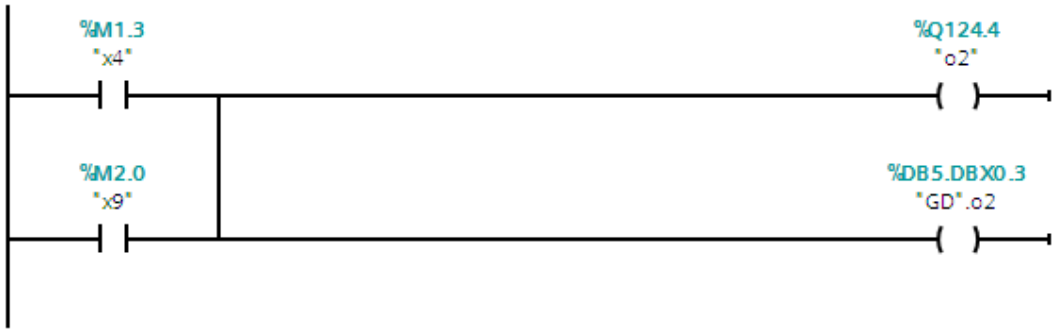
Network 26:

Comment



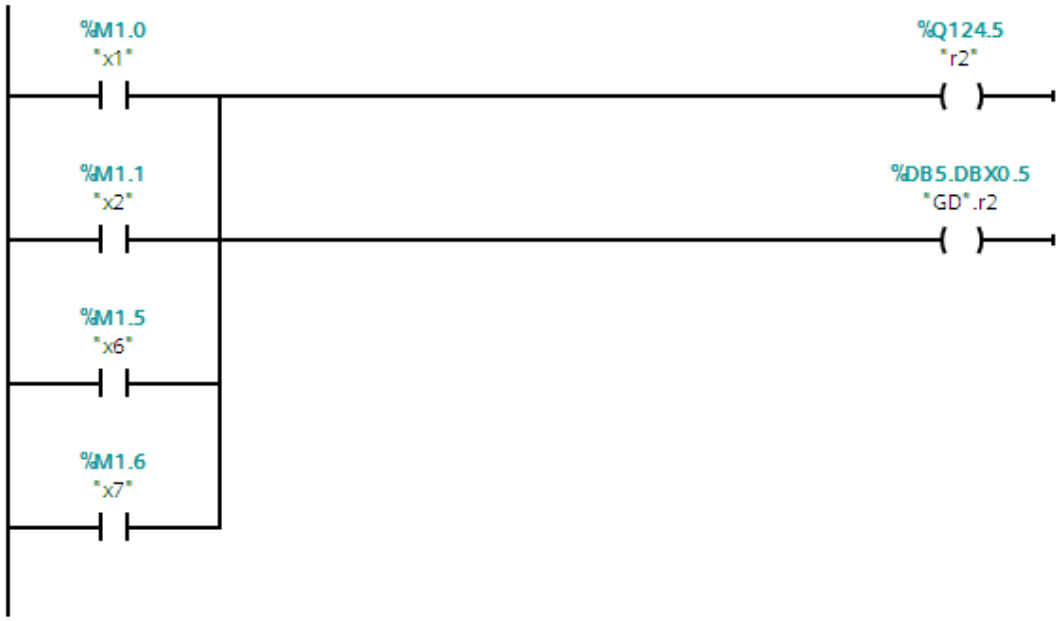
Network 27:

Comment



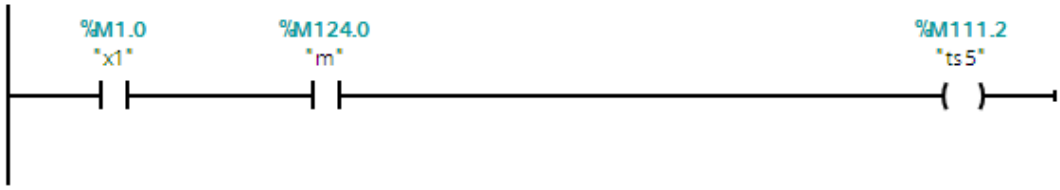
Network 28:

Comment



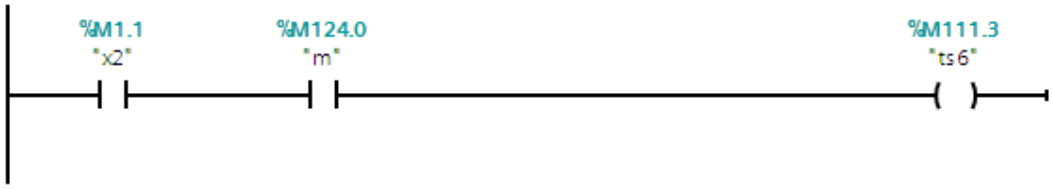
Network 29:

Comment



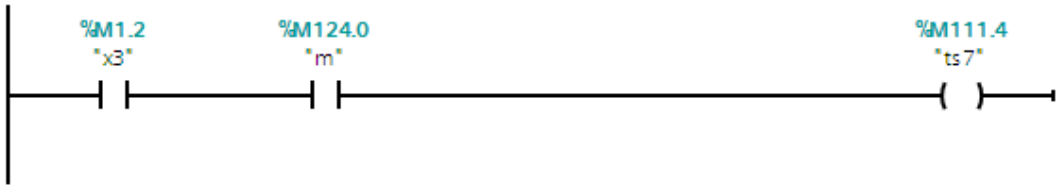
Network 30: .....

Comment



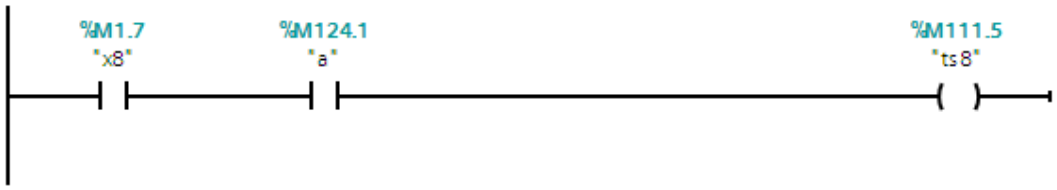
Network 31: .....

Comment



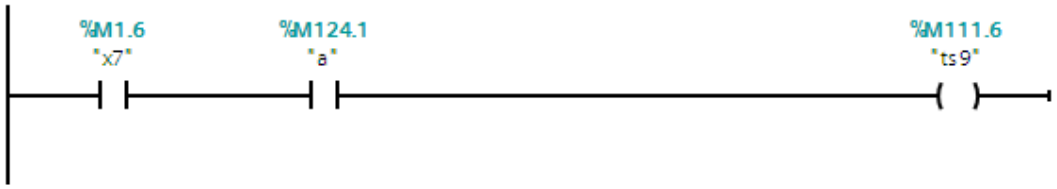
Network 32: .....

Comment



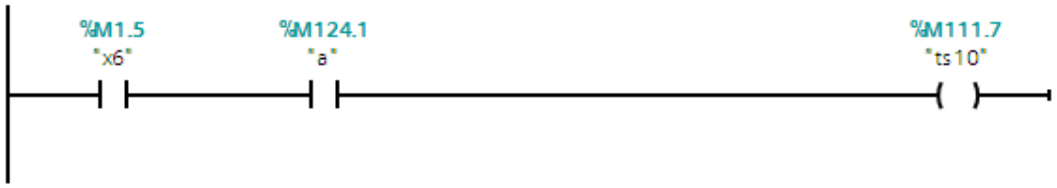
Network 33: .....

Comment



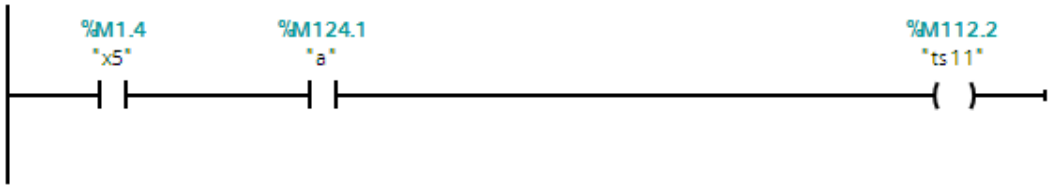
Network 34: .....

Comment



Network 35:

Comment

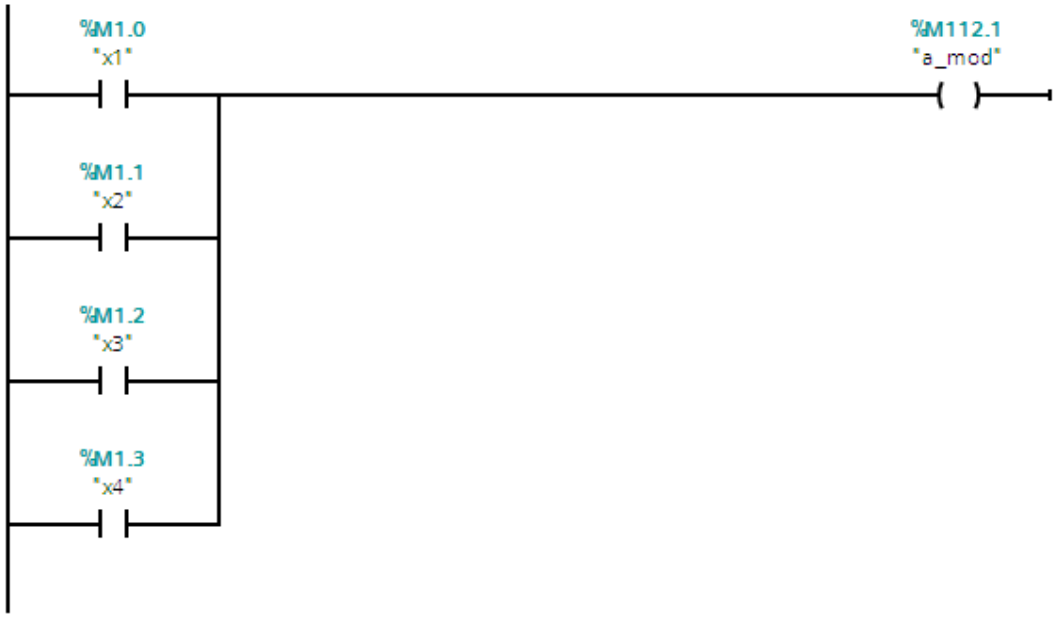


Network 36:

Comment

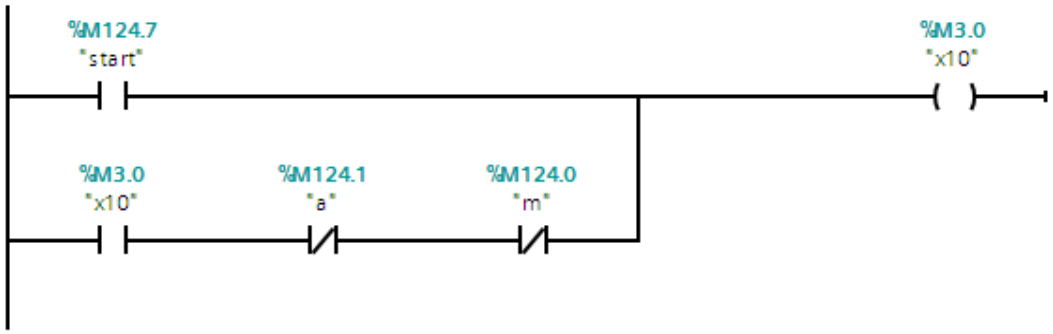
Network 37:

Comment



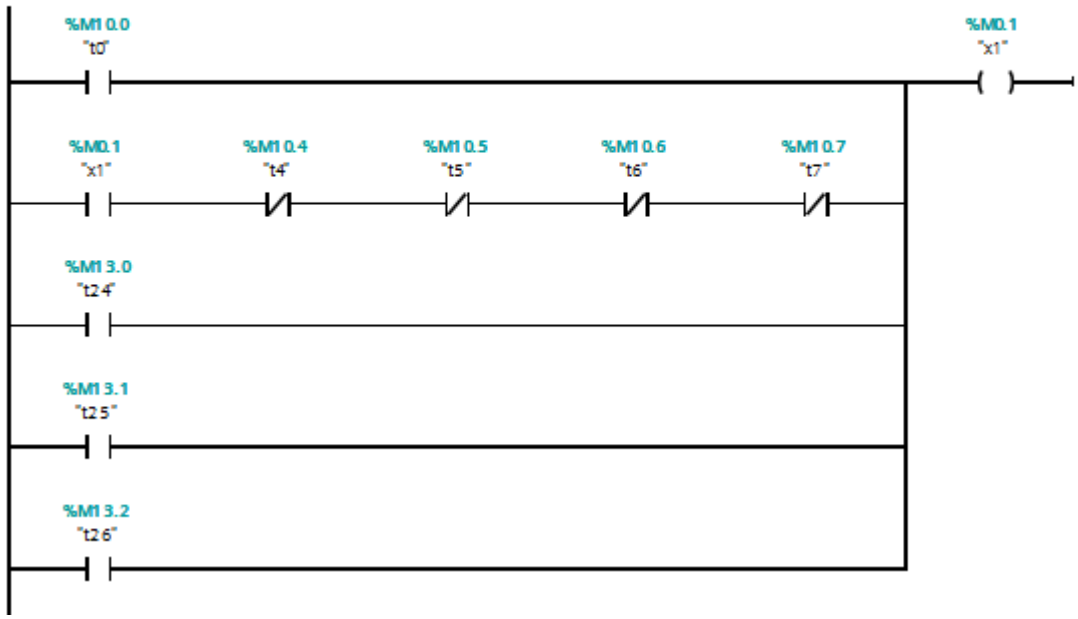
Network 38:

Comment



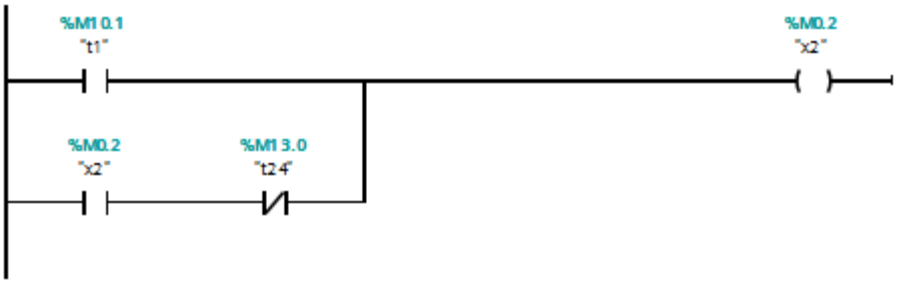
Network 1:

Comment



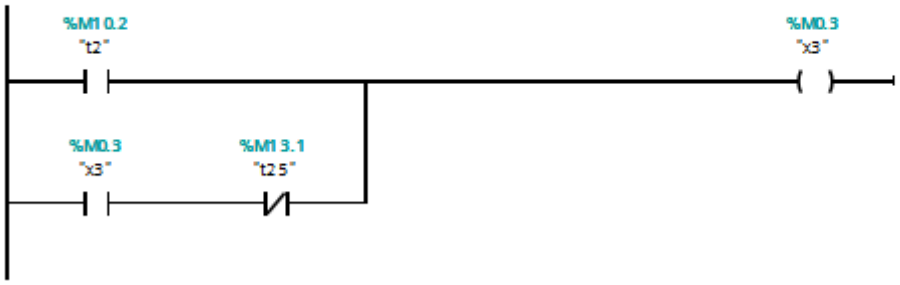
Network 2:

Comment

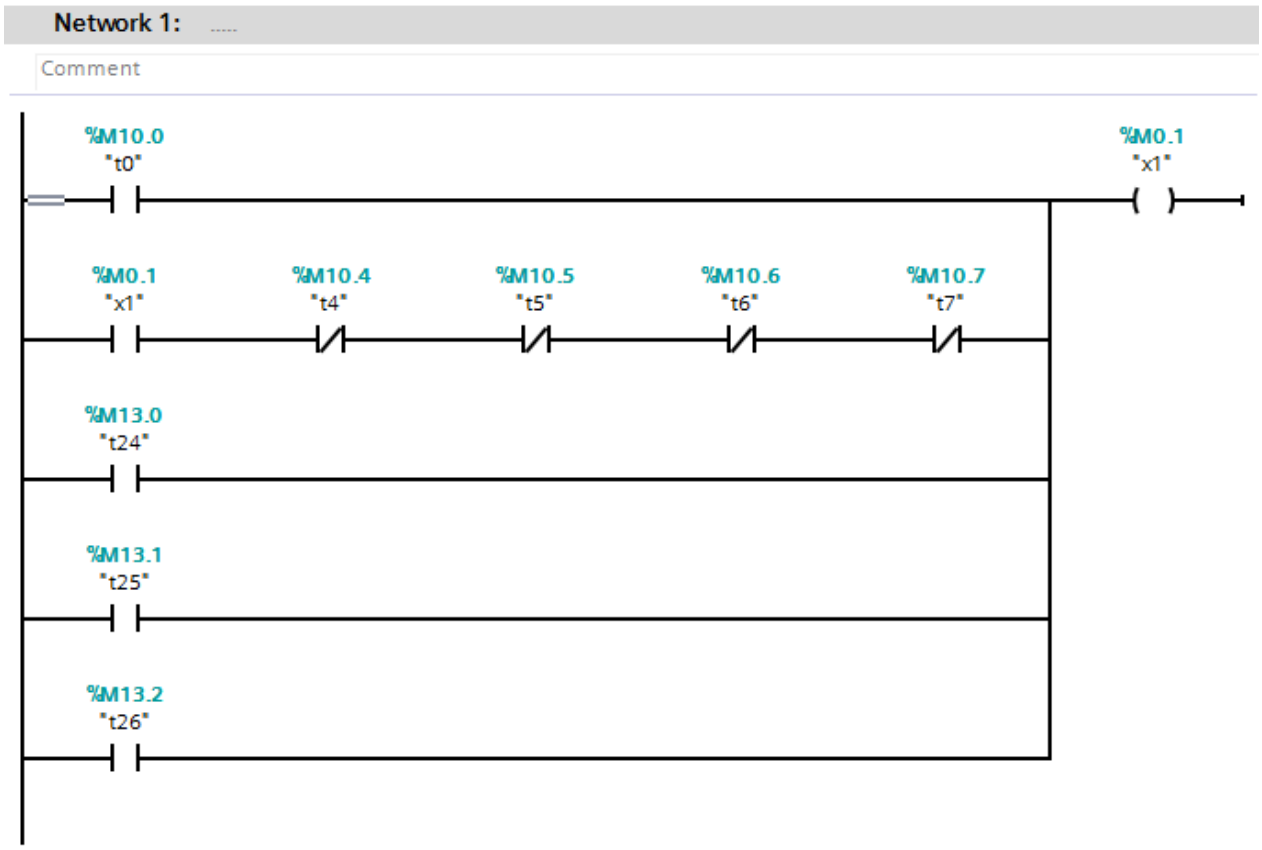


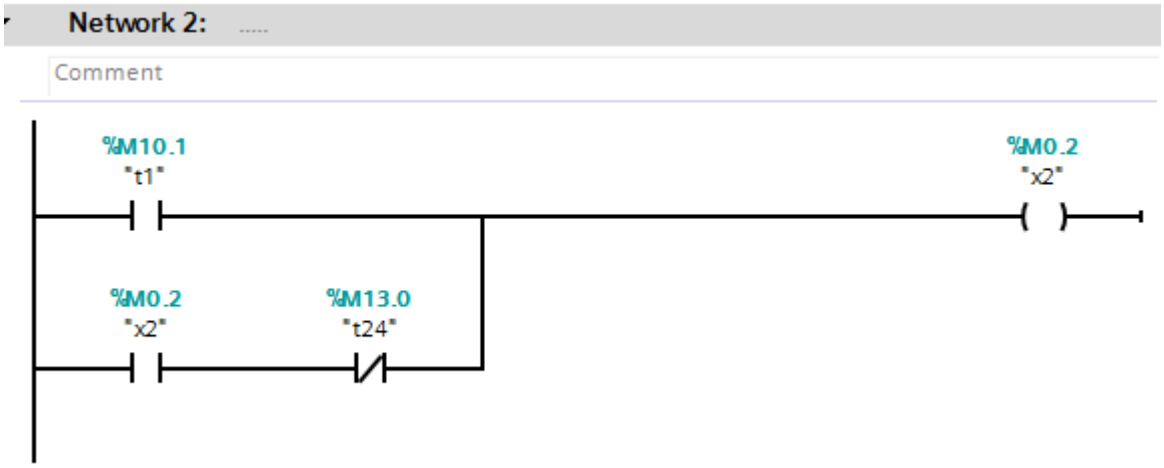
Network 3:

Comment



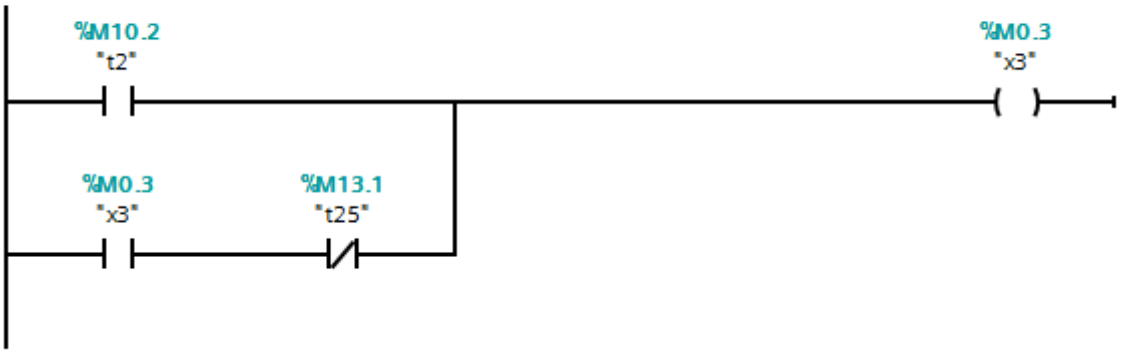
Annexe 2


Network 1:
Comment
%M10.0
"t0"
%M0.1
"x1"
%M0.1
"x1"
%M10.4
"t4"
%M10.5
"t5"
%M10.6
"t6"
%M10.7
"t7"
%M13.0
"t24"
%M13.1
"t25"
%M13.2
"t26"


Network 2:
Comment
%M10.1
"t1"
%M0.2
"x2"
%M0.2
"x2"
%M13.0
"t24"

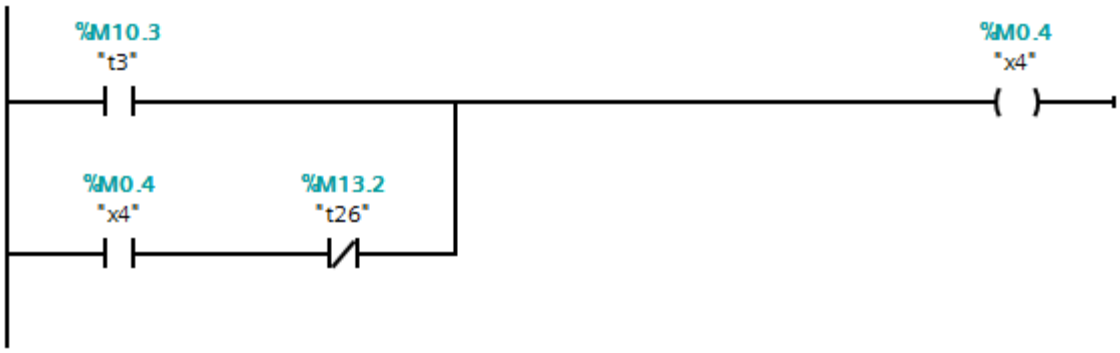
Network 3: .....

Comment



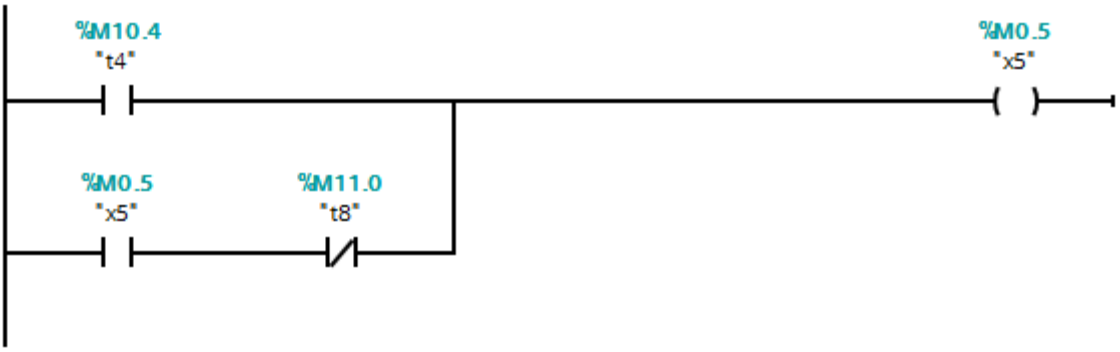
Network 4: .....

Comment



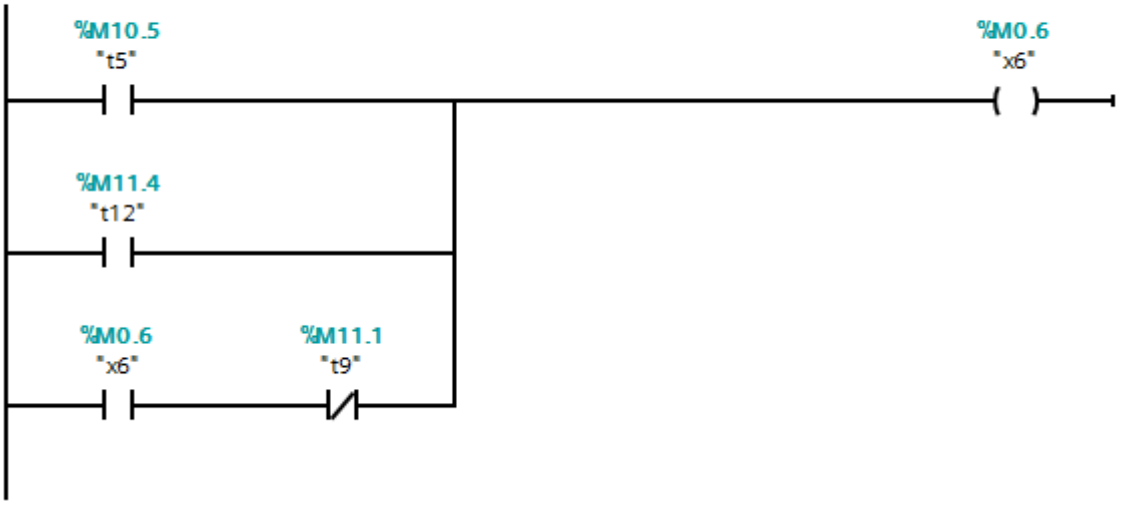
Network 5: .....

Comment



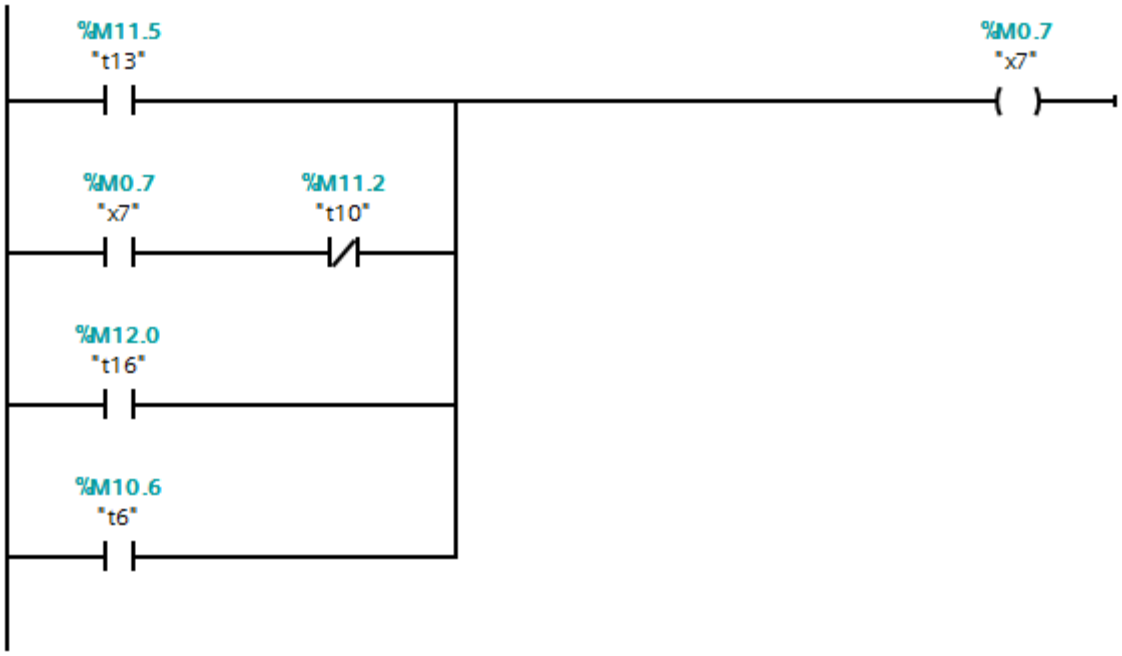
Network 6:

Comment

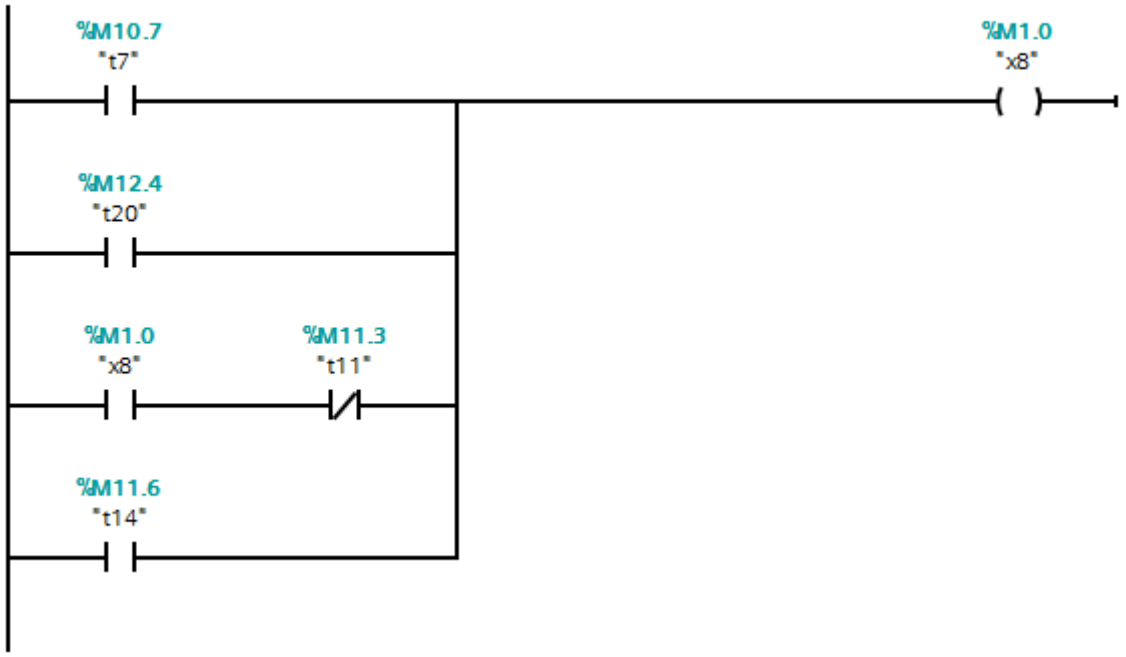


Network 7:

Comment

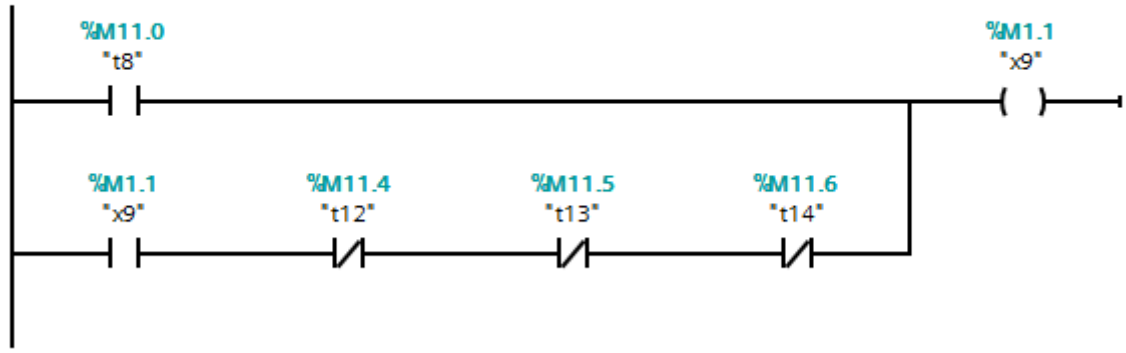


Comment



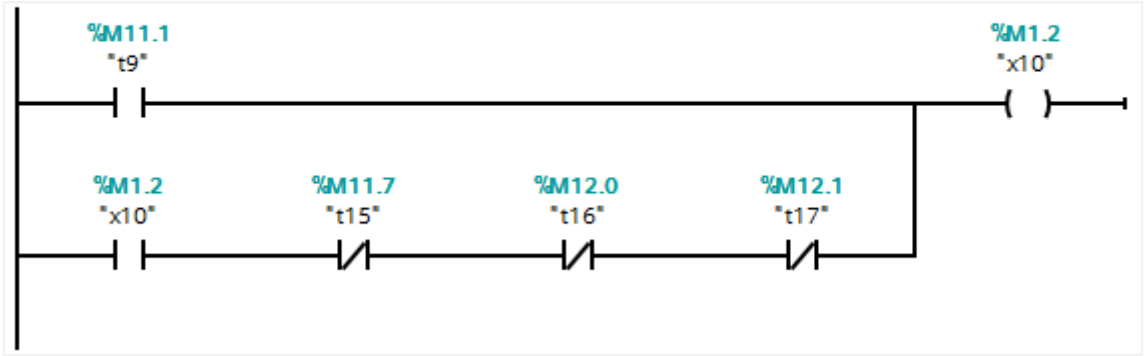
Network 9: .....

Comment



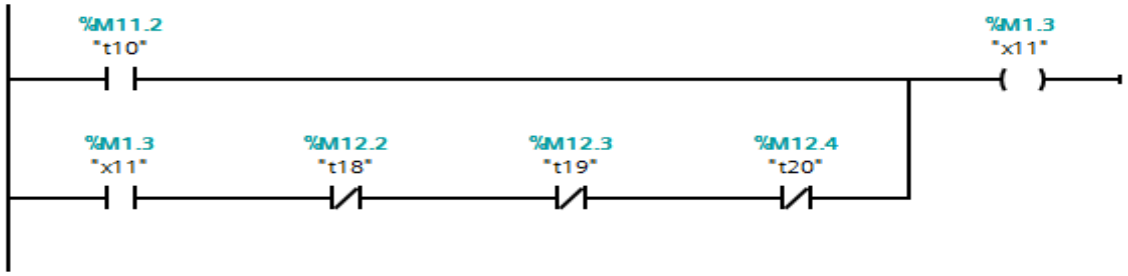
Network 10: .....

Comment



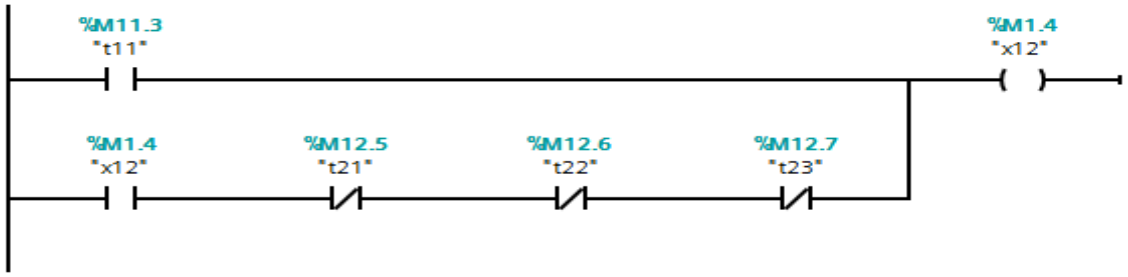
Network 11: .....

Comment



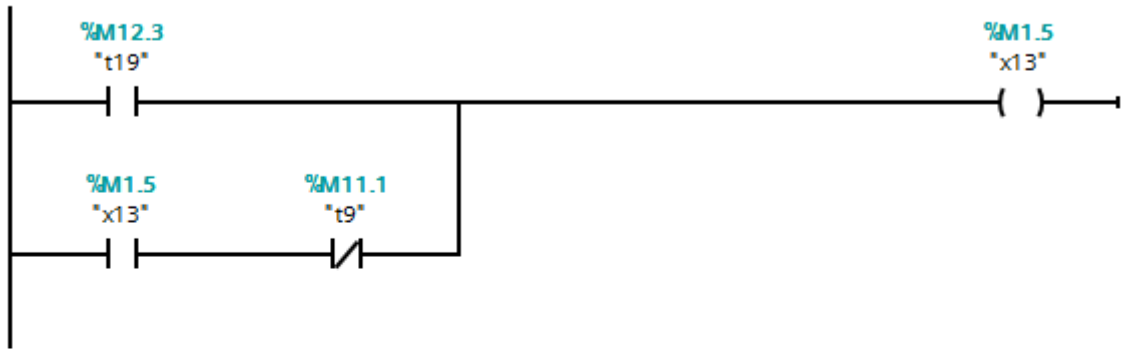
Network 12: .....

Comment



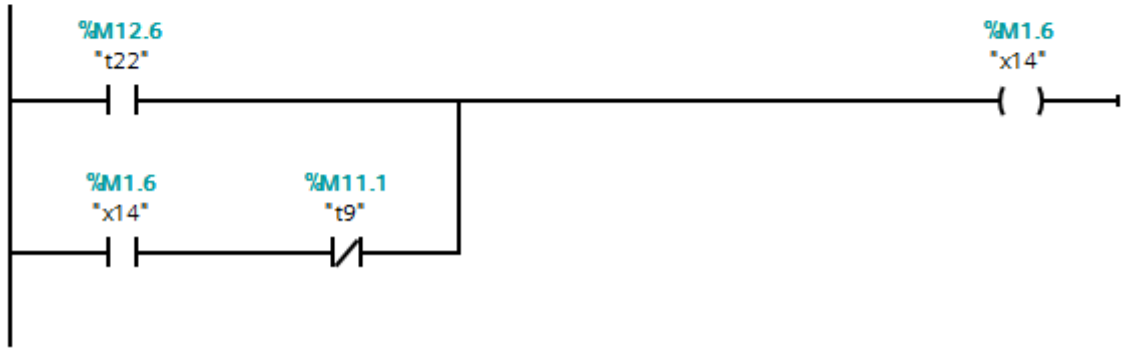
Network 13: .....

Comment



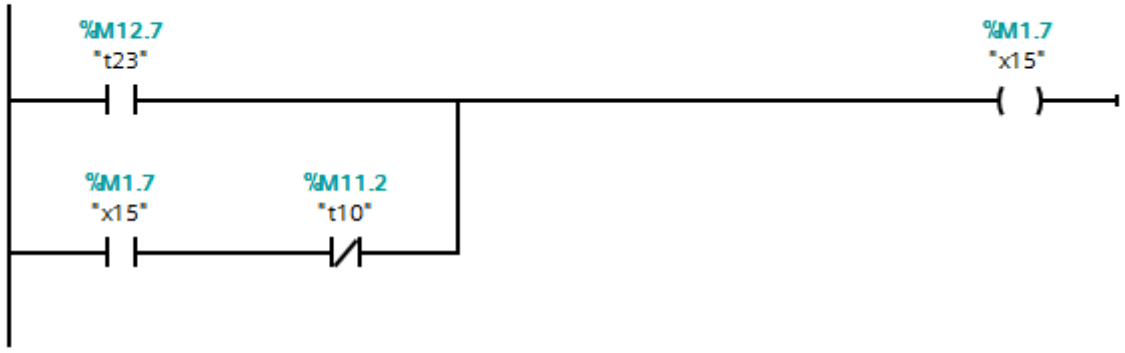
Network 14: .....

Comment



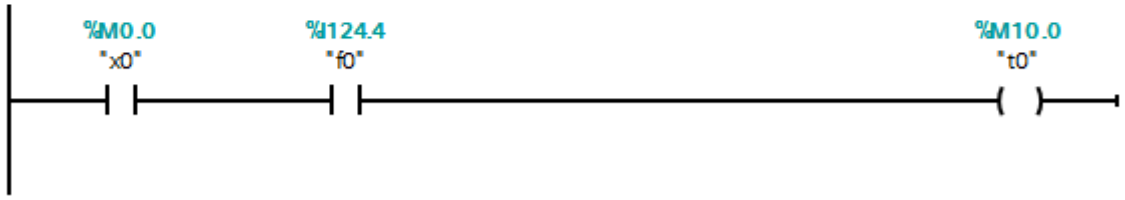
Network 15: .....

Comment



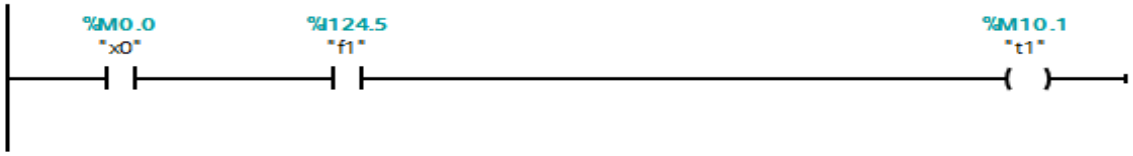
Network 16: .....

Comment



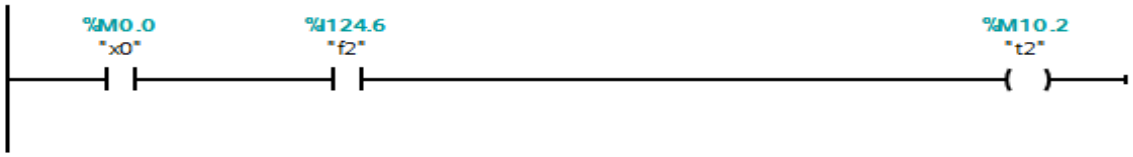
Network 17: .....

Comment



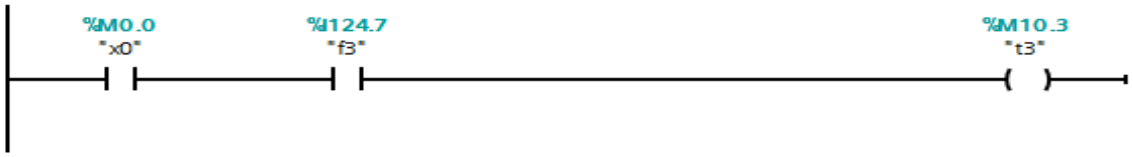
Network 18: .....

Comment



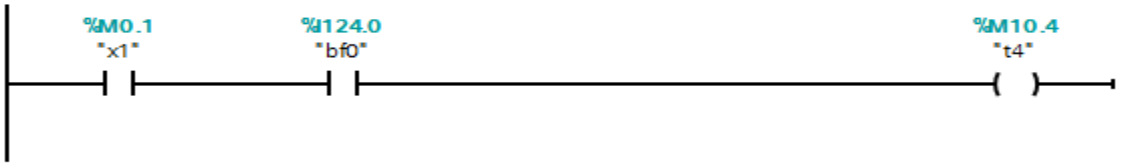
Network 19: .....

Comment



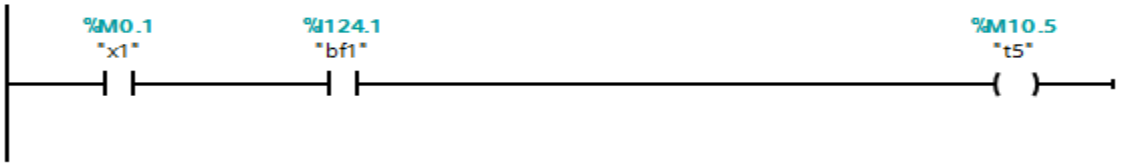
Network 20: .....

Comment



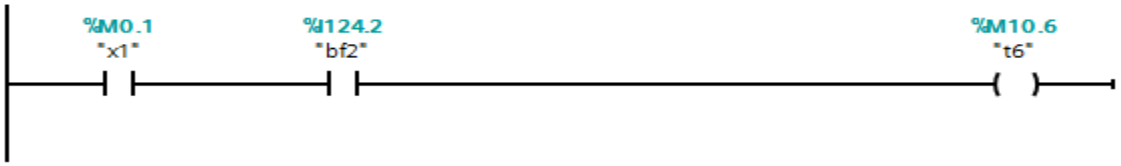
Network 21: .....

Comment



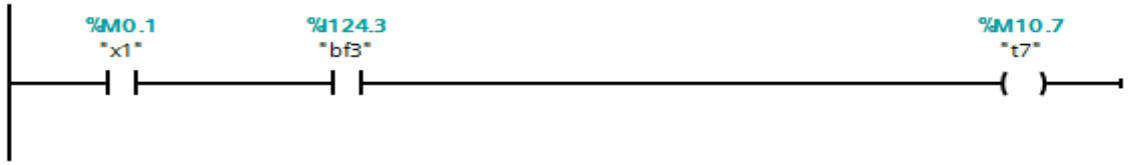
Network 22: .....

Comment



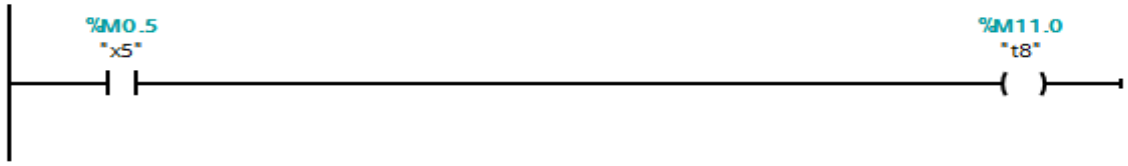
Network 23: .....

Comment



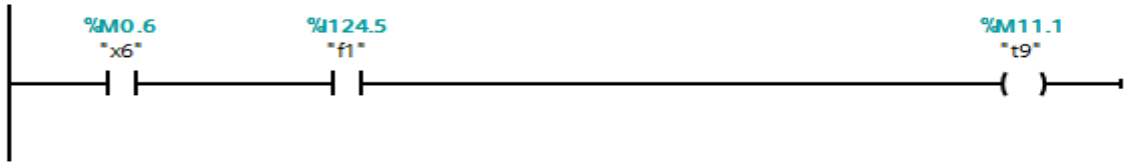
Network 24: .....

Comment



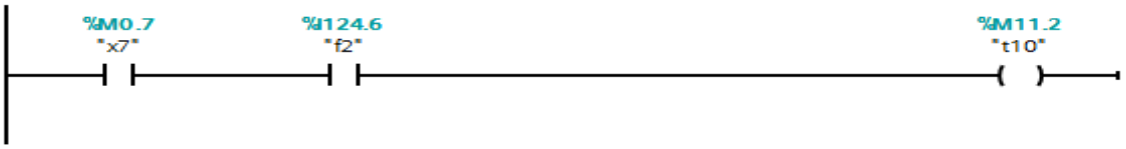
Network 25: .....

Comment



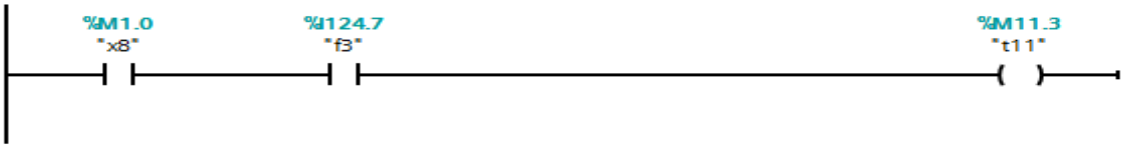
Network 26: .....

Comment



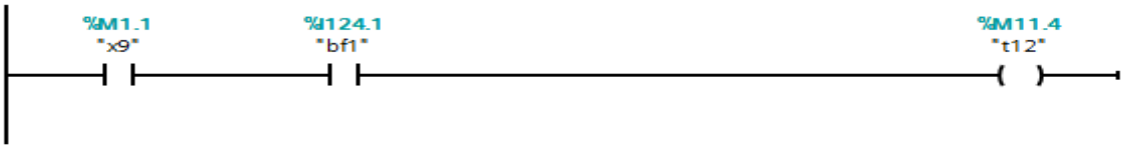
Network 27: .....

Comment



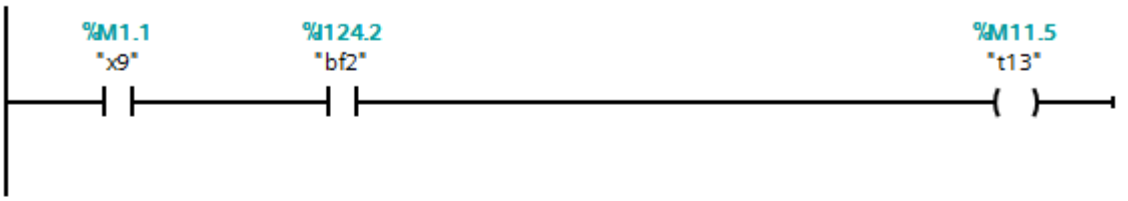
Network 28: .....

Comment



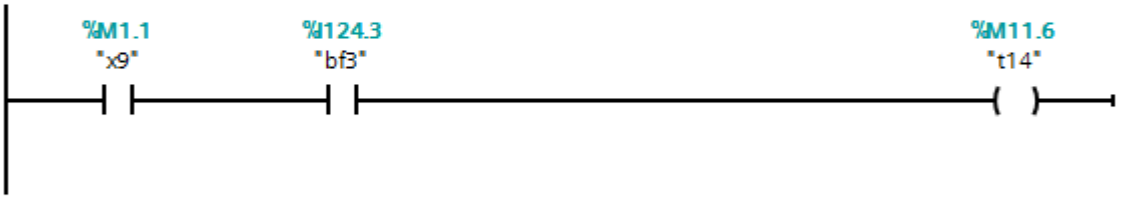
Network 29: .....

Comment



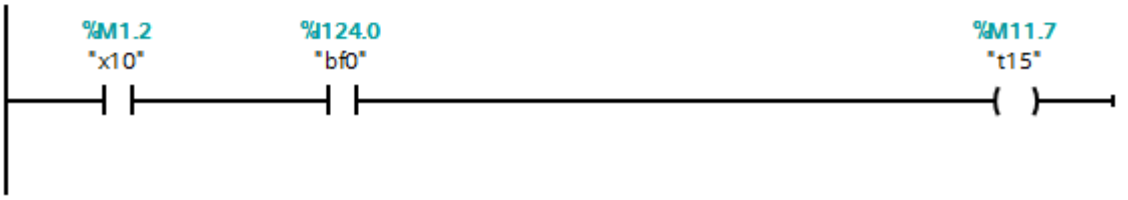
Network 30: .....

Comment



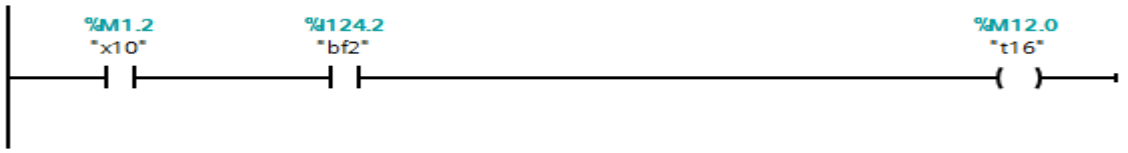
Network 31: .....

Comment



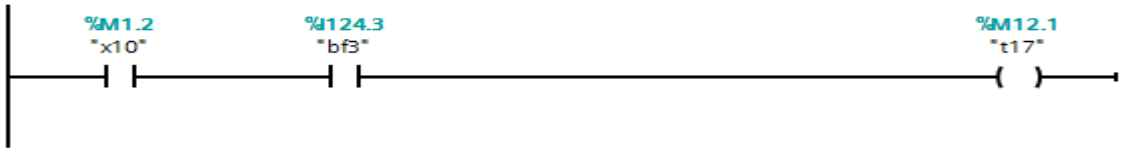
Network 32: .....

Comment



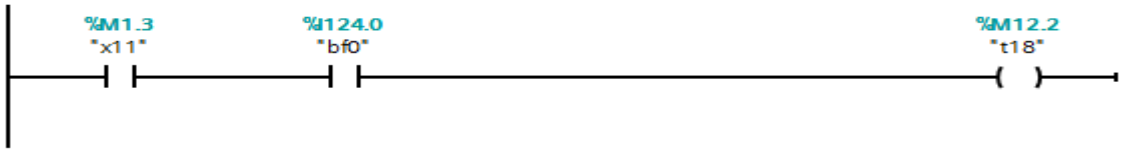
Network 33: .....

Comment



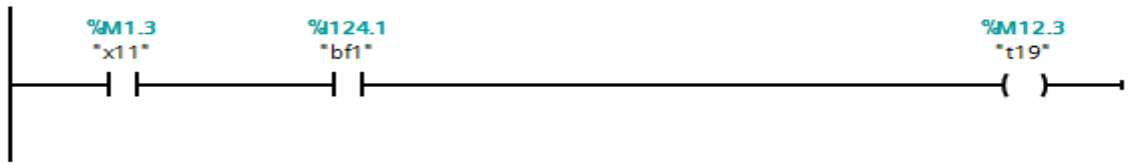
Network 34: .....

Comment



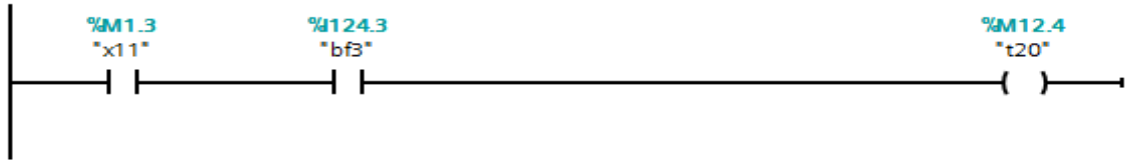
Network 35: .....

Comment



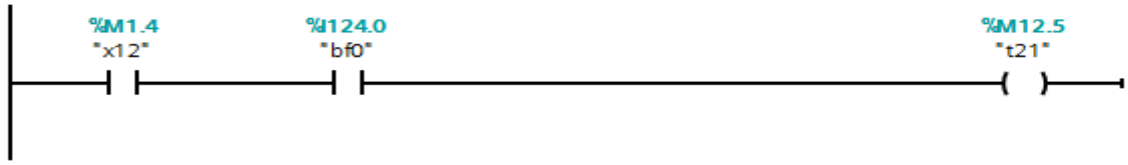
Network 36: .....

Comment



Network 37: .....

Comment

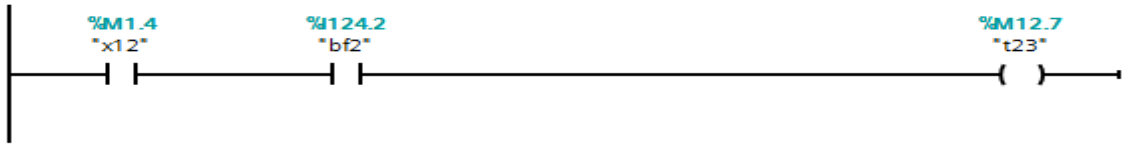


Network 38:

Comment

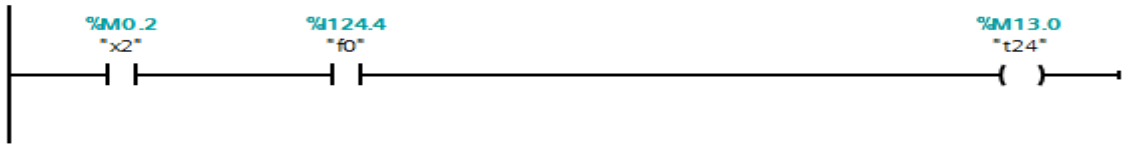
Network 39:

Comment



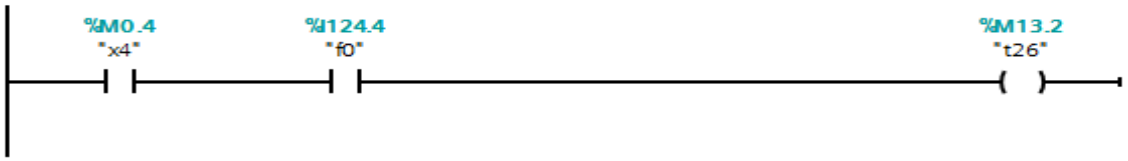
Network 40:

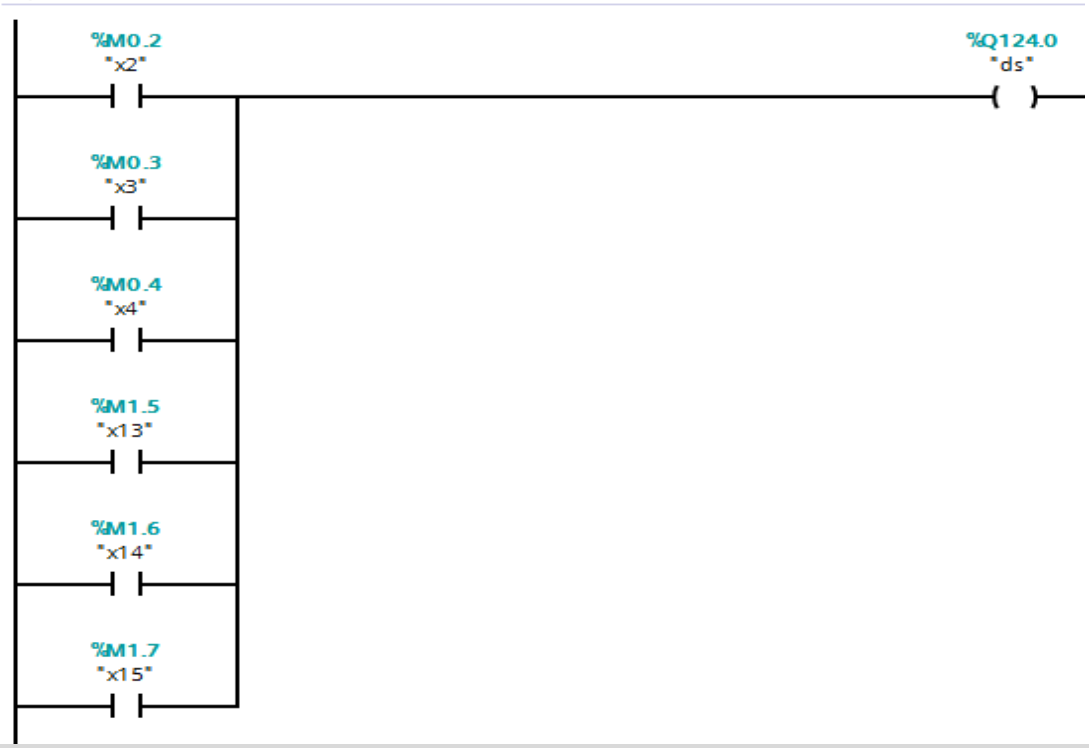
Comment



Network 42:

Comment



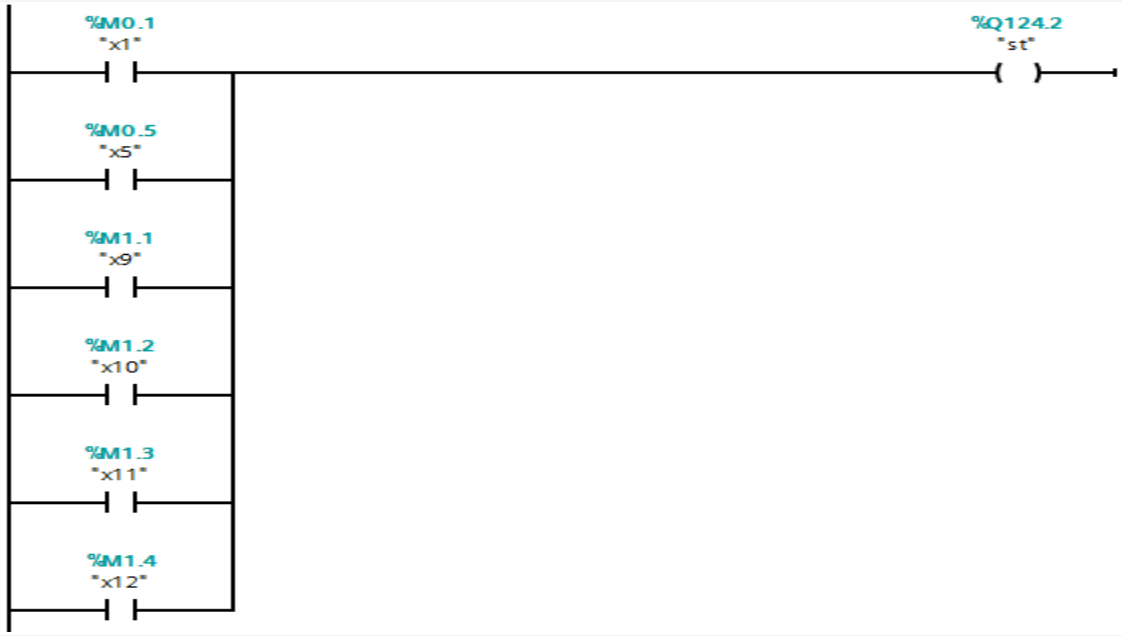


Network 44:

Comment

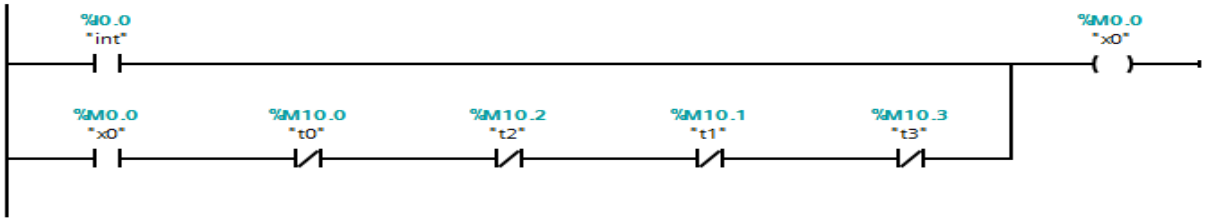
Network 45:

Comment



Network 46:

Comment



Bibliographie

[1]SIMATIC PC Adapter TS Adapter Quick Reference Guide Edition 01/2003 A5E00078070-02

[2]ImanMorsi, LoayMohy El-Din, SCADA system for oil refinery control, Measurement Vol. 47, Page 5-13, 2014

[3]université de Liège:Faculté des Sciences Appliquées, les automates programmable Tome I. Caractéristique et méthodologie de programmation DR.IR.H. LECOCQ (mise ajour 2005)

[4]https://en.wikipedia.org/wiki/SCADA

[5]https://en.wikipedia.org/wiki/Network_topology

[6]http://www.prosofttechnology.com/var/plain_site/storage/images/media/images/sche___matic-diagrams/an-x-pb/an-x-pb-schematic/180868-1-eng-US/AN-X-PB-Schematic.png

[7]http://iebmedia.com/images/art_images/ieb35asi.gif

[8]https://i.ytimg.com/vi/oPa6fw0if1c/hqdefault.jpg

[9] ttp://jp.mathworks.com/cmsimages/69942_wl_SimulinkPLCCoder_Figure8_wl.jpg

[10] https://support.industry.siemens.com/cs/attachments/1139855/S7WSPSCB.pdf

[11]https://alexsentcha.wordpress.com/